

# Grundprogramm 2018

Rolf-Dieter Klein ([www.rdklein.de](http://www.rdklein.de))  
Andres Rohmann ([www.ndr-nkc.de](http://www.ndr-nkc.de))  
Steffen Reimer (DL2LCE)

## Vorwort

Sowohl der NDR Klein Computer als auch der Z80 Mikroprozessor sind immer noch ungebrochen beliebt bei Fans und sogar bei manchen Neueinsteigern.

Schon zum 30-jährigen Jubiläum hatte ich mich entschlossen, das ursprüngliche Grundprogramm von Rolf-Dieter Klein zu aktualisieren, damit Daten mittels eines USB-Sticks zwischen dem PC und dem NKC übertragen werden können.

Im Herbst 2017 nahm Steffen Kontakt zu mir auf um mich auf einige Probleme und Fehler hinzuweisen. Relativ schnell haben wir beschlossen, gemeinsam an der weiteren Verbesserung zu arbeiten und ein vollkommen überarbeitetes Grundprogramm herauszugeben.

Schnell war klar, dass die geplanten Erweiterungen nicht in einem einzigen ROM untergebracht werden können und wir entschlossen zu einer Version mit zwei Speicherbausteinen.

Besonders möchte ich an dieser Stelle die hervorragende Zusammenarbeit mit Steffen hervorheben. Ohne seine konstruktiven Vorschläge und die von ihm codierten Teile des Grundprogramms wäre dieses Projekt nicht möglich gewesen. Wir bedanken uns außerdem bei Jens, der uns auf einige Fehler in der ersten Version hingewiesen hat.

Wir wünschen Ihnen / Euch viel Spaß bei der Nutzung des neuen Systems.

## Systemvoraussetzungen

Das Grundprogramm kann flexibel mit verschiedenen Hardwarekonfigurationen eingesetzt werden. Grundsätzlich wird jedoch eine Vollausbau CPU benötigt, das Grundprogramm kann nicht auf der Baugruppe SBC2 eingesetzt werden. Außerdem werden immer eine KEY Baugruppe und eine Grafikausgabe GDP64K oder GDP64HS benötigt.

### SPEICHER

Als Minimalausstattung kann das Grundprogramm auf den ersten beiden Steckplätzen einer ROA64 Speicherbaugruppe zusammen mit 16 kB RAM ab Adresse 6000h eingesetzt werden. Zu empfehlen ist jedoch ein Ausbau mit mehr Speicher.

### BANKBOOT (optional)

Auf einer BANKBOOT Baugruppe kann das Grundprogramm ebenfalls auf den ersten beiden Plätzen eingesetzt werden. In diesem Fall müssen zusätzlich mindestens 8 kB RAM auf dem vierten Steckplatz der BANKBOOT installiert werden. Zusätzlich wird eine Baugruppe ROA64 mit 64 kB RAM oder eine SRAM1024 mit mindestens 512 kB benötigt.

Einige Funktionen des Grundprogramms sind nur mit eingesetzter BANKBOOT Baugruppe verfügbar. Bei Verwendung einer BANKBOOT Baugruppe können neue Betriebssysteme von USB oder CAS geladen und gestartet werden.

Die aktuell eingestellte Banknummer wird in der Kopfzeile angezeigt. Da die BANKBOOT Baugruppe keine Rückmeldung erlaubt welche Bank eingestellt ist, ist die Anzeige nur korrekt, wenn zuvor Bank wechseln oder Bank einblenden ausgeführt wurde.

### USB (optional)

Wie schon in der Version 3.0 des Grundprogramms wird das USB Modul VDIP1 unterstützt. Dabei wurden einige Fehler behoben, die Nutzungsmöglichkeiten erweitert und die Funktionalitäten für eigene Programme zur Verfügung gestellt. Das Grundprogramm unterstützt die Anzeige des Inhalts eines USB-Laufwerks und das Laden und Speichern von Programmen.

Als Datenträger können alle Medien mit der Formatierung FAT32 verwendet werden. Das Grundprogramm selbst stellt keine Funktionen zur Formatierung der Datenträger bereit.

### CAS (optional)

Eine CAS Baugruppe oder eine CAS-NEO kann verwendet werden. Das Grundprogramm unterstützt das Laden und Speichern von Programmen sowie das Verifizieren einer Aufnahme.

### PROMER (optional)

Über das Menü des Grundprogramms kann der Inhalt von EPROM's gelesen und leere EPROM's programmiert werden.

### SER (optional)

Das Grundprogramm stellt Routinen zur Nutzung einer seriellen Schnittstelle bereit, die in eigenen Programmen benutzt werden können.

### UHR (optional)

Falls eine UHR3 Baugruppe verfügbar ist können Datum, Zeit und Wochentag gestellt werden. Das aktuelle Datum, der Wochentag und die Uhrzeit werden in der Kopfzeile dargestellt und aktualisiert solange das Grundprogramm auf eine Menüauswahl des Benutzers wartet. Innerhalb von Eingabefeldern findet keine Aktualisierung der Uhrzeit statt.

## Kompatibilität

Innerhalb des Grundprogramms haben sich praktisch alle Startadressen der Routinen und Unterprogramme verschoben. Bereits kompilierte eigene Programme, die Routinen aus dem Grundprogramm benutzen, müssen angepasst bzw. neu übersetzt werden.

### GOSI

Die Grafisch Orientierte Sprache I kann nicht mehr mit dem neuen Grundprogramm eingesetzt werden, da das zweite ROM des Grundprogramms den Steckplatz des GOSI ROM einnimmt.

### BASIC

Das BASIC ROM kann uneingeschränkt in Verbindung mit dem Grundprogramm 2018 verwendet werden. Das Grundprogramm erkennt automatisch, wenn das BASIC ROM vorhanden ist und stellt einen Menüpunkt zum Aufruf bereit. Wenn BASIC nicht erkannt wurde, wird ein Menüpunkt zum Aufruf einer anderen Routine ab der Adresse 4000h dargestellt.

### EZASS

Der Assembler kann nicht mehr mit dem Grundprogramm genutzt werden, da der Speicherbereich ab 6000h mit RAM belegt sein muss. Stattdessen wird es einen neuen Assembler mit Monitor geben, mit dem Listings gespeichert und mit Kommentaren versehen werden können.

### Anwenderprogramme

Eigene Programme müssen ebenfalls neu übersetzt werden, wenn absolute Adressen aus dem alten Grundprogramm verwenden wurden. Weitere Informationen zur Anpassung finden sich im Kapitel Sprungvektoren. Im Anhang werden die vom Grundprogramm bereitgestellten Routinen und deren Benutzung in eigenen Programmen ausführlich beschreiben.

### Symbolverwaltung

Das Grundprogramm enthält nicht mehr die Symbolverwaltung, die von EZASS genutzt wurde und den Aufruf von Routinen mittels symbolischer Adressen ermöglichte. Der Zugriff auf Routinen des Grundprogramms muss jetzt über die Sprungtabelle erfolgen.

## Optimaler Systemaufbau

Damit alle Funktionen des Grundprogramms genutzt werden können empfiehlt sich der Aufbau eines Systems in folgender Zusammenstellung.

- CPUZ80
- GDP64K oder GDP64HS
- KEY oder KEY3 mit Tastatur
- BANKBOOT
- ROA64 mit 64 kB RAM oder SRAM1024
- IOE-VDIP1 oder IO-USB
- CAS oder CAS-NEO
- PROMER

Auf der Baugruppe BANKBOOT

- Steckplatz 1: EPROM 27C64 mit Grundprogramm 2018 #0000
- Steckplatz 2: EPROM 27C64 Grundprogramm 2018 #2000
- Steckplatz 3: EEPROM vom Typ AT28C64 oder kompatibel
- Steckplatz 4: 8 kB RAM vom Typ HM6264 oder kompatibel

## Dokumentationsstand

Dieses Dokument bezieht sich auf jeweils aktuellste Version des Grundprogramms. Die angegebenen Prüfsummen wurden mit dem Grundprogramm über den Adressbereich von 0000h bis 3FFFh ermittelt.

GP2018v1 vom 29. Januar 2018

Prüfsumme: AA9B

Erste veröffentlichte Version des Grundprogramms

GP2018v2 vom 14. Februar 2018

Prüfsumme: 0377

Neue Funktion ReAssembler

Unterstützung des Grafikprozessors EF9367

Allgemeine Fehlerbehebungen

## Menüsystem

Das Grundprogramm verfügt über zwei Menüs, deren Menüpunkte jeweils durch einen einfachen Tastendruck ausgelöst werden können. Mit der Leertaste kann zwischen den beiden Menüs des Grundprogramms gewechselt werden.

### Grundprogramm 2018

Das folgende Bild zeigt das Menü des Grundprogramms, in dem die am häufigsten genutzten Funktionen für die Benutzung des Computers zusammengefasst sind.



### Monitor 2018

Das folgende Bild zeigt das Menü der Monitorfunktionen, die hauptsächlich bei der Erstellung eigener Programme und zum Debugging genutzt werden können.



## Seitenumschaltung

Die Grafik-Baugruppen GDP64K und GDP64HS verfügen über einen Bildspeicher von 64 kB, von denen zur Darstellung einer Bildschirmseite jedoch nur 16 kB genutzt werden. Die Baugruppen verfügen über eine Seitenumschaltung, mit welcher der aktive Bereich einen von 4 Bereichen im Bildspeicher belegen kann. Im alten Grundprogramm wurde dies für die Flip-Funktion genutzt.

Im Grundprogramm können nunmehr alle 4 Bildschirmseiten genutzt werden, um mehrere Funktionen quasi gleichzeitig benutzen zu können. In beiden Menüs können die Tasten 1...4 benutzt werden, um zwischen den Bildschirmseiten zu wechseln.

Wenn eine Seite noch unbenutzt ist, wird nach der Aktivierung der Seite ein neues Menü angezeigt. Die gerade aktive Bildschirmseite wird in der Kopfzeile über dem Menü benannt.

Funktionen des Grundprogramms oder des Monitors, die eine Seitenumschaltung unterstützen gestatten ebenfalls eine Umschaltung der Bildschirmseite. Über die Tasten 1...4 kann eine andere Bildschirmseite gewählt werden ohne dass die Bildschirmausgabe der Funktion auf der alten Seite verloren geht.

Später kann wieder auf eine Bildschirmseite zurück gewechselt werden auf der noch die Anzeige der Ergebnisse einer Funktion sichtbar sind. Die so wieder aktivierte Funktion kann ganz normal weiter genutzt werden bis diese durch den Menüpunkt M verlassen wird und das Hauptmenü angezeigt wird.

Welche der Funktionen des Grundprogramms die Seitenumschaltung unterstützen ist jeweils bei der Beschreibung der Funktion angegeben.

## Funktionen des Grundprogramms

Die Funktionen auf der Menüseite des Grundprogramms können durch einen einfachen Tastendruck gestartet werden. Die zu drückende Taste ist unmittelbar vor der Funktion angegeben. Dabei wird zwischen Groß- und Kleinschreibung unterschieden.

### L = laden CAS

---

Mit dieser Funktion können Daten und Programme über die Baugruppe CAS von einer Musikkassette oder über die Baugruppe CAS-NEO von einer SD-Karte geladen werden.

Nach dem Start der Funktion kann sofort das Band gestartet werden. Dabei muss die Kassette zur gewünschten Position gespult sein. Bei der CAS-NEO muss die gewünschte Datei ausgewählt werden.

Die Angabe einer Zieladresse ist nicht notwendig, da die Informationen über die Startadresse und Endadresse beim Speichern mit den Daten gesichert werden.

Der Lesevorgang kann während des Ladens mit einer beliebigen Taste abgebrochen werden. Im Falle von Lesefehlern wird nach dem Abschluss der Funktion eine Fehlermeldung angezeigt.

### S = speichern CAS

---

Mit dieser Funktion können Daten aus dem Speicher des NKC über die Baugruppe CAS auf einer Musikkassette oder mit einer Baugruppe NAS-NEO auf einer SD-Karte gespeichert werden.

Nach dem Start der Funktion wird die Startadresse im RAM (Start), die Endadresse im RAM (Ende) und der gewünschte Dateiname (Name) abgefragt. Nach der Eingabe der Daten muss das Band gestartet werden.

Beim Speichern von Programmen und Daten auf Kassette werden in einem Vorspann die Start- und Endadresse sowie der Dateiname aufgezeichnet. Außerdem werden die vom Anwender definierten Symbole auf das Band geschrieben.

### V = prüfen CAS

---

Mit dieser Funktion kann die Aufzeichnung auf Kassette oder SD-Karte überprüft werden. Nach dem Start der Funktion kann das Band sofort gestartet werden. Dabei muss die Kassette auf den Anfang der zu prüfenden Datei zurückgespult sein bzw. bei der CAS-NEO die aufgezeichnete Datei ausgewählt werden.

Falls die Prüfung fehlschlägt wird eine Fehlermeldung (SUMME) ausgegeben.

### W = Bank Lader CAS

---

Mit dem Bank Lader ist es möglich in eine RAM-Bank (ROA64 mit 64kB RAM / SRAM1M) ein neues Betriebssystem zu laden und dort zu starten. Das neue Betriebssystem muss bei der Adresse 0000h beginnen.

