Grundprogramm 2019

Rolf-Dieter Klein (www.rdklein.de)
Andres Rohmann (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 immer wieder als Betatester unterstützt hat. Wir wünschen Ihnen / Euch viel Spaß bei der Nutzung des neuen Systems.

Andreas Rohmann, 2018

> Seit der Version GP2018v2 sind viele neue Ideen und neue Funktionen hinzugekommen, so dass sich das Betriebssystem als GP-2019 in der vorliegenden Version vorstellt. Dabei habe ich die hervorragende Doku von Andreas weiter geschrieben. Änderungen gegenüber der vorherigen Version GP2018v2 sind kenntlich gemacht. Vielen Dank wiederum an unseren Betatester Jens und Lektor Marcus.

Steffen Reimer , 2019

Inhalt

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 8 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 werden weitere ROA64 mit 64 kB RAM oder eine SRAM1024 mit mindestens 512 kB empfohlen.

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 automatisch gestartet (gebootet) 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 denn korrekt, wenn zuvor „Bank wechseln“ oder „Bank einblenden“ ausgeführt wurde. Abhilfe schafft der Einsatz der BANKBOOT2, diese erlaubt eine direkte Rückmeldung und zeigt immer korrekt die eingestellte Bank.

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.

CASneo (optional)

Eine CASNEO kann verwendet werden. In der vorliegenden Version wird die CASNEO Baugruppe ausschließlich als „SD-Festplatte betrieben“.

Die bisherige CAS, als Kassetten- bzw. Tonaufzeichnungsgerät wird nicht mehr unterstützt.

PROMER (optional)

Die PROMER-Funktionen mussten leider aus Platzgründen aus den ersten 16 kB des Grundprogrammes entfernt werden.

Diese werden aber als nachladbare Ergänzung angeboten werden.

SER (wird nicht mehr unterstützt)

Das Grundprogramm in der Version GP-2019 stellt aus Kapazitätsgründen leider keine Routinen zur Nutzung der seriellen Schnittstelle SER mehr bereit.

> Diese können bei Bedarf und je nachdem welche Hardware (SER oder SIO) Verwendung finden soll, als umfangreiche „SER Remote“ (für SER) Funktion in den „Erweiterungs-“ Adressraum ab 7000h geladen werden. GP2019 wird die Anwesenheit erkennen und im Extended Menü einen entsprechenden Menüeintrag bereitstellen.

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 2019 verwendet werden. Das Grundprogramm erkennt automatisch, wenn das BASIC ROM vorhanden ist und stellt einen Menüpunkt zum Aufruf bereit.

EZASS

Der Assembler kann nicht mehr mit dem Grundprogramm genutzt werden, da der Speicherbereich ab 6000h mit dem System-RAM belegt sein muss. Stattdessen wird der Einsatz eines Editor-Assemblers (z.B. AC1-EDAS*4) empfohlen, der die Listings auch mit dem für das Verständnis eines Programmes notwendigen Kommentaren speichert.

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 kann jetzt über die Sprungtabelle erfolgen.

Optimaler Systemaufbau

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

- CPUZ80
- GDP64K oder GDP64HS
- KEY oder KEY3 mit Tastatur mit Firmware KEYr4_2018_04
- BANKBOOT2, mit Einschränkungen bish. BANKBOOT
- SRAM1024 batteriegepuffert
- IOE-VDIP1 oder IO-USB
- CASNEO (ohne Bedienteil als „Festplatte“)
- UHR3
- IOE als Centronics

Auf der Baugruppe BANKBOOT

- Steckplatz 1: EPROM 27C64 mit Grundprogramm 2019 #0000
- Steckplatz 2: EPROM 27C64 Grundprogramm 2019 #2000
- Steckplatz 3: EEPROM vom Typ AT28C64 oder kompatible, ggf. BASIC oder AC1-EDAS
- Steckplatz 4: 8 kB RAM vom Typ HM6264 oder kompatible, als System-RAM

Mit der empfohlenen Konfiguration ist es z.B. möglich:

1. auf dem Betriebssystem der BANKBOOT zu verbleiben (Speicherbereich 0000…7FFFh) und den Adressbereich 8000..FFFFh wahlweise mit „Bank einblenden“ von verschiedenen Banken zu verwenden.