Zunächst wird die Banknummer (0...F) eingegeben. Nach der Eingabe der Banknummer muss direkt das Laufwerk gestartet werden. Das zu ladende Programm wird immer ab der Adresse 0000h geladen und an dieser Adresse gestartet.

Voraussetzung ist es, dass der NKC mit der Baugruppe BankBoot ausgerüstet ist, das GP auf der BankBoot läuft und mindestens in Bank0 64kB RAM vorhanden sind. Diese Programmroutine ist nur auf der BankBoot lauffähig! Nach Auswahl der Bank wird zunächst vom GP die neue Bank ab Adresse 8000h „hinter“ der BankBoot eingeblendet und auf dieser ein Ladeprogramm in den RAM oberhalb F800h geladen und dort gestartet. Dieses Ladeprogramm schaltet die BankBoot ab, so dass nur noch die neue Bank im Zugriff steht.

Anschließend wird die gewählte Datei ab Adresse 0000h geladen (egal von welcher Adresse die Datei über CAS gespeichert wurde). Nach dem Ladevorgang wird die Adresse 0000h angesprungen und das neue System damit im RAM gestartet.

Ein Rücksprung zum GP auf der BankBoot ist nur mit RESET möglich.

## G = starten PRG

---

Mit dieser Funktion können eigene Programme gestartet werden, die sich an einer beliebigen Adresse im Speicher befinden. Dazu muss zu Beginn die Startadresse eingegeben und bestätigt werden.

Eigene Programme müssen mit einem RET-Befehl (C9) abgeschlossen werden. Nach dem Ende des Programms kann die Bildschirmseite gewechselt oder in das Monitor-Menü zurückgesprungen werden.

Soll in einem Programm die Bildschirmseite gewechselt werden so bietet es sich an, den Inhalt dieser Seite vorher zu löschen.

## C = BASIC Start

---

Falls nach dem Einschalten des NKC ab der Adresse 4000h das Vorhandensein des BASIC EPROM festgestellt wurde, wird dieser Menüpunkt sichtbar. So kann der BASIC Interpreter schnell aus dem Grundprogramm gestartet werden. Falls kein BASIC EPROM erkannt wurde erscheint stattdessen „Start 4000“ im Menü um ein anderes Programm an dieser Stelle starten zu können.

Nach dem Start des BASIC erscheint nur ein Fragezeichen auf dem Bildschirm. Erst nachdem auf der Tastatur ein großes C oder W getippt wurde, erscheint die Einschaltmeldung des BASIC Interpreters. Dabei steht das C für Kaltstart mit Initialisierung des BASIC-Systems und das W für einen Warmstart unter Beibehaltung des evtl. noch geladenen BASIC-Programms.

BASIC kann durch die Eingabe von „CALL 0“ wieder verlassen werden, das Grundprogramm wird dadurch neu gestartet.

Das BASIC EPROM muss ab der Adresse 4000h verfügbar sein.

## I = lesen I/O

---

Die Funktion dient zum Einlesen und Anzeigen von Eingabewerten von einem I/O Port.

Nach der Eingabe der Portadresse im Bereich 00h bis 0FFh wird der Port gelesen und das gelesene Byte unmittelbar auf dem Bildschirm dargestellt.

P=Port	Eingabe einer neuen Port-Adresse
D=Dauer	Dauerbetrieb, der I/O Port wird dauerhaft abgefragt Das Ergebnis wird ständig aktualisiert
S=Stop	Stopp, beenden des Dauerbetriebs
M=Menü	Zurück zum Grundprogramm-Menü
1-4=Seite	Umschaltung der Bildschirmseite

## O = setzen IO

---

Die Funktion dient zur Ausgabe eines Wertes auf einem I/O Port.

Nach der Eingabe der Portadresse im Bereich von 00h bis 0FFh kann ein Wert für die Ausgabe im Bereich von 00h bis 0FFh eingegeben werden. Nach Eingabe des Wertes wird dieser unmittelbar auf dem Port ausgegeben

P=Port	Eingabe einer neuen Port-Adresse
M=Menü	Zurück zum Grundprogramm-Menü
1-4=Seite	Umschaltung der Bildschirmseite

## I = laden USB

---

Die Funktion dient zum Laden von Programmen oder Daten von einem USB-Datenträger.



Beim ersten Aufruf einer USB-Funktion wird das USB-Modul VDIP1 initialisiert, dieser Vorgang kann einige Sekunden dauern. Während der Initialisierung wird der Text „Init USB ...“ angezeigt

Nach dem Start wird die Zieladresse im RAM (Adr) und der Dateiname (Name) abgefragt. Nach der Eingabe des Dateinamens wird die Datei unmittelbar an die angegebene Adresse geladen. Bei der Eingabe einer leeren Startadresse oder einem leeren Dateinamen wird die Funktion abgebrochen. Der Dateiname darf aus maximal 12 Zeichen (Format 8.3) bestehen.

Bei der Eingabe des Dateinamens wird nicht zwischen Groß- und Kleinschreibung unterschieden. Alle Buchstaben eines Namens werden automatisch in Großbuchstaben umgewandelt.

Es wird immer die komplette Datei geladen, es findet keine Kontrolle statt, ob die Datei in den verfügbaren Speicher passt.

Zum Einsatz der Funktion ist eine IOE- oder IO-USB-Baugruppe VDIP1-USB-Modul notwendig.

## s = speichern USB

---

Mit dieser Funktion können Daten und Programme aus dem Speicher des NKC auf einem USB-Datenträger gespeichert werden.

Beim ersten Aufruf einer USB-Funktion wird das USB-Modul VDIP1 initialisiert, dieser Vorgang kann einige Sekunden dauern. Während der Initialisierung wird der Text „Init USB ...“ angezeigt

Nach dem Start der Funktion werden die Startadresse, die Endadresse und der Dateiname abgefragt. Bei der Eingabe einer leeren Adresse oder einem leeren Dateinamen wird die Funktion abgebrochen.

Der Speichervorgang findet unmittelbar nach der Eingabe des Dateinamens statt.

Bei der Eingabe des Dateinamens wird nicht zwischen Groß- und Kleinschreibung unterschieden. Kleinbuchstaben im Dateinamen werden automatisch in Großbuchstaben gewandelt. Der Dateiname darf aus maximal 12 Zeichen (Format 8.3) bestehen und darf nicht leer sein.

Das Speichern auf einem USB-Stick dauert etwa eine Sekunde pro zu speicherndem Kilobyte. Ein begonnener Vorgang kann nicht abgebrochen werden. Während des Speichervorgangs blinkt die Zugriffs-LED auf dem VDIP1 Modul.

Zum Einsatz der Funktion ist eine IOE- oder IO-USB-Baugruppe VDIP1-USB-Modul notwendig.

## v = USB Inhalt

---

Mit dieser Funktion kann der Inhalt eines USB-Laufwerks angezeigt werden.

Beim ersten Aufruf einer USB-Funktion wird das USB-Modul VDIP1 initialisiert, dieser Vorgang kann einige Sekunden dauern. Während der Initialisierung wird der Text „Init USB ...“ angezeigt

Die Anzeige der auf dem Datenträger vorhandenen Dateien ist dreispaltig, so dass 48 Dateien gleichzeitig angezeigt werden können. Falls der Datenträger mehr Dateien enthält, wird nach einem Druck auf die Eingabetaste jeweils eine weitere Seite des Inhaltsverzeichnisses angezeigt.

L=Lade	Dient zum direkten Laden einer Datei
D=Lösche	Dient zum Löschen einer Datei vom Datenträger
CR=weiter	falls das Inhaltsverzeichnis nicht auf eine Bildschirmseite passt wird die nächste Seite des Verzeichnisses angezeigt.
M=Menü	Zurück zum Grundprogramm-Menü
1-4=Seite	Umschaltung der Bildschirmseite

Zum Einsatz der Funktion ist eine IOE- oder IO-USB-Baugruppe VDIP1-USB-Modul notwendig.

## a = USB auswerfen

---

Mit dieser Funktion wird das USB-Modul wieder in den Grundzustand versetzt. Ein angeschlossener Datenträger wird stromlos und kann entfernt werden. Bei einem erneuten Zugriff wird das USB Modul neu initialisiert.

## w = Bank Lader USB

---

Mit dem Bank Lader ist es möglich in eine RAM-Bank (ROA64 mit 64kB RAM / SRAM1M) ein neues Betriebssystem zu laden und dort zu starten. Das neue Betriebssystem muss bei der Adresse 0000h beginnen.

Zunächst wird die Banknummer (0...F) eingegeben. Anschließend muss der Dateiname eingegeben werden, unter dem das System im USB-Speicher abgelegt ist. Das zu ladende Programm wird immer ab der Adresse 0000h geladen und an dieser Adresse gestartet.

Voraussetzung ist es, dass der NKC mit der Baugruppe BankBoot ausgerüstet ist, das GP auf der BankBoot läuft und mindestens in Bank0 64kB RAM vorhanden sind. Diese Programmroutine ist nur auf der BankBoot lauffähig! Nach Auswahl der Bank wird zunächst vom GP die neue Bank ab Adresse 8000h „hinter“ der BankBoot eingeblendet und auf dieser ein Ladeprogramm in den RAM oberhalb F800h geladen und dort gestartet. Dieses Ladeprogramm schaltet die BankBoot ab, so dass nur noch die neue Bank im Zugriff steht.

Anschließend wird die gewählte Datei ab Adresse 0000h geladen. Nach dem Ladevorgang wird die Adresse 0000h angesprungen und das neue System damit im RAM gestartet.

Ein Rücksprung zum GP auf der BankBoot ist nur mit RESET möglich.

## B = Bank wechseln

---

Nach Eingabe der gewünschten Banknummer 0...F wird auf die Adressen OFFFEh und OFFFFh ein Befehl zur Bankumschaltung geschrieben. Die Programmausführung wird nach der Umschaltung in der gewählten Bank ab Adresse 0000h fortgeführt. Die BankBoot-Baugruppe wird/bleibt dabei deaktiviert.

## E = Bank einblenden

---

Nach Eingabe der gewünschten Banknummer 0...F wird im Hintergrund der BANKBOOT Baugruppe die ausgewählte RAM-Bank im Bereich 8000h bis OFFFFh eingeblendet. Die Funktion kann nur auf der BANKBOOT Baugruppe verwendet werden.

## u = Uhr stellen

---

Sofern der NKC mit der UHR3-Baugruppe ausgestattet ist, kann man mit dieser Funktion das Datum, Uhrzeit und Wochentag verändern. Dabei wird die Uhrzeit und Datum in Echtzeit auf dem Bildschirm dargestellt. Im Falle der Veränderung der Uhrzeit werden die Sekunden dabei auf 00 gesetzt.

Bedienmöglichkeiten innerhalb der Funktion:

D=Datum	Datum einstellen
J=Jahr	Jahr einstellen
Z=Zeit	Uhrzeit einstellen
W=Wochentag	Wochentag einstellen
M=Menü	Zurück zum Grundprogramm-Menü
1-4=Seite	Umschaltung der Bildschirmseite

Wie die Eingaben formatiert sein müssen wird jeweils auf dem Bildschirm angezeigt.

## r = lesen EPROM

---

Mit dieser Funktion kann ein Speicherbereich von einem in der Baugruppe PROMER eingesteckten EPROM in den RAM-Speicher übertragen werden.

Nach dem Start werden die Startadresse im EPROM (von), die Endadresse im EPROM (bis) und die Zieladresse im RAM (nach) abgefragt. Die Übertragung findet unmittelbar nach der Eingabe der Zieladresse statt.

### p = prog. EPROM \_\_\_\_\_

Mit dieser Funktion kann ein Speicherbereich aus dem Hauptspeicher in ein leeres EPROM auf der Baugruppe PROMER übertragen werden.

Nach dem Start werden die Startadresse im RAM (von), die Endadresse im RAM (bis) und die Zieladresse im EPROM (nach) abgefragt. Durch einen Druck auf die Taste B wird die Programmierung gestartet.

Standardmäßig wird ein neuer schnellerer Algorithmus zur Programmierung verwendet. Das Beschreiben eines kompletten EPROM's 2764 dauert etwa 20 Sekunden. Während des Vorgangs findet keine Bildschirmausgabe statt.

### P = EEPROM schreiben \_\_\_\_\_

Die Funktion dient zum direkten Programmieren (Beschreiben) von EEPROMs vom Typ 28C64 im Speicherbereich.

Nach dem Start werden die Startadresse im RAM, die Endadresse im RAM und die Zieladresse im EEPROM abgefragt. Gleich zu Beginn wird über den Quellbereich eine CRC-Prüfsumme berechnet und angezeigt. Darunter erscheint die Adresse der soeben beschriebenen Adresse im EEPROM und abschließend zur Kontrolle die Berechnung der CRC-Prüfsumme über den beschriebenen Bereich im EEPROM.

Es werden nur solche Bytes im EEPROM beschrieben, die tatsächlich geändert werden müssen. Daher kann es sein, dass die Funktion unterschiedliche Laufzeiten aufweist.

### 1-4 = Bildschirm \_\_\_\_\_

Aktiviert eine der 4 Bildschirmseiten der Baugruppe GDP64K oder GDP64HS.

Auf der neuen Seite wird wieder das Grundprogramm – Menü angezeigt, falls auf der Seite vorher keine andere Funktion aktiviert war. Falls dort eine aktive Funktion mittels der Seitenumschaltung verlassen wurde wird diese Funktion wieder aktiviert und kann weiter bedient werden.

### SPACE = Monitor \_\_\_\_\_

Wechselt vom aktiven Grundprogramm – Menü zum Monitor – Menü.

## Funktionen des Monitors

Die Funktionen auf der Menüseite des Monitors können durch einen einfachen Tastendruck gestartet werden. Die zu drückende Taste ist unmittelbar vor der Funktion angegeben. Dabei wird zwischen Groß- und Kleinschreibung unterschieden.

### m = ändern MEM

---

Die Funktion dient zur Eingabe von Programmcode in Maschinensprache. Am Anfang wird die Startadresse des zu ändernden Speicherbereiches abgefragt, woraufhin die folgenden Befehle angezeigt werden.

Im Eingabefeld für die aktuelle Adresse können mehrere durch ein Leerzeichen getrennte Werte oder Text eingegeben werden, die aufeinanderfolgend ab der aktuellen Adresse abgelegt werden.

Beispiele für Gültige Eingaben

```
CD 00 01      ; Speichert die Bytes CD, 00 und 01
10.W          ; Speichert die Bytes 10 und 00
ABCD          ; Speichert die Bytes CD und AB
ABCD.W        ; Speichert die Bytes CD und AB
ABCD.B        ; Speichert das Byte CD
"ABCD"        ; Speichert die Bytes 41, 42, 43 und 44
'ABCD'        ; Speichert die Bytes 41, 42, 43 und 44
```

### d = ansehen MEM

---

Mit dieser Funktion kann der Inhalt des Speichers auf dem Bildschirm ausgegeben werden. Es werden jeweils 256 Bytes in hexadezimaler Schreibweise und als ASCII Zeichen ausgegeben.

Nach dem Start der Funktion muss eine Adresse eingegeben werden, die als Startadresse für die Anzeige des Speicherbereiches dient. Die Adresse wird bei Bedarf automatisch so angepasst, dass am Anfang einer Zeile immer eine glatte hexadezimal-Adresse angezeigt wird.

CR=vor	Eine Bildschirmseite (256 Bytes) vorwärts blättern
BS=zur	Eine Bildschirmseite (256 Bytes) zurück blättern
R=Adr	Eingabe einer neuen Startadresse
M=Menü	Zurück zum Monitor Menü
1-4=Seite	Umschaltung der Bildschirmseite

### a = Abbild MEM

---

Diese Funktion stellt einen Speicherbereich in Blöcken von jeweils einem kB als ASCII-Zeichen auf dem Bildschirm dar. Damit ist es schnell möglich „auf Sicht“ zu prüfen, ob sich ein gleichmäßiges Muster im Speicher (in einzelnen Zellen) verändert hat.

Ebenso ist es möglich ASCII-Textfiles zu lesen. ASCII-Steuerzeichen werden allerdings dabei ebenso ignoriert wie solche mit gesetztem Bit 7. Daher wird zusätzlich die CRC-Prüfsumme über den aktuell angezeigten Speicherbereich berechnet und angezeigt.

CR=vor	Eine Bildschirmseite (256 Bytes) vorwärts blättern
BS=zur	Eine Bildschirmseite (256 Bytes) zurück blättern
R=Adr	Eingabe einer neuen Startadresse
M=Menü	Zurück zum Monitor Menü
1-4=Seite	Umschaltung der Bildschirmseite

### f = füllen MEM

---

Mit dieser Routine ist es möglich einen Speicherbereich komplett mit einem gewählten Byte zu beschreiben. Dazu werden die Startadresse und Endadresse des Speicherbereichs sowie der Wert mit dem der Speicherbereich gefüllt werden soll abgefragt.

Nach der Ausführung wird sofort zum Monitor-Menü zurückgekehrt.

Hinweis: Die Funktion enthält keine Sicherheitsmechanismen. Das Überschreiben des Variablenbereichs des Grundprogramms oder des Stapelspeichers (8000h...8800h) führt zu einem Absturz des Grundprogramms.

### t = transfer MEM

---

Mit dieser Programmroutine kann ein beliebiger Speicherbereich im RAM verschoben werden. Nach dem Aufruf werden drei Adressen abgefragt.

von ADR	Startadresse des Quellbereichs
bis ADR	Endadresse des Quellbereichs
nach ADR	Startadresse des Zielbereichs

Dabei ist es egal ob der Bereich nach vorn oder nach hinten verschoben wird, d.h. ob von ADR größer oder kleiner als nach ADR ist. Es ist auch möglich einen Bereich z.B. um nur 1 Byte zu verschieben.

von ADR	8000
bis ADR	8FFF
nach ADR	8001

### v = vergleiche MEM

---

Die Funktion dient zum Vergleichen von zwei Speicherbereichen. Nach dem Aufruf werden drei Adressen abgefragt.

von ADR	Startadresse des 1. Bereichs
bis ADR	Endadresse des 1. Bereichs
mit ADR	Startadresse des 2. Bereichs

Falls Unterschiede gefunden werden, werden jeweils Adresse und Inhalt des 1. Bereichs und des zweiten Bereichs angezeigt. Falls mehr Unterschiede gefunden werden, als auf dem Bildschirm passen kann mit der Eingabetaste die folgenden Ergebnisse abgerufen werden.

N=Neu	Neustart der Funktion
CR=weiter	Weitere Ergebnisse falls vorhanden
M=Menü	Zurück zum Monitor Menü
1-4=Seite	Umschaltung der Bildschirmseite

### s = Mustersuche MEM

---

Mit dieser Funktion ist es möglich eine Bytefolge anzugeben nach der in einem Adressbereich gesucht werden soll. Dabei werden die Fundstellen mit Adresse und Folgebyte aufgelistet.

Adresse	Startadresse der Mustersuche
Byte's	Durch Leerzeichen getrennte Werte

Im Eingabefeld müssen die zu suchenden Bytes mit einem Leerzeichen getrennt werden.

Die Suche kann unter Umständen, z.B. wenn ein großer Speicherbereich nur mit dem ersten Such-Byte gefüllt sind, je nach Taktrate bis zu 3-5 Minuten dauern. In den meisten Fällen dauert die Suche selbst über den gesamten Speicherbereich nur wenige Sekunden.

N=Neu	Neustart der Funktion
M=Menü	Zurück zum Monitor Menü
1-4=Seite	Umschaltung der Bildschirmseite

Passt die Anzahl der Treffer nicht auf eine Bildschirmseite, so kann die Funktion mit N=Neu ab einer späteren Adresse neu gestartet werden.

Mit dieser Funktion kann ein im Speicher vorliegendes Programm im Einzelschritt-Modus ausgeführt werden. Zu Beginn muss die Startadresse eingegeben werden.

Adresse	Startadresse des auszuführenden Programms
---------	---

Nach jeder Ausführung eines Maschinensprachebefehls werden im unteren Bereich des Bildschirms diverse hilfreiche Informationen dargestellt.

Drei getrennte Speicherbereiche von jeweils 16 Byte Umfang (A, B, C)
Inhalt des Stapelspeichers (Stack)
Vier Bytes vor und 18 Bytes nach dem aktuellen Befehl (File)
Programmzähler (PC)
Aktueller Befehl in hexadezimaler Darstellung und Assemblerbefehl
Alle Registerinhalte des Z80 Prozessors

Bedienmöglichkeiten innerhalb der Funktion:

M=Menü	Rückkehr zum Monitor Menü
1-4=Seite	Seitenumschaltung
R=Adr.	Eingabe einer neuen Startadresse
N=nMal	Ausführen einer bestimmten Anzahl von Befehlen
B=Bis	Ausführen des Programms bis zu einer bestimmten Adresse
I=Input	Eingabe von Registerwerten und Adressen
CR=Step	Ausführen des aktuellen Befehls
+=PC+1	Programmzähler erhöhen (keine Ausführung)
-=PC-1	Programmzähler verringern (keine Ausführung)
#=Speich	direkte Änderung von Speicherinhalten

Nicht sichtbarer Menüpunkt

S=Skip	Verdecktes Ausführen eines Unterprogramms
--------	---

Es sollte beachtet werden, dass das System unter Umständen nicht mehr reagiert, wenn das Programm bei den Funktionen S=Skip und B=Bis in einer Endlosschleife verweilt oder die gewählte Adresse nicht erreicht wird.

Nach der Auswahl von I=Input können in einem Eingabefeld Adressen und Registerinhalte manipuliert werden. Die folgende Tabelle zeigt die Möglichkeiten der Beeinflussung. Registerkürzel und Wert müssen durch ein Leerzeichen getrennt sein.

A Adresse	Festlegen der Adresse für Speicherbereich A
B Adresse	Festlegen der Adresse für Speicherbereich B
C Adresse	Festlegen der Adresse für Speicherbereich C
AF Wert	Inhalt von ACCU und FLAGS setzen
BC Wert	Inhalt des Doppelregisters BC setzen
DE Wert	Inhalt des Doppelregisters DE setzen
HL Wert	Inhalt des Doppelregisters HL setzen
IX Wert	Inhalt des Indexregisters IX setzen
IY Wert	Inhalt des Indexregisters IY setzen
I Wert	Inhalt des Interrupt-Registers I setzen
R Wert	Inhalt des Refresh-Registers R setzen
AF' Wert	Inhalt des Schattenregisters AF setzen
BC' Wert	Inhalt des Schattenregisters BC setzen
DE' Wert	Inhalt des Schattenregisters DE setzen
HL' Wert	Inhalt des Schattenregisters HL setzen
SP Adresse	Inhalt des Stapelzeigers SP setzen
PC Adresse	Inhalt des Programmzählers PC setzen

## g = starten PRG

---

Mit dieser Funktion können eigene Programme gestartet werden, die sich an einer beliebigen Adresse im Speicher befinden. Dazu muss zu Beginn die Startadresse eingegeben und bestätigt werden.

Eigene Programme müssen mit einem RET-Befehl (C9) abgeschlossen werden.

Nach der Rückkehr aus dem aufgerufenen Programm wird das Funktionsmenü angezeigt.

M=Menü	Rückkehr zum Monitor Menü
1-4=Seite	Seitenumschaltung

Soll in einem Programm die Bildschirmseite gewechselt werden so bietet es sich an, den Inhalt dieser Seite vorher zu löschen, da auf anderen Bildschirmseiten

## c = Prüfsumme

---

Die Funktion dient zum Berechnen einer Prüfsumme CRC-CCITT über einen Speicherbereich, der durch die Startadresse und Endadresse angegeben wird. Nach Abfrage der Adressen wird über diesen Bereich eine Prüfsumme gebildet und auf dem Bildschirm ausgegeben.

Die Prüfsummen Routine wird innerhalb der Z80-SIO verwendet und war in den 80'er Jahren im Homecomputer-Bereich weit verbreitet. Kennzeichnend sind markante Prüfsummen für leere EPROMS (alle Speicherstellen 0FFh) 2708 = 77EB oder 2732 = 0FE1.

Bedienmöglichkeiten innerhalb der Funktion

N=Neu	Neustart der Funktion mit neuen Adresse
M=Menü	Rückkehr zum Monitor Menü
1-4=Seite	Seitenumschaltung

## b = Berechnen

---

Die Funktion dient zum Berechnen von Summe und Differenz zweier 16-Bit Werte, die zu Beginn eingegeben werden können. Es müssen zwei Werte eingegeben werden, ein leeres Eingabefeld führt zum Abbruch der Funktion.

Die Eingabe der Werte darf mit führendem # auch dezimal erfolgen.

In den Eingabefeldern sind Ausdrücke erlaubt.

100+#50	entspricht: 0132
1000-100	entspricht: 0F00
1+2+3-4	entspricht: 0003

Nach der Berechnung werden die folgenden Daten ausgegeben

Summe A + B	in hexadezimaler Schreibweise
Differenz A – B	in hexadezimaler Schreibweise
Differenz B – A	in hexadezimaler Schreibweise
Dez A	Dezimalwert A und Zweierkomplement von A
Dez B	Dezimalwert B und Zweierkomplement von B

Bedienmöglichkeiten innerhalb der Funktion:

N=Neu	Neustart der Funktion mit neuen Werten
M=Menü	Rückkehr zum Monitor Menü
1-4=Seite	Seitenumschaltung

## R = RAM-Test

---

Die Firma MOSTEK entwickelte in den 80'ern ein Testprogramm für dynamische RAM Bausteine. Dabei wird der zu prüfende Bereich über eine „Formel“ mit einem Bitmuster, welches für jede Zelle wechselt, beschrieben. Anschließend wird geprüft, ob der Inhalt fehlerfrei erhalten geblieben ist. Danach wird das Muster gewechselt und der Vorgang beginnt von vorne.

Nach 256 Durchläufen ist der Test beendet. Wurde ein Fehler gefunden bricht das Programm den Test ab und stellt den Fehler dar. Zu jeder Zeit kann der Ablauf durch einen beliebigen Tastendruck abgebrochen werden.

## \* = löschen System-RAM

---

Der RAM-Bereich 6000h...6FFFh wird mit 00h überschrieben und danach ein Software-RESET ausgeführt. Dabei werden alle gespeicherten Symbole, Variablen und der Stapelspeicher gelöscht. Die zuletzt angewählte Bank wird im Akku über den RESET übergeben und wird wiederhergestellt.