2. In komplette RAM-Banken (Adressbereich 0000..FFFFh) selbstständige Betriebssysteme zu laden und mit „Bank wechseln“ zu starten. So könnten
   auf der BANKBOOT das **GP2019 mit FLOMONCN4000** im 3. EPROM,
   in einer weiteren RAM-Bank das **GP2019 mit BASIC**,
   auf einer anderen RAM-Bank das **GP2019 mit dem AC1-EDAS**,
   in einer Dritten das **GP 3.0 mit GOSI** und
   in einer weiteren Bank eine **AC1-Simulation**
   lauffähig für die Benutzung bereit liegen. Mit „Bank wechseln“ wird dann das jeweilige Betriebssystem gestartet.

   ACHTUNG!
   GP2018 oder GP2019 zusammen mit FLOMON4000 (von Jens für CASneo) ist nur auf der BANKBOOT lauffähig, da das FLOMON zwischen Bank E und BANKBOOT hin und her wechselt und auf der BANKBOOT sein lauffähiges System finden will. In den übrigen Banken wird dabei eine RAM-Disk realisiert.

Dokumentationsstand

Dieses Dokument bezieht sich auf die 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

GP2019 vom 27. Mai.2019 Prüfsumme: 79D5

Neue dritte Menüseite.
Neuer Unterprogramm-Sprungverteiler über RST 08h mit Programmnummer
CASneo als „SD-Festplatte“
TYPE zum Anzeigen von Texten
Read.Me-Datei vom USB-Stick am BS anzeigen
Allgemeine Fehlerbehebungen

Das Menüsystem

Das Grundprogramm verfügt über drei Menüseiten, deren Menüpunkte jeweils durch einen einfachen Tastendruck ausgelöst werden können. Mit der Leertaste kann zwischen den Menüseiten gewechselt werden.

In der Kopfzeile werden folgende Informationen angezeigt:

- die aktuell gezeigte Bildschirmseite
- die aktuell über die BankBoot eingestellte RAM-Bank z.B.:
  RAM: 0; die BankBoot ist aktiv 0000..7FFFh und 0.8000..0.FFFFh der RAM-Bank 0 oder
  RAM: 3; die BankBoot ist aktiv 0000..7FFFh und 3.8000..3.FFFFh der RAM-Bank 3
  Bank: 1; BankBoot ist aus und die kpl. Bank 1 = 1.0000..1.FFFFh ist aktiv
  Fehlt der Doppelpunkt, dann ist die BankBoot der ersten Generation eingebaut und diese Anzeigen sind nur Software gesteuert und daher nur meistens richtig.
- die aktuelle Programmversion (wichtig für verschiedene Systeme in versch. Banken)
- Datum und Uhrzeit, falls die Baugruppe UHR3 eingebaut ist
- ein Stern an der Stelle vor dem Datum weist auf eine veränderte Prüfsumme des Systems hin, was z.B. durch einen Programmabsturz (in der RAM-Bank) verursacht sein kann. Dann sind die Systemfunktionen nicht mehr sicher lauffähig. Es empfiehlt sich in dem Fall die RAM-Bank mit dem Banklader neu zu laden.