Mit dieser Funktion werden die verwendeten System-RAM Zellen deutlicher erkennbar.

## # = löschen oberer RAM

---

Der RAM-Bereich 8000h..FFFFh wird analog eines gelöschten EPROMS mit 0FFh überschrieben.

Mit dieser Funktion werden nicht von Programmen verwendete Bereiche deutlicher erkennbar.

## @ = ReAssembler

---

Der ReAssembler dient zum Rückübersetzen eines Programmcodes in Assembler-Befehle. Dabei wird aus dem Maschinensprache-Code eine Textdatei erstellt, und auf dem Bildschirm angezeigt. Im Gegensatz zu einem Disassembler werden vom ReAssembler keine Labels erzeugt, hierfür ist manuelle Nacharbeitung der Textdatei notwendig.

Nach dem Start des ReAssemblers muss zunächst die Start- und Endadresse des zu übersetzenden Bereiches und die Startadresse für die Ablage der Textdatei eingegeben werden. Anschließend wird abgefragt, ob die Textdatei als Assemblerdatei oder als List-Datei erzeugt werden soll.

### Assemblerdatei (Beispiel)

```
CALL 4056          ; 4000
LD HL, (60CF)       ; 4003
LD A, (HL)          ; 4006
OR A                ; 4007
RET                 ; 4008
```

Im Listing der Assemblerdatei beginnt jede Zeile mit 4 Leerzeichen, gefolgt vom übersetzten Befehl. Am Ende jeder Zeile wird ein Kommentar mit der Startadresse des Befehls ausgegeben.

### Listdatei (Beispiel)

```
4000: CD 56 40      CALL 4056
4003: 2A CF 60      LD HL, (60CF)
4006: 7E           LD A, (HL)
4007: B7           OR A
4008: C9           RET
```

In der Listdatei werden zunächst die Adresse des Befehls und die Datenbytes des Befehls ausgegeben, danach der übersetzte Befehl. Hier wird nach jedem CALL und RET Befehl eine Leerzeile eingefügt.

Der ReAssembler übersetzt auch alle nicht dokumentierten Befehle des Z80-Mikroprozessors.



Nachdem der Übersetzungsvorgang beendet wurde, erscheint das Funktionsmenü.

CR=vor	Nächste Seite des Listings anzeigen
BS=rück	Vorherige Seite des Listings anzeigen
M=Menü	Zurück zum Monitor-Menü
1-4=Seite	Seitenumschaltung

Im oberen Bereich des Bildschirms wird nach dem Übersetzungsvorgang der Umfang der erzeugten Textdatei angezeigt. Der Bereich schließt das letzte Zeichen des generierten Listings ein, jedoch nicht die Ende-Kennung 00h. Um die Datei für den NKC abzuspeichern sollte die Endadresse um 1 erhöht werden, für die Weiterverarbeitung auf einem PC muss die Ende-Kennung nicht gespeichert werden.

Die Anzeigeroutine des ReAssemblers kann auch genutzt werden, um eine von USB oder CAS erneut eingelesene Textdatei zur Anzeige zu bringen. Dazu muss die Startadresse des geladenen Textes manuell in die Speicherstelle 6097h (6097h = NwB, 6098h = HwB) eingetragen werden.

Danach kann die Textdatei angezeigt werden, indem die Funktion PRG starten ausgeführt wird. Als Startadresse muss 2102h verwendet werden.

Diese Hilfe ist ausschließlich bei der aktuellen Version des Grundprogramms möglich. In einer späteren Version soll das Grundprogramm um einen vollwertigen Editor erweitert werden.

## 1-4 = Bildschirm

---

Aktiviert eine der 4 Bildschirmseiten der Baugruppe GDP64K oder GDP64HS.

Auf der neuen Seite wird wieder das Monitor – Menü angezeigt, falls auf der Seite vorher keine andere Funktion aktiviert war. Falls dort eine aktive Funktion mittels der Seitenumschaltung verlassen wurde wird diese Funktion wieder aktiviert und kann weiter bedient werden.

## SPACE = Grundprogramm

---

Wechselt vom aktiven Monitor – Menü zum Grundprogramm – Menü.

## Programmieren mit dem Grundprogramm

Im originalen Grundprogramm war der Aufruf einiger Funktionen über symbolische Bezeichner vorgesehen. Über diese Symbole konnten die Funktionen in eigenen Programmen genutzt werden.

Leider blockierten die Einsprünge für diese Funktionen die ersten Adressen im Speicher, so dass die RST-Befehle des Z80 nicht genutzt werden konnten.

### Aufruf von Systemfunktionen

Ab dieser Version des Grundprogramms werden die symbolischen Bezeichner nicht länger unterstützt. Stattdessen verfügt das Grundprogramm ab der Adresse 100h im ROM über eine umfangreiche Tabelle mit Einsprünge zu fast allen wichtigen Routinen des Grundprogramms.

In dieser Sprungtabelle ist für jede Routine ein Sprungbefehl zur effektiven Adresse im ROM vorhanden. Alle Routinen werden mit einem Return-Befehl beendet, so dass die Adressen in der Sprungtabelle wie ein normales Unterprogramm benutzt werden können.

Die Position der Sprungtabelle wird auch in zukünftigen Versionen des Grundprogramms an der gleichen Stelle liegen. Bei einem Update des Grundprogramms müssen eigene Programme also nicht mehr neu übersetzt werden.

### Vorbemerkungen

Falls ein Anwenderprogramm die aktuelle Bildschirmseite wechseln soll, muss dafür Sorge getragen werden, dass eine eventuell von Funktionen des Grundprogramms oder Monitors belegte Seite des Bildschirms freigegeben und der Bildschirminhalt gelöscht wird. Zum Beenden / Abbrechen einer eventuell laufenden Funktion dient der Systemaufruf KILL.

### Liste der Systemfunktionen

Die folgende Liste enthält die Adressen aller Routinen, die für eigene Programme zur Verfügung stehen. Die Liste mit den Konstanten kann in vielen Assemblern direkt verwendet werden, um die Routinen des Grundprogramms mit einem Namen ansprechen zu können.

In den späteren Beispielprogrammen werden die klarschriftlichen Bezeichner der Routinen benutzt.

CSTS	.EQU 100h	; Status von KEY abfragen
CI	.EQU 103h	; Zeichen von KEY einlesen
KI	.EQU 106h	; Zeichen von KEY einlesen
WAIT	.EQU 109h	; Warten bis GDP bereit
WAITSYNC	.EQU 10Ch	; Warten auf Strahlrücklauf GDP
CMD	.EQU 10Fh	; Kommando an GDP senden
FAST	.EQU 112h	; Schnelle Ausgabe ohne Anzeige
SLOW	.EQU 115h	; Normale Ausgabe mit Anzeige
SETPEN	.EQU 118h	; Schreibmodus GDP aktivieren
ERAPEN	.EQU 11Bh	; Löschmodus GDP aktivieren
SETVIEWPAGE	.EQU 11Eh	; Anzeigeseite GDP setzen
SETWRTPAGE	.EQU 121h	; Schreibseite GDP setzen
AKTPAGE	.EQU 124h	; Seite in GDP aktivieren
SETAKTPAGE	.EQU 127h	; GDP Seite setzen und aktivieren
CLRALL	.EQU 12Ah	; Alle Seiten löschen
CLRAKT	.EQU 12Dh	; Aktuelle Seite löschen
CLRINVIS	.EQU 130h	; Unsichtbare Seite löschen
KILL	.EQU 133h	; GP Funktion zurücksetzen
GETAUSBUF	.EQU 136h	; Textpuffer (IX) initialisieren
PRTHL	.EQU 139h	; HL HEX nach Textpuffer, 4 Bytes
PRTHLD	.EQU 13Ch	; HL DEZ nach Textpuffer, 5 Bytes
PRTAC	.EQU 13Fh	; ACCU HEX nach Textpuffer, 2 Bytes
PRTACD	.EQU 142h	; ACCU DEZ nach Textpuffer, 3 Bytes
PRTBIN	.EQU 145h	; ACCU BIN nach Textpuffer

PRINT	.EQU 148h	; Text ab (HL) nach Textpuffer
ZEICH	.EQU 14Bh	; Zeichen in ACCU nach Textpuffer
BLANK	.EQU 14Eh	; Leerzeichen nach Textpuffer
MINUS	.EQU 151h	; Minuszeichen nach Textpuffer
PLUS	.EQU 154h	; Pluszeichen nach Textpuffer
DOT	.EQU 157h	; Punkt nach Textpuffer
COLON	.EQU 15Ah	; Doppelpunkt nach Textpuffer
NEWLINE	.EQU 15Dh	; CR LF nach Textpuffer
PRINTBUF	.EQU 160h	; Textpuffer ausgeben
TEXTAUS	.EQU 163h	; Text ab (IX) direkt ausgeben
TEXTMULTI	.EQU 166h	; Mehrere Texte ausgeben
PRINTIN	.EQU 169h	; Inline-Textausgabe
TEXTEIN	.EQU 16Ch	; Eingabefeld über Versorgungsblock
TEXTXY	.EQU 16Fh	; Texteingabefeld erzeugen
GETHL	.EQU 172h	; Eingabe als Zahl interpretieren
GETTEXT	.EQU 175h	; Zeiger auf Eingabe zurückgeben
EXPR	.EQU 178h	; Auswertung eines Ausdrucks
GETPARA	.EQU 17Bh	; Kombinierte Ausgabe und Eingabe
MOVETO	.EQU 17Eh	; Grafikcursor setzen
DRAWTO	.EQU 181h	; Linie zeichnen
TMOVE	.EQU 184h	; Turtle Position setzen
TSCHREITE	.EQU 187h	; Turtle vorwärts bewegen
TSCHR16TEL	.EQU 18Ah	; Turtle langsam bewegen
TDREHE	.EQU 18Dh	; Turtle drehen
THOCH	.EQU 190h	; Turtle anheben
TRUNTER	.EQU 193h	; Turtle absenken
TURTLE	.EQU 196h	; Turtle darstellen
SPRITE	.EQU 199h	; Figur ausgeben
UPPER	.EQU 19Ch	; Zeichen in Großbuchstaben
WAITMS	.EQU 19Fh	; Wartezeit in Millisekunden
WAIT100MS	.EQU 1A2h	; 100 Millisekunden warten
ADJ360	.EQU 1A5h	; Winkel anpassen
SIN	.EQU 1A8h	; Sinus berechnen
COS	.EQU 1ABh	; Cosinus berechnen
CPLHL	.EQU 1AEh	; Zweierkomplement von HL
HEXDEZ	.EQU 1B1h	; Umwandlung HL(hex) CDE(dez)
DEZHEX	.EQU 1B4h	; Umwandlung CDE(dez) HL(hex)
LENGTH	.EQU 1B7h	; Länge eines Befehls berechnen
SIST	.EQU 1BAh	; Status serielle Eingabe
SI	.EQU 1BDh	; Zeichen von SER lesen
SOST	.EQU 1B0h	; Status serielle Ausgabe
SO	.EQU 1C3h	; Zeichen auf SER schreiben
SETBAUD	.EQU 1C6h	; Baudrate setzen
GETDATE	.EQU 1C9h	; Datum aus UHR3 lesen
GETTIME	.EQU 1CCh	; Zeit aus UHR3 lesen
USB_START	.EQU 1CFh	; USB Initialisieren
USB_ISDISK	.EQU 1D2h	; Laufwerk prüfen
USB_DIR	.EQU 1D5h	; Test ob Datei vorhanden ist
USB_LOAD	.EQU 1D8h	; Datei komplett laden
USB_SAVE	.EQU 1DBh	; Datei komplett schreiben
USB_OPW	.EQU 1DEh	; Datei zum Schreiben öffnen
USB_WR	.EQU 1E1h	; Bytes in Datei schreiben
USB_OPR	.EQU 1E4h	; Datei zum Lesen öffnen
USB_RD	.EQU 1E7h	; Bytes aus Datei lesen
USB_CLF	.EQU 1EAh	; Datei schließen
USB_SEK	.EQU 1EDh	; Dateipointer setzen
USB_DLF	.EQU 1F0h	; Datei löschen
USB_REN	.EQU 1F3h	; Datei umbenennen
VNC_READ	.EQU 1F6h	; Byte aus Datei lesen
VNC_WRITE	.EQU 1F9h	; Byte in Datei schreiben
USB_REM	.EQU 1FCh	; Datenträger auswerfen

## Beschreibung der Systemfunktionen

Es folgt eine ausführliche Beschreibung der einzelnen Funktionen, die über die Sprungtabelle angesprochen werden können.

Bei jeder Funktion werden die notwendigen Parameter und die Rückgabewerte angegeben, außerdem die Register die während des Ablaufs der Funktion zusätzlich zu den Rückgabewerten verändert werden.

---

CSTS	ABFRAGE DES TASTATURSTATUS	100H
------	----------------------------	------

---

Dient zum Abfragen des Status der Tastatur über die Baugruppe KEY.

Parameter	keine
Rückgabe	ACCU    \$FF, wenn Zeichen zum Einlesen bereitsteht ACCU    \$00, wenn kein Zeichen bereitsteht. ZERO    gelöscht, wenn ein Zeichen bereitsteht ZERO    gesetzt, wenn kein Zeichen bereitsteht
Register	keine

WAITKEY:

```
CALL CSTS           ; Aufruf CSTS
JR Z,WAITKEY        ; Warten, wenn keine Taste gedrückt
RET
```

---

CI	EINLESEN EINES ZEICHENS VON DER TASTATUR	103H
----	--	------

---

Dient zum Einlesen eines Zeichens von der Tastatur über die Baugruppe KEY oder kompatible. Die Funktion wartet solange, bis tatsächlich ein Zeichen eingegeben wurde.

Parameter	keine
Rückgabe	ACCU    ASCII Code des eingegebenen Zeichens (\$00 - \$7F)
Register	keine

KEYLOOP:

```
CALL CI             ; Zeichen einlesen
CP 0DH              ; Taste CR gedrückt?
JR NZ,KEYLOOP       ; wiederholen bis CR gedrückt
RET
```

---

KI	EINLESEN EINES ZEICHENS VON DER TASTATUR	106H
----	--	------

---

Dient zum Einlesen eines Zeichens von der Tastatur über die Baugruppe KEY oder kompatible Baugruppen. Kleinbuchstaben werden in Großbuchstaben gewandelt. Die deutschen Umlaute werden korrekt behandelt. Die Funktion wartet solange, bis tatsächlich ein Zeichen eingegeben wurde.

Parameter	keine
Rückgabe	ACCU    ASCII Code des eingegebenen Zeichens (\$00 - \$7F)
Register	keine

KEYLOOP:

```
CALL KI             ; Zeichen einlesen
CP 'M'              ; Taste M oder m gedrückt?
JR NZ,KEYLOOP       ; wiederholen bis M oder m gedrückt
RET
```

An die Baugruppe GDP64K oder GDP64HS dürfen nur dann neue Kommandos übergeben werden, wenn das vorangegangene Kommando abgearbeitet ist. WAIT wartet auf das Fertigstellen eines Kommandos.

Die Funktion muss nur dann verwendet werden, wenn eigene Programme die GDP direkt ansprechen. Die im Grundprogramm eingebauten Funktionen warten automatisch auf das Fertigstellen der vorangegangenen Funktion.

Parameter      keine  
Rückgabe       keine  
Register       keine

```
START:
        CALL WAIT           ; Warten auf Bereitschaft
        LD A,41H            ; Buchstabe A
        OUT (70H),A         ; Kommando an GDP senden
        CALL CI             ; Warten auf Tastendruck
        RET
```

Die Funktion wartet solange bis der gesamte sichtbare Bereich eines Bildes an den Monitor übertragen wurde. Anschließend können Bildschirmausgaben erfolgen ohne dass es zu unerwünschten Darstellungsfehlern kommt.

Parameter      keine  
Rückgabe       keine  
Register       ACCU

```
START:
        LD A,41H            ; Buchstabe A
LOOP:
        CALL WAITSYNC       ; Warten auf Bildschirmrücklauf
        CALL CMD            ; Buchstabe ausgeben
        INC A               ; nächster Buchstabe
        CP 5BH              ; Vergleich auf Buchstabe Z
        JR NZ,LOOP          ; bis alles ausgegeben ist
        CALL CI             ; Warten auf Tastendruck
        RET
```

Überträgt ein Kommando, ein Zeichen oder einen Kurzvektor an den Grafikprozessor EF9366 auf der Baugruppe GDP64K oder GDP64HS. Die Funktion wartet, bis der Grafikprozessor die vorangegangene Ausgabe beendet hat.

Parameter      ACCU      Auszugebendes Kommando, Zeichen oder Kurzvektor  
Rückgabe       keine  
Register       keine

```
START:
        LD A,4              ; Bildschirm löschen
        CALL CMD            ; Kommando ausführen
        CALL CI             ; Warten auf Tastendruck
        RET
```

Während der Ausgabe von Kommandos an den Grafikprozessor EF9366 auf der GDP64K wird die Erzeugung des Bildes für den Monitor ausgesetzt. Dadurch, dass nicht mehr gleichzeitig aus dem Speicher gelesen werden muss, erfolgen Änderungen des Bildinhaltes schneller.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	

---

SLOW	NORMALE BILDSCHIRMAUSGABE	115H
------	---------------------------	------

---

Während der Ausführung von Kommandos an den Grafikprozessor EF9366 auf der GDP64K wird gleichzeitig das Bild für den Monitor erzeugt. Dies ist der Standardmodus der Ausgabe.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	

---

SETPEN	SCHREIBMODUS AKTIVIEREN	118H
--------	-------------------------	------

---

Versetzt den Grafikprozessor in den Schreibmodus. Nachfolgende Ausgaben erzeugen sichtbare Pixel auf dem Bildschirm.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	

START:

```

CALL CLRAKT      ; Bildschirm löschen
CALL SETPEN      ; Schreibmodus aktivieren
LD A,41H         ; Buchstabe A
CALL CMD         ; ausgeben
LD HL,1000       ; 1000 Millisekunden
CALL WAITMS      ; warten
CALL ERAPEN      ; Löschmodus aktivieren
LD A,41H         ; Buchstabe A
CALL CMD         ; Löschend überschreiben
CALL CI          ; Warten auf Tastendruck
RET

```

---

ERAPEN	LÖSCHMODUS AKTIVIEREN	11BH
--------	-----------------------	------

---

Versetzt den Grafikprozessor in den Löschmodus. Bei nachfolgenden Ausgaben werden Pixel in der Hintergrundfarbe ausgegeben.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	

---

SETVIEWPAGE	BESTIMMT DIE ANZUZEIGENDE BILDSCHIRMSEITE	11EH
-------------	---	------

---

Legt fest, welche der vier Seiten des Bildspeichers angezeigt werden soll. Die Ausführung der Funktion hat keine unmittelbare Auswirkung, die anzuzeigende Seite wird nur gespeichert und kann später mit AKTPAGE aktiviert werden.

Parameter	ACCU	Seitennummer (0 ... 3)
Rückgabe	keine	
Register	keine	

---

SETWRTPAGE	BESTIMMT DIE ZU BESCHREIBENDE BILDSCHIRMSEITE	121H
------------	---	------

---

Legt fest, auf welcher der vier Seiten des Bildspeichers nachfolgende Vorgänge ausgegeben werden sollen. Die Ausführung der Funktion hat keine unmittelbare Auswirkung, die anzuzeigende Seite wird nur gespeichert und kann später mit AKTPAGE aktiviert werden.

Parameter	ACCU	Seitennummer (0 ... 3)
Rückgabe	keine	
Register	keine	

---

AKTPAGE	AKTIVIEREN VON ANZEIGESEITE UND SCHREIBSEITE	124H
---------	--	------

---

Aktiviert die durch SETVIEWPAGE und SETWRTPAGE bestimmten Seiten als Anzeigeseite und Schreibseite. Bei unterschiedlichen Angaben erfolgen nachfolgende Grafikbefehle unsichtbar auf der Schreibbreite während die Anzeigeseite sichtbar ist.

Parameter	keine
Rückgabe	keine
Register	ACCU

Da das Grundprogramm möglicherweise schon alle 4 Bildschirmseiten benutzt hat, sollten Programme nach dem aktivieren einer neuen Schreibseite zunächst den Bildschirm löschen.

---

SETAKTPAGE	SETZEN UND AKTIVIEREN DER BILDSCHIRMSEITE	127H
------------	---	------

---

Die Funktion wirkt wie eine Zusammenfassung der Funktionen SETVIEWPAGE, SETWRTPAGE und AKTPAGE. Schreibseite und Anzeigeseite sind immer synchron, folgende Grafikbefehle sind sofort sichtbar.

Parameter	ACCU	zu aktivierende Bildschirmseite (0 ... 3)
Rückgabe	keine	
Register	ACCU	
START:		
LD B,0		; Bildschirmseite 0
LOOP:		
CALL WAITSYNC		; Warten auf Synchronisation
LD A,B		; Bildschirmseite holen
CALL SETAKTPAGE		; Bildschirmseite aktivieren
INC B		; nächste Seite
CALL CSTS		; Taste gedrückt?
JR Z,LOOP		; wiederholen bis Taste gedrückt
RET		

---

CLRALL	LÖSCHT DEN INHALT ALLER BILDSCHIRMSEITEN	12AH
--------	--	------

---

Die Funktion löscht den Inhalt aller Bildschirmseiten.

Parameter	keine
Rückgabe	keine
Register	ACCU

Es sollte beachtet werden, dass das Grundprogramm alle Bildschirmseiten nutzen kann und auf den Seiten Ergebnisse von Funktionen stehen könnten.

---

CLRAKT	LÖSCHT DEN INHALT DER AKTUELLEN SCHREIBSEITE	12DH
--------	--	------

---

Die Funktion löscht den Inhalt der aktuellen Bildschirmseite.

Parameter	keine
Rückgabe	keine
Register	ACCU

START:		
LD A,0		; Seite 0
CALL SETVIEWPAGE		; Anzeigeseite setzen
LD A,1		; Seite 1
CALL SETWRTPAGE		; Schreibseite setzen
CALL AKTPAGE		; Seiten aktivieren

```
CALL CLRAKT      ; Inhalt Seite löschen
CALL CI          ; Warten auf Tastendruck
RET
```

---

<b>CLRINVIS</b>	<b>LÖSCHT EINE UNSICHTBARE BILDSCHIRMSEITE</b>	<b>130H</b>
-----------------	--	-------------

---

Die Funktion dient zum Löschen einer nicht sichtbaren Bildschirmseite, da das Kommando der GDP nur auf der aktuell sichtbaren Seite angewendet werden kann. Es wird die durch die Funktion SETWRTPAGE adressierte Bildschirmseite gelöscht. Die Routine hat eine etwas längere Ausführungszeit, da das Löschen durch Ausgabe von Blöcken erfolgt.

Parameter	keine
Rückgabe	keine
Register	ACCU, BC, DE, HL

---

<b>KILL</b>	<b>BEENDET EINE LAUFENDE FUNKTION DES GRUNDPROGRAMMS</b>	<b>133H</b>
-------------	--	-------------

---

Beendet eine eventuell laufende Funktion des Grundprogramms oder des Monitors. Laufende Funktionen des Monitors oder Grundprogramms sollten beim Wechsel der Bildschirmseite innerhalb eines Anwenderprogramms abgebrochen werden, da sonst das Ergebnis des Anwenderprogramms bei der Seitenumschaltung überschrieben wird.

Parameter	Bildschirmseite (0...3)
Rückgabe	keine
Register	ACCU, HL

---

<b>GETAUSBUF</b>	<b>GIBT EINEN ZEIGER AUF DEN AUSGABEPUFFER ZURÜCK</b>	<b>136H</b>
------------------	---	-------------

---

Der Ausgabepuffer des Grundprogramms umfasst 1096 Bytes, die für 16 Zeilen zu je 85 Zeichen ausreichend sind. Der Puffer kann durch die nachfolgenden Funktionen gefüllt und anschließend auf dem Bildschirm ausgegeben werden.

Parameter	keine
Rückgabe	IX      Startadresse des Puffers
Register	keine

Alle Funktionen zum Beschreiben des Ausgabepuffers schreiben nach der Ausgabe den Wert 00H in den Puffer um das Ende zu markieren.

START:

```
CALL GETAUSBUF    ; IX = Pufferadresse
LD HL,1234H       ; Wert nach HL
.....
```

Das Beispiel wird bei PRTHL fortgesetzt

---

<b>PRTHL</b>	<b>AUSGABE EINER 16 BIT HEXADEZIMALZAHL</b>	<b>139H</b>
--------------	---	-------------

---

Schreibt den Inhalt von HL als Hexadezimalzahl in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Die Ausgabe erfolgt unabhängig vom Wert in HL mit 4 Ziffern.

Parameter	HL	auszugebender Wert
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

Fortsetzung des Beispiels von GETAUSBUF

```
CALL PRTHL        ; HL in Puffer schreiben
LD HL,100         ; X-Position
LD DE,200         ; Y-Position
LD A,21H          ; Schriftgröße
```



```
CALL PRINTBUF      ; Puffer ausgeben
CALL CI            ; Warten auf Tastendruck
RET
```

---

<b>PRTD</b>	<b>AUSGABE EINER 16 BIT DEZIMALZAHL</b>	<b>13Ch</b>
-------------	---	-------------

---

Schreibt den Inhalt von HL als positive Dezimalzahl in den Ausgabepuffer und schleift den Puffer mit der Kennung 00H ab. Die Ausgabe erfolgt unabhängig vom Wert in HL mit 5 Stellen und führenden Leerzeichen.

Parameter	HL	auszugebender Wert
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf Ende-Kennung im Puffer
Register	ACCU	

---

<b>PRTAC</b>	<b>AUSGABE EINER 8 BIT HEXADEZIMALZAHL</b>	<b>13Fh</b>
--------------	--	-------------

---

Schreibt den Inhalt von A als Hexadezimalzahl in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Die Ausgabe erfolgt unabhängig vom Wert im ACCU mit 2 Ziffern.

Parameter	ACCU	auszugebender Wert
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

<b>PRTACD</b>	<b>AUSGABE EINER 8 BIT DEZIMALZAHL</b>	<b>142H</b>
---------------	--	-------------

---

Schreibt den Inhalt von A als Dezimalzahl in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Die Ausgabe erfolgt unabhängig vom Wert im ACCU mit 3 Stellen und führenden Leerzeichen.

Parameter	ACCU	auszugebender Wert
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	ACCU	

---

<b>PRTBIN</b>	<b>AUSGABE EINER 8 BIT ZAHL</b>	<b>145H</b>
---------------	---------------------------------	-------------

---

Schreibt den Inhalt von A als Binärzahl in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Die Ausgabe erfolgt unabhängig vom Wert im ACCU mit 8 Ziffern.

Parameter	ACCU	auszugebender Wert
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

<b>PRINT</b>	<b>AUSGABE EINES TEXTES</b>	<b>148H</b>
--------------	-----------------------------	-------------

---

Schreibt den ab der Adresse in HL beginnenden Text in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Die Textdefinition ab HL muss mit 00H abgeschlossen sein.

Parameter	HL	Startadresse des Textes
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

**TEXT:**

```
.DEFB „Hallo“
.DEFB 0
```

**START:**

```
CALL GETAUSBUF      ; IX = Pufferadresse
```

```

LD HL,TEXT          ; Adresse der Textdefinition
CALL PRINT          ; Text in Puffer übertragen
LD HL,100           ; X-Position
LD DE,200           ; Y-Position
LD A,21H            ; Schriftgröße
CALL PRINTBUF       ; Puffer ausgeben
CALL CI             ; Warten auf Tastendruck
RET

```

---

ZEICH	AUSGABE EINES ASCII ZEICHENS	14BH
-------	------------------------------	------

---

Schreibt ein ASCII Zeichen in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab. Bei der nachfolgenden Ausgabe des Puffers werden die Steuerzeichen CR und LF beachtet.

Parameter	ACCU	auszugebendes Zeichen
	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

BLANK	AUSGABE EINES LEERZEICHENS	14EH
-------	----------------------------	------

---

Schreibt ein Leerzeichen (Space) in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

MINUS	AUSGABE EINES MINUS-ZEICHENS	151H
-------	------------------------------	------

---

Schreibt ein Minus-Zeichen (-) in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

PLUS	AUSGABE EINE PLUS-ZEICHENS	154H
------	----------------------------	------

---

Schreibt ein Plus-Zeichen (+) in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

DOT	AUSGABE EINES PUNKTES	157H
-----	-----------------------	------

---

Schreibt ein Punkt-Zeichen (.) in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

COLON	AUSGABE EINES DOPPELPUNKTES	15AH
-------	-----------------------------	------

---

Schreibt ein Doppelpunkt-Zeichen (:) in den Ausgabepuffer und schließt den Puffer mit der Kennung 00H ab.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

NEWLINE	NEUE ZEILE IM AUSGABEPUFFER	15DH
---------	-----------------------------	------

---

Schreibt einen Zeilenwechsel (Carriage Return und Line Feed) in den Ausgabepuffer. Der Puffer wird mit der Kennung 00H abgeschlossen.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

---

PRINTBUF	AUSGABE DES PUFFERS AUF DEM BILDSCHIRM	160H
----------	--	------

---

Bringt den im Ausgangspuffer zusammengestellten Text auf dem Bildschirm zur Anzeige.

Die Steuerzeichen CR (Carriage Return) und NL (New Line) werden während der Ausgabe beachtet und schalten abhängig von der Schriftgröße auf die nächste Ausbeizeile.

Parameter	HL	X-Position
	DE	Y-Position
	ACCU	Schriftgröße
Rückgabe	keine	
Register	ACCU, HL, DE	
TEXT:		
START:	<pre> .DEFB "HL: ",0      ; Textdefinition  CALL GETAUSBUF      ; Adresse Puffer nach IX LD HL,TEXT          ; Startadresse des Textes CALL PRINT          ; Text in Puffer ablegen LD HL,1234H         ; HL belegen CALL PRTHL          ; Wert in Puffer ablegen LD HL,100           ; X-Position LD DE,200           ; Y-Position LD A,21H            ; Schriftgröße CALL PRINTBUF       ; Puffer ausgeben RET </pre>	

---

TEXTAUS	DIREKTE TEXTAUSGABE AUF DEM BILDSCHIRM	163H
---------	--	------

---

Schreibt einen Text unmittelbar ohne Verwendung des Ausgabepuffers auf den Bildschirm. Die Funktion wird durch einen Versorgungsblock gespeist, der auch die Position des Textes und die Schriftgröße enthält.

Parameter	HL	Adresse des Versorgungsblocks
Rückgabe	HL	zeigt auf die Ende-Kennung des Versorgungsblocks
Register	ACCU, IY	

Aufbau des Versorgungsblocks

	WORD	X-Position
	WORD	Y-Position
	BYTE	Schriftgröße
	BYTE	Zeichenausrichtung
	TEXT	auszugebender Text
	BYTE	Ende-Kennung 00H
TEXT:	<pre> .DEFW 100          ; Y-Position .DEFW 200          ; Y-Position .DEFB 21H,0        ; Schriftgröße und Ausrichtung .DEFB "NDR-NKC"    ; Text </pre>	

```

START:      .DEFB 0
            LD HL,TEXT      ; Versorgungsblock laden
            CALL TEXTAUS    ; ausführen
            CALL CI         ; Warten auf Tastendruck
            RET

```

---

<b>TEXTMULTI</b>	<b>AUSGABE MEHRERER TEXTE</b>	<b>166H</b>
------------------	-------------------------------	-------------

---

Dient zur gleichzeitigen Ausgabe mehrerer Textblöcke auf den Bildschirm.

Parameter	HL	Adresse des Versorgungsblocks
Rückgabe	HL	zeigt auf die Ende-Kennung des Versorgungsblocks
Register	ACCU	

Aufbau des Versorgungsblocks

WORD	X-Position	
WORD	Y-Position	
BYTE	Schriftgröße	
BYTE	Zeichenausrichtung	
TEXT	auszugebender Text	
BYTE	Ende-Kennung 00H	
.....	Wiederholung(en) wie oben	
BYTE	Ende-Kennung 0FFH	

---

<b>PRINTIN</b>		<b>169H</b>
----------------	--	-------------

---

Schreibt einen Text, der unmittelbar hinter dem Aufruf der Funktion codiert ist, in den Ausgabepuffer. Der Programmcode wird unmittelbar nach der Textdefinition fortgeführt.

Für einmalig zu verwendende Texte ist diese Version um 3 Bytes kürzer als die Ausgabe über die Funktion PRINT und belegt kein zusätzliches Register.

Parameter	IX	aktuelle Position im Ausgabepuffer
Rückgabe	IX	zeigt auf die Ende-Kennung des Puffers
Register	keine	

```

START:      CALL GETAUSBUF    ; IX = Pufferadresse
            CALL PRINTIN     ; Text in Puffer übertragen
            .DEFB "Hallo "   ;      auszugebender      Text
            .DEFB „Welt !"   ;      auszugebender      Text
            .DEFB 0          ; Ende-Kennung des Textes
            LD HL,100        ; X-Position
            LD DE,200        ; Y-Position
            LD A,21H         ; Schriftgröße
            CALL PRINTBUF    ; Puffer ausgeben
            CALL CI         ; Warten auf Tastendruck
            RET

```

---

<b>TEXTEIN</b>		<b>16CH</b>
----------------	--	-------------

---

Gestattet die Eingabe durch den Benutzer in einem Texteingabefeld. Die Eingaben werden in einem Puffer gespeichert.

Parameter	HL	Adresse des Versorgungsblocks
	C	Bei C=1 wird das Textfeld umrandet
Rückgabe	IX	Zeigt auf das Ende des Eingabepuffers
Register	ACCU, HL, BC	

Aufbau des Versorgungsblocks

WORD	X-Position des Eingabefeldes
WORD	Y-Position des Eingabefeldes
BYTE	Schriftgröße
BYTE	Schriftart
BYTE	Maximale Anzahl einzeibbarer Zeichen
BYTE	reserviert für tatsächliche Anzahl eingegebener Zeichen
TEXT	Puffer für einzugebende Zeichen

---

TEXTXY	ABFRAGE VON BENUTZEREINGABEN	16FH
--------	------------------------------	------

---

Gestattet die Eingabe durch den Benutzer in einem umrandeten Texteingabefeld. Die Eingaben werden in einem 85 Zeichen umfassenden Puffer gespeichert und können anschließend als Zahl, Ausdruck oder als Text weiterverarbeitet werden.

Parameter	HL	X-Position
	DE	Y-Position
	ACCU	Schriftgröße
	B	Anzahl der einzugebenden Zeichen (Feldbreite)
Rückgabe	IX	Zeigt auf das Ende des Eingabepuffers
Register	ACCU, HL, BC	

TEXT:

```
.DEFB "Eingabe"      ; auzugebender Text
.DEFB 0              ; Endekennung
```

START:

```
LD HL,100            ; X-Position
LD DE,200            ; Y-Position
LD A,21H             ; Schriftgröße
LD IX,TEXT           ; Startadresse Text
CALL TEXTPRINT       ; ausführen
```

```
LD HL,200            ; X-Position
LD DE,200            ; Y-Position
LD A,21H             ; Schriftgröße
LD B,15              ; Feldbreite
CALL TEXTXY          ; ausführen
.....
```

---

GETHL	EINGABE ALS ZAHL INTERPRETIEREN	172H
-------	---------------------------------	------

---

Interpretiert einen zuvor mit TEXTXY ermittelten Ausdruck als Zahl. Die Eingabe kann hexadezimal oder dezimal mit oder ohne Vorzeichen erfolgen.

Parameter	IX	Pointer auf die Zeichenkette
Rückgabe	HL	Wert des eingegebenen Ausdrucks
	CARRY	Fehlerhafte Eingabe
Register	ACCU, BC, DE, IX	

Gültige Eingaben (Beispiele)

100	Hexadezimal 100H
-1	Hexadezimal FFFFH
+41	Hexadezimal 41
#100	Dezimal 100
-#10	Dezimal 65526

Das Beispiel setzt den Beispielcode von TEXTXY fort

```
CALL GETHL           ; Eingabe interpretieren
CALL GETAUSBUF       ; Pufferadresse holen
CALL PRTHL           ; in Ausgabepuffer
LD HL,200            ; X-Position
LD DE,180            ; Y-Position
```

```
LD A,21H          ; Schriftgröße
CALL TEXTAUS      ; Ergebnis ausgeben
CALL CI           ; Warten auf Tastendruck
RET
```

---

<b>GETTEXT</b>	<b>RÜCKGABE DES EINGEGEBENEN TEXTES</b>	<b>175H</b>
----------------	---	-------------

---

Die Funktion gibt einen Pointer auf einen mit TEXTXY eingegebenen Text zurück. Ab dieser Stelle liegt der vom Benutzer eingegebene Text, welcher mit 00H als Abschluss versehen ist.

Parameter	keine	
Rückgabe	IX	zeigt auf den eingegebenen Text
Register	keine	

---

<b>EXPR</b>	<b>AUSWERTUNG VON AUSDRÜCKEN</b>	<b>178H</b>
-------------	----------------------------------	-------------

---

Interpretiert einen zuvor mit TEXTXY eingegebenen Text unter Berücksichtigung von Symbolen und mathematischen Ausdrücken.

Parameter	IX	Zeiger auf den eingegebenen Text
Rückgabe	HL	Ergebnis der Berechnung
	CARRY	ungültiger Ausdruck
Register	ACCU, BC, DE, IX	

Gültige Eingaben (Beispiele)

100 + 200	Addition
100 - 200	Subtraktion
200.W + 10.B	Wortlänge

---

<b>GETPARA</b>	<b>PARAMETEREINGABE</b>	<b>17BH</b>
----------------	-------------------------	-------------

---

Gestattet die Eingabe durch den Benutzer in einem umrandeten Texteingabefeld mit gleichzeitiger Ausgabe eines Textes vor dem Textfeld. Die Eingaben werden in einem Puffer gespeichert und können anschließend als Zahl, Ausdruck oder als Text weiterverarbeitet werden.

Parameter	HL	Adresse des Versorgungsblocks
Rückgabe	HL	Eingegebener Wert
	IX	Zeigt auf das Ende des Eingabepuffers
	CARRY	ungültiger Ausdruck
Register	ACCU	

Aufbau des Versorgungsblocks

WORD	X-Position Textausgabe
WORD	Y-Position Textausgabe
BYTE	Schriftgröße Text und Eingabefeld
BYTE	Schriftart Text und Eingabefeld
TEXT	auszugebender Text
BYTE	Ende-Kennung Text 00h
WORD	X-Position Eingabefeld
BYTE	Breite des Eingabefeldes

Die Funktion setzt die aktuellen Positionen im Grafikprozessor der GDP Baugruppe. Bei diesen Koordinaten erfolgt die nächste Ausgabe. SETPEN und ERAPEN werden beachtet.

Parameter	HL	X-Position (0 ... 511)
	DE	Y-Position (0 ... 255)

Rückgabe	keine
Register	ACCU

```

START:
        LD HL,0           ; X-Position
        LD DE,0           ; Y-Position
        CALL MOVETO       ; Grafikcursor setzen
        .....
    
```

Zeichnet eine Linie von der letzten Ausgabeposition zur angegebenen Position unter Verwendung des Bresenham-Algorithmus. SETPEN und ERAPEN werden beachtet.

Parameter	HL	X-Position (0 ... 511)
	DE	Y-Position (0 ... 255)

Rückgabe	keine
Register	ACCU, DE, HL

Das Beispiel ist die Fortsetzung von MOVETO

```

        LD HL,511        ; neue X-Position
        LD DE,255        ; neue Y-Position
        CALL DRAWTO      ; Linie zeichnen
        CALL CI          ; Werten auf Tastendruck
        RET
    
```

Die Schildkrötengrafik (Turtlegrafik) bietet einfache Funktionen zum Zeichnen von Linien wobei die Richtung durch einen Winkel angegeben wird. Nach einem RESET liegt die Position der Schildkröte in der Mitte des Bildschirms und zeigt nach oben.

Parameter	HL	X-Position (0 ... 511)
	DE	Y-Position (0 ... 255)
	BC	Winkel im Uhrzeigersinn, 0=oben

Rückgabe	keine
Register	ACCU, BC, DE, HL

Im Gegensatz zum originalen Grundprogramm kann mit MOVE nicht gleichzeitig der Drehwinkel der Schildkröte angegeben werden. Dazu muss die Funktion DREHE mit einem zusätzlichen Aufruf verwendet werden.

```

START:
        LD HL,256        ; X-Position
        LD DE,128        ; Y-Position
        CALL MOVE        ; Position setzen
        .....
    
```

Das Beispiel wird bei SCHREITE fortgesetzt

Bewegt die Schildkröte um die angegebene Anzahl von Schritten (Pixel) vorwärts.

Parameter	HL	Anzahl der Schritte
Rückgabe	keine	
Register	ACCU, BC, DE, HL	

Fortsetzung des Beispiels

```
LD HL,100           ; 100 Schritte
CALL SCHREITE       ; Linie nach oben zeichnen
CALL CI             ; Warten auf Tastendruck
RET
```

Bewegt die Schildkröte um die angegebene Anzahl an 16tel Schritten vorwärts. Durch die Verwendung von 16tel Schritten können Kurven exakter dargestellt werden. Innerhalb der Turtle-Routinen werden alle Berechnungen mit 16facher Genauigkeit ausgeführt.

Parameter	HL	Anzahl der 16tel Schritte
Rückgabe	keine	
Register	ACCU, BC, DE, HL	

**START:**

```
LD HL,256           ; X-Position
LD DE,128           ; Y-Position
LD BC,0             ; Winkel
CALL MOVE           ; Anfangsposition setzen
```

```
LD B,72             ; 72 Liniensegmente
```

**LOOP:**

```
LD HL,8             ; 8/16 Pixel
CALL SCHR16TEL      ; kurze Linie zeichnen
```

Das Beispiel wird bei DREHE fortgeführt

Dreht die Schildkröte um die angegebene Anzahl Grad im Uhrzeigersinn. Die Winkelangabe darf den Bereich von 0 bis 359 Grad überschreiten, es findet eine automatische Normierung statt.

Parameter	HL	Winkel in Grad
Rückgabe	keine	
Register	keine	

Das Beispiel ist die Fortsetzung von SCHR16TEL

```
LD HL,5             ; 5 Grad
CALL DREHE          ; Turtle drehen
DJNZ LOOP           ; bis Kreis abgeschlossen
CALL CI             ; Warten auf Tastendruck
RET
```



Nachfolgende Aufrufe von SCHREITE und SCHR16TEL hinterlassen keine sichtbare Spur, es wird lediglich die Position verändert.

Parameter	keine
Rückgabe	keine
Register	keine

Ein Beispiel findet sich bei der Funktion RUNTER

Nachfolgende Aufrufe von SCHREITE und SCHR16TEL hinterlassen eine sichtbare Spur.

Parameter	keine
Rückgabe	keine
Register	keine

Das Beispiel zeigt das Zeichnen eines unterbrochenen Kreises

```
START:
        LD HL,256           ; X-Position
        LD DE,128           ; Y-Position
        LD BC,0             ; Winkel
        CALL MOVE           ; Anfangsposition setzen

        LD B,36             ; 36 Liniensegmente

LOOP:
        LD HL,4             ; 4 Pixel
        CALL SCHREITE       ; Linie zeichnen
        LD HL,5             ; 5 Grad
        CALL DREHE          ; Turtle drehen
        CALL HEBE           ; Turtle anheben
        LD HL,4             ; 4 Pixel
        CALL SCHREITE       ; Linie zeichnen
        LD HL,5             ; 5 Grad
        CALL DREHE          ; Turtle drehen
        CALL SENKE          ; Turtle absenken
        DJNZ LOOP           ; bis Kreis abgeschlossen
        .....

```

Das Beispiel wird bei TURTLE fortgeführt

Zeichnet eine stilisierte Schildkröte auf dem Bildschirm unter Beachtung der aktuellen Position und Richtung.

Parameter	keine
Rückgabe	keine
Register	ACCU, BC, DE, HL

Das Beispiel ist die Fortführung von RUNTER

```
        CALL TURTLE        ; Turtle anzeigen
        CALL CI            ; Warten auf Tastendruck
        RET

```

Die Figur wird unter Verwendung der Kurzvektoren des Grafikprozessors EF9366 auf der Baugruppe GDP64K oder GDP64HS gezeichnet.

Parameter IX Zeiger auf die Figur-Definition  
B Größe der Figur (0 ... 3)

Rückgabe keine  
Register ACCU

Figur-Definition

1	Linie nach oben	N
2	Linie nach rechts oben	NO
3	Linie nach rechts	O
4	Linie nach rechts unten	SO
5	Linie nach unten	S
7	Linie nach links unten	SW
8	Linie nach links	W
9	Linie nach links oben	NW
10	unsichtbar bewegen	wie HEBE
11	sichtbar bewegen	wie SENKE
12	Schreibmodus erzwingen	wie SETPEN
13	Löschmodus erzwingen	wie ERAPEN
0	Ende-Kennung	

FIG:

```
.DEFB 1,2,3,4      ; Vektoren
.DEFB 5,6,7,8
.DEFB 9
.DEFB 0             ; Endekennung
```

START:

```
LD IX,FIG           ; Figurdefinition
LD B,3              ; Größe
CALL FIGUR          ; Figur zeichnen
CALL CI             ; Warten auf Tastendruck
RET
```

Wandelt ein im ACCU vorliegendes ACSII Zeichen in Großschrift. Die deutschen Umlaute werden korrekt umgewandelt.

Parameter ACCU zu wandelndes ASCII Zeichen  
Rückgabe ACCU ASCII Zeichen in Großbuchstaben  
Register keine

START:

```
CALL CI             ; Zeichen von Tastatur holen
CALL UPPER          ; Zeichen in Großbuchstaben wandeln
CP 0Dh              ; Eingabetaste?
RET Z               ; Ende
CALL CMD            ; Zeichen ausgeben
JR START            ; neu starten
```

WAITMS	VARIABLE WARTEZEIT IN MILLISEKUNDEN		19FH
Wartet eine vorgegebene Anzahl von Millisekunden. Die Routine ist für eine Taktfrequenz von 4 MHz ausgelegt. Bei höheren Taktfrequenzen muss der Wert entsprechend angepasst werden.			
Parameter	HL	Wartezeit in Millisekunden	
Rückgabe	keine		
Register	ACCU		

WAIT100MS	WARTET 100 MILLISEKUNDEN		1A2H
Unterbricht die Programmausführung für 100 Millisekunden bei 4 MHz Taktfrequenz.			
Parameter	keine		
Rückgabe	keine		
Register	ACCU		

ADJ360	BEREICHSANPASSUNG FÜR WINKEL		1A5H
Dient zur Anpassung eines Winkels im Anschluss an eine Berechnung zur Verwendung in der Turtle-Grafik.			
Parameter	HL	anzupassender Wert	
Rückgabe	HL	Winkel im Bereich 0 bis 359	
Register	ACCU, BC, DE		

SIN	BERECHNET DEN SINUS EINES WINKELS		1A8H
Berechnet den Sinus-Wert eines übergebenen Winkels. Der Winkelwert wird automatisch an den gültigen Bereich angepasst.			
Parameter	HL	Winkel in Grad	
Rückgabe	HL	256 * SIN(Winkel)	
Register	ACCU, BC, DE		

COS	BERECHNET DEN COSINUS EINES WINKELS		1ABH
Berechnet den Cosinus-Wert eines übergebenen Winkels. Der Winkelwert wird automatisch an den gültigen Bereich angepasst.			
Parameter	HL	Winkel in Grad	
Rückgabe	HL	256 * COS(Winkel)	
Register	ACCU, BC, DE		

CPLHL			1AEH
Bildet das Zweierkomplement des übergebenen Wertes.			
Parameter	HL	übergebener Wert	
Rückgabe	HL	Zweierkomplement	
Register	ACCU		

HEXDEZ		1B1H
Umwandlung einer 16 Bit Zahl in eine Dezimalzahl.		
Parameter	HL	Umzuwandelnder Wert
Rückgabe	C	1. Stelle der Dezimalzahl
	D	2. Und 3. Stelle der Dezimalzahl
	E	4. Und 5. Stelle der Dezimalzahl
Register	ACCU	
DEZHEX		1B4H
Umwandlung einer Dezimalzahl in einen 16 Bit Wert.		
Parameter	CDE	Dezimalzahl
Rückgabe	HL	umgewandelter Wert
Register	ACCU	
LENGTH		1B7H
Berechnet die Länge eines Z80 Assemblerbefehls.		
Parameter	HL	Zeiger auf den Befehlscode
Rückgabe	B	Länge des Befehls in Byte
Register	ACCU, DE, HL	
SIST	STATUS SERIELLE SCHNITTSTELLE LESEN	1BAH
Prüft die serielle Schnittstelle auf Vorhandensein eines Zeichens im Eingangspuffer.		
Parameter	keine	
Rückgabe	ZERO	kein Zeichen vorhanden
Register	ACCU	
SI	ZEICHEN VON SERIELLER SCHNITTSTELLE LESEN	1BDH
Liest ein Zeichen von der seriellen Schnittstelle. Die Funktion wartet so lange, bis ein Zeichen zum Einlesen bereitsteht.		
Parameter	keine	
Rückgabe	ACCU	gelesenes Zeichen
Register	keine	
SOST		1C0H
Prüft ob der Ausgabepuffer der seriellen Schnittstelle Zeichen aufnehmen kann		
Parameter	keine	
Rückgabe	ZERO	Sendepuffer ist voll
Register	ACCU	

SO	ZEICHEN AUF SERIELLE SCHNITTSTELLE AUSGEBEN	1C3H
----	---	------

Gibt ein Zeichen über die serielle Schnittstelle aus. Die Funktion wartet so lange, bis der Ausgabepuffer frei ist

Parameter	C	auszugebendes Zeichen
Rückgabe	keine	
Register	keine	

SETBAUD	BAUDRATE SERIELLE SCHNITTSTELLE SETZEN BAUD	1C6H
---------	---	------

Mit dieser Funktion kann die Baudrate der seriellen Schnittstelle geändert werden. Nach dem Systemstart ist die Schnittstelle auf 9600 Baud, 8 Bit, kein Parity und 1 Stopp-Bit eingestellt.

Parameter	ACCU	Baudrate gemäß Tabelle
Rückgabe	keine	
Register	ACCU	
Baudrate	0	ungültig, externer Taktgenerator
	1	50 Baud
	2	75 Baud
	3	110 Baud
	4	135 Baud
	5	150 Baud
	6	300 Baud
	7	600 Baud
	8	1200 Baud
	9	1800 Baud
	10	2400 Baud
	11	3600 Baud
	12	4800 Baud
	13	7200 Baud
	14	9600 Baud
	15	19200 Baud

GETDATE	DATUM VON UHR3 LESEN	1C9H
---------	----------------------	------

Liest das Datum aus einer angeschlossenen Baugruppe UHR3 aus.

Parameter	keine	
Rückgabe	E D C	Tag, Monat, Jahr
	B	Jahrhundert
Register	ACCU	

GETTIME	UHRZEIT UND WOCHENTAG VON UHR3 LESEN	1CCH
---------	--------------------------------------	------

Liest die Uhrzeit aus einer angeschlossenen Baugruppe UHR3 aus.

Parameter	keine	
Rückgabe	E D C	Sekunde, Minute, Stunde
	B	Wochentag, 1=Montag ... 7=Sonntag
Register	ACCU	

USB_START	STARTET DAS USB SYSTEM	1CFH
-----------	------------------------	------

Die Funktion muss vor der Benutzung der USB Routinen einmalig aufgerufen werden um das Vorhandensein der Hardware zu testen und zurückzusetzen.

Parameter      keine  
Rückgabe      CARRY    Hardware nicht gefunden  
Register      ACCU

```

FNAME:
        .DEFB "TEST.DAT"    ; Dateiname
        .DEFB 0             ; Endekennung

START:
        CALL USB_START     ; USB initialisieren
        RET C              ; Ausgang, keine Hardware

```

Das Beispiel wird bei USB\_ISDISK fortgeführt

USB_ISDISK	TEST AUF VERBUNDENEN DATENRÄGER	1D2H
------------	---------------------------------	------

Die Funktion testet, ob ein Datenträger am VDIP1 Modul angesteckt ist.

Parameter      keine  
Rückgabe      CARRY    kein Datenträger gefunden oder nicht lesbar  
Register      ACCU

Fortführung des Beispiels von USB\_START

```

        CALL USB_ISDISK    ; Test auf Datenträger
        RET C              ; Ausgang, kein Datenträger

```

Das Beispiel wird bei USB\_DIR fortgeführt

USB_DIR	TEST AUF VORHANDENSEIN EINER DATEI	1D5H
---------	------------------------------------	------

Testet, ob eine bestimmte Datei im Stammverzeichnis des angeschlossenen Datenträgers vorhanden ist.

Parameter      IX      Zeiger auf einen gültigen Dateinamen  
Rückgabe      DE      Länge der Datei  
                 CARRY    Datei nicht vorhanden  
Register      ACCU

Fortführung des Beispiels von USB\_ISDISK

```

        LD IX,FNAME        ; Pointer auf Dateiname
        CALL USB_DIR       ; Test auf Vorhandensein Datei
        RET C              ; Ausgang, Datei nicht vorhanden

```

Das Beispiel wird bei USB\_LOAD fortgeführt

USB_LOAD	DATEI KOMPLETT LADEN	1D8H
----------	----------------------	------

Liest den kompletten Inhalt einer Datei in den Speicher ein. Es findet keine Kontrolle statt, ob die Datei in den Speicher passt oder ob durch den Ladevorgang die Variablen oder der Stack überschrieben werden.

Parameter      IX      Zeiger auf einen gültigen Dateinamen  
                 HL      Zieladresse im Speicher  
Rückgabe      DE      Anzahl gelesener Zeichen  
                 CARRY    Fehler, Datei nicht gefunden  
Register      ACCU

Fortführung des Beispiels von USB\_DIR

```

START:
LD IX,FNAME      ; Pointer auf Dateiname
LD HL,9000H      ; Zieladresse im Speicher
CALL USB_LOAD    ; Ganze Datei laden
RET

```

---

<b>USB_SAVE</b>	<b>DATEI KOMPLETT SCHREIBEN</b>	<b>1DBH</b>
-----------------	---------------------------------	-------------

---

Schreibt einen kompletten Speicherbereich in eine Datei. Falls die Datei schon vorhanden ist wird der Inhalt überschrieben. Wenn die Datei noch nicht vorhanden ist wird sie angelegt. Es findet keine Kontrolle der übergebenen Parameter statt.

Parameter	IX	Zeiger auf einen gültigen Dateinamen
	HL	Startadresse im Speicher
	DE	Anzahl der zu sichernden Bytes
Rückgabe	CARRY	Fehler
Register	ACCU	

```

FNAME:
.DEFB "TEST.DAT" ; Dateiname
.DEFB 0           ; Endekennung

```

```

START:
LD IX,FNAME      ; Pointer auf Dateiname
LD HL,9000H      ; Zieladresse im Speicher
LD DE,1000H      ; Anzahl zu sichernden Bytes
CALL USB_SAVE    ; Ganze Datei schreiben
RET

```

---

<b>USB_OPW</b>	<b>DATEI ZUM SCHREIBEN ÖFFNEN</b>	<b>1DEH</b>
----------------	-----------------------------------	-------------

---

Öffnet eine bestimmte Datei zum Beschreiben mit Daten. Der Dateizeiger verweist auf das erste Zeichen innerhalb der Datei. Nach dem Beschreiben muss die Datei wieder geschlossen werden.

Parameter	IX	Zeiger auf einen gültigen Dateinamen
Rückgabe	CARRY	Fehler
Register	ACCU	

---

<b>USB_WR</b>		<b>1E1H</b>
---------------	--	-------------

---

Schreibt Daten in eine zuvor mit USB\_OPW geöffnete Datei.

Parameter	HL	Zeiger auf den Beginn der Daten im Speicher
	DE	Anzahl der zu schreibenden Bytes
Rückgabe	CARRY	Fehler
Register	ACCU	

---

<b>USB_OPR</b>	<b>DATEI ZUM LESEN ÖFFNEN</b>	<b>1E4H</b>
----------------	-------------------------------	-------------

---

Öffnet eine bestimmte Datei zum Lesen von Daten. Der Dateizeiger verweist auf das erste Zeichen innerhalb der Datei. Nach dem Lesen von Daten muss die Datei wieder geschlossen werden. Es kann immer nur eine Datei zur gleichen Zeit zum Lesen geöffnet werden, da sich die nachfolgenden Lesevorgänge auf die geöffnete Datei beziehen.

Parameter	IX	Zeiger auf einen gültigen Dateinamen
Rückgabe	CARRY	Fehler
Register	ACCU	

USB_RD		1E7H
Liest Daten aus einer zuvor mit USB_OPR geöffneten Datei.		
Parameter	HL	Zeiger auf die Zieladresse im Speicher
	DE	Anzahl der zu lesenden Bytes
Rückgabe	CARRY	Fehler
Register	ACCU	
USB_CLF		DATEI SCHLIEßEN 1EAH
Schließt eine zuvor mit USB_OPR oder USB_OPW geöffnete Datei.		
Parameter	IX	Zeiger auf einen gültigen Dateinamen
Rückgabe	CARRY	Fehler
Register	ACCU	
USB_SEK		DATEI POINTER SETZEN 1EDH
Stellt den Dateizeiger einer geöffneten Datei an eine bestimmte Stelle ab dem Beginn der Datei.		
Parameter	DE	Position innerhalb der Datei
Rückgabe	CARRY	Fehler
Register	ACCU	
<pre> DATEN:         .DEFB "TEST.DAT"    ; Dateiname         .DEFB 0              ; Endekennung  START:         LD IX,DATEN         ; Pointer auf Dateiname         CALL USB_OPR        ; Datei öffnen         RET C               ; Ausgang bei Fehler         LD DE,100H          ; 100H Bytes         CALL USB_SEK        ; Dateipointer setzen         RET C               ; Ausgang bei Fehler         CALL USB_RD         ; Byte aus Datei lesen         CALL USB_CLF        ; Datei schließen         RET </pre>		
USB_DLF		DATEI LÖSCHEN 1F0H
Löscht eine Datei vom Datenträger.		
Parameter	IX	Zeiger auf einen gültigen Dateinamen
Rückgabe	CARRY	Fehler
Register	ACCU	
<pre> FILE:         .DEFB "TEST.DAT"    ; Zu löschende Datei         .DEFB 0              ; Endekennung  START:         LD IX,FILE          ; Pointer auf Dateiname         CALL USB_DLF        ; Datei löschen         RET </pre>		



USB_REN	DATEI UMBENENNEN	1F3H
---------	------------------	------

Benennt eine vorhandene Datei um.

Parameter	IX	Zeiger auf den alten Dateinamen
	IY	Zeiger auf den neuen Dateinamen
Rückgabe	CARRY	Fehler
Register	ACCU	
OLD:		
	.DEFB "OLD.DAT"	; Umzubenennende Datei
	.DEFB 0	; Endekennung
NEW:		
	.DEFB "NEW.DAT"	; Neuer Dateiname
	.DEFB 0	; Endekennung
START:		
	LD IX,OLD	; Alter Dateiname
	LD IY,NEW	; Neuer Dateiname
	CALL USB_REN	; OLD.DAT umbenennen
	RET	

USB_READ	BYTE AUS DATEI LESEN	1F6H
----------	----------------------	------

Liest ein einzelnes Byte aus einer zum Lesen geöffneten Datei ab der aktuellen Position des Dateizeigers.

Parameter	keine	
Rückgabe	ACCU	gelesenes Byte
	CARRY	Fehler, Ende der Datei erreicht
Register	keine	

USB_WRITE	BYTE IN DATEI SCHREIBEN	1F9H
-----------	-------------------------	------

Schreibt ein einzelnes Byte in eine zum Schreiben geöffneten Datei an die aktuelle Position des Dateizeigers.

Parameter	ACCU	zu schreibendes Byte
Rückgabe	ACCU	00h = akzeptiert, 0FFh = abgewiesen
	CARRY	Fehler, Puffer voll
Register	keine	

USB_REM	DATENTRÄGER AUSWERFEN	1FCH
---------	-----------------------	------

Mit dieser Funktion kann ein Datenträger abgemeldet werden. Dabei wird das USB-Modul zurückgesetzt, der Datenträger ist stromlos. Nach dem Auswerfen muss das USB-Modul neu initialisiert werden.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	immer 00h

# Sonderzeichen

## Deutsche Umlaute

Die Routinen zur Ausgabe von Texten auf dem Bildschirm wurden um die Möglichkeit zur Ausgabe von deutschen Umlauten ergänzt. Alle Routinen im Grundprogramm unterstützen diese Umlaute.

Die Umlaute werden manuell gezeichnet und sind deswegen etwas langsamer als normale Zeichen. Es werden alle Schriftgrößen unterstützt, die Ausgabe großer Zeichen dauert länger als die Ausgabe von Zeichen in kleinen Schriftgrößen.

In eigenen Programmen bietet es sich an, folgende Konstanten zu definieren:

```
AE      .equ 0C4h      ; Umlaut A
OE      .equ 0D6h      ; Umlaut O
UE      .equ 0DCh      ; Umlaut U
ae      .equ 0E4h      ; Umlaut a
oe      .equ 0F6h      ; Umlaut o
ue      .equ 0FCh      ; Umlaut u
sz      .equ 0DFh      ; Umlaut s
us      .equ 05Fh      ; Unterstrich _
```

Der Unterstrich ist zwar im Zeichensatz der Grafikprozessors EF9366 enthalten, ist aber auf einer ASCII-Position angesiedelt, die nicht mit üblichen Personal-Computern kompatibel ist.

### Beispielcode

```
TEXT:
        .defw 256,200    ; Position
        .defb 21h,0      ; Schriftgröße
        .defb "Test "    ; Text
        .defb AE,OE,UE   ; Sonderzeichen
        .defb 0          ; Ende des Textes

START:
        LD HL,TEXT       ; Text laden
        CALL TEXTSETAUS   ; Text ausgeben
        RET
```

## Flags im ROM

### Portadresse für USB

Für die Verwendung der Baugruppe IOE mit VDIP1 USB Modul oder für die Baugruppe IO-USB kann die Portadresse mit einem Byte im ROM definiert werden. Dazu kann die gewünschte Portadresse an der Adresse 3FFDh am Ende des zweiten ROMs eingetragen werden. Als Standard ist die Adresse 30h vorgegeben.

### Steuerung des GP

An der Adresse 3FFEh am Ende des zweiten ROMs befindet sich ein Byte mit Schaltern für das Verhalten des Grundprogramms. Jedes Bit dieses Bytes hat eine andere Bedeutung gemäß nachstehender Tabelle.

Bit 0	Einblenden der Routinen zur Bankumschaltung
Bit 1	Anzeigen der Versionsnummer in der Kopfzeile
Bit 2	unbenutzt
Bit 3	unbenutzt
Bit 4	Baugruppe UHR3 vorhanden
Bit 5	reserviert
Bit 6	reserviert
Bit 7	Aktivieren des Interrupt Mode 2

Als Standardwert ist hier 00010001b entsprechend 11h eingetragen, somit ist die Bankumschaltung und die Zeitanzeige aktiv.

### Interrupt Mode 2

Im Interrupt Mode 2 wird die Adresse der auszuführenden Interrupt-Routine durch den Inhalt des Registers I und dem Wert auf dem Datenbus gebildet. Die auslösende Hardware legt dazu ein Datenbyte auf den Bus und löst dann die Unterbrechung aus. Die Adresse berechnet sich wie folgt.

$$\text{Adresse} = 256 * I + \text{Daten}$$

Wenn der Interrupt Mode 2 des Grundprogramms aktiv ist, wird das I-Register mit dem Wert 61h geladen. Bei einem Interrupt wird also ein Sprung nach 61xxh ausgelöst. Der RAM-Bereich von 6100h bis 6180h ist für Interrupt-Routinen im Interrupt Mode 2 freigehalten und kann durch den Anwender beliebig belegt werden.

### Erweiterung des GP

Im Anschluss an die Initialisierung des Grundprogramms wird ein Sprung an die Adresse 601Bh im RAM Bereich ausgeführt. Dort ist zunächst nur ein RET-Befehl hinterlegt, der die Initialisierung abschließt.

Durch den Anwender kann an der Adresse 601B ein Sprung zu einer eigenen Routine hinterlegt werden, die dann nach jedem RESET angesprungen wird. Innerhalb einer solchen Routine kann z.B. eigene Hardware initialisiert werden. Die eigene Initialisierungsroutine muss mit einem RET-Befehl enden.

Um diesen Vorgang zu automatisieren kann das Grundprogramm gepatcht werden. Dazu ist an den Adressen 3FEBh bis 3FEDh ein Sprung zu einer eigenen Initialisierungsroutine einzutragen. Wenn zum Beispiel ab Adresse 4000h eine eigene Routine im ROM Verfügbar ist, müssten folgende Änderungen am GP vorgenommen werden.

3FED	C3	JMP-Befehl
3FEE	00	Adresse 4000h LSB
3FEF	40	Adresse 4000h MSB