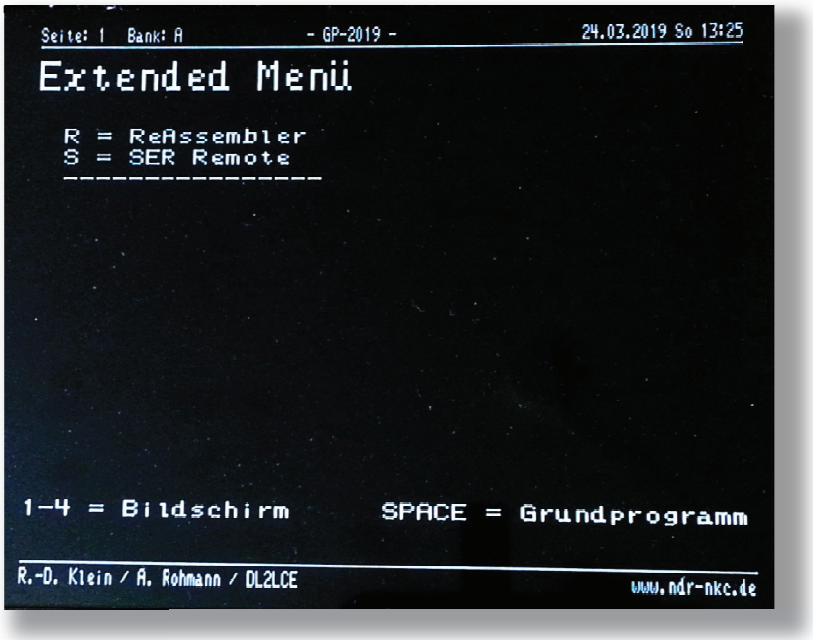
Grundprogramm - Monitor

Das folgende Bild zeigt das Menü des Monitor‘s, in dem die residenten Funktionen für die Benutzung des Computers zusammengefasst sind.

BASIC, FLOMON oder AC1-EDAS sind nicht Bestandteil dieser Auslieferung und werden nur dann als Menüpunkt angezeigt, wenn eines dieser Programme ab Adresse 4000h vorhanden ist.

Extended Menü

In diesem Menü werden Programme aufgelistet, die über den aktuellen 64 kB Adressraum entsprechend einer bestimmten Kennung gesucht werden. Damit ist es möglich z.B. in den Adressbereich 7000h nachgeladene Programme oder beliebige User-Programme per Menü zugänglich zu machen.



Die Funktion SER Remote ist nicht Bestandteil dieser Auslieferung, wird aber in Kürze als nachladbare Ergänzung angeboten werden. Der Aufruf erfolgt dann, wie hier zu sehen, über das Extended Menü.

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.

Dateinamen-Empfehlungen

Für das Speichern auf USB, wie auch auf die SD-Card muss im 8.3-Format erfolgen. D.h. ein Dateiname darf aus maximal 8 Zeichen bestehen, gefolgt von einem Punkt und drei weiteren Zeichen (Dateinamenerweiterung) zur Kennzeichnung der Art der Datei.

Im vorliegenden Grundprogramm werden folgende Dateinamen-Erweiterungen empfohlen:

Test.**bin**	Datei eines vollständigen Betriebssystems, welches mit den Bankladern in einer RAM-Bank gebootet werden kann.
Test.**400**	Eine Hunderter-Zahl kennzeichnet Dateien die auf eine bestimmte Adresse – (.400 = 4000h) geladen werden müssen
Test.**exe**	kennzeichnet eine Datei, die in den ersten beiden Bytes die individuelle Ladeadresse enthält (wird nach dem Laden mit 2x NOP ersetzt). Mit der Funktion „EXE starten“ wird diese Datei auf die vorbestimmte Adresse geladen und dort sofort gestartet.
Test.**eas**	ist für eine Assemblerlisting-Datei des AC1-EDAS vorbehalten. Diese Dateien müssen auf die jeweils verwendete Pufferadresse des EDAS geladen werden. In der Grundeinstellung ist das ab 8000h.
Test.**44f**	o.ä ist nicht festgelegt, kann aber individuell als Versionsnummer einer Entwicklungsserie verwendet werden. Hierbei ist die Ladeadresse der Datei nur aus dem Kontext zu erkennen.
Read.Me	ist eine Textdatei, die vom USB-Stick direkt gelesen und am Bildschirm angezeigt werden kann. Die Datei kann mit jedem einfachen Editor (auch PC) im txt-Format geschrieben werden. Diese kann 20 Zeilen mit je max. 42 Zeichen enthalten und muss mit $$$ enden.

L = laden SD

Diese Funktion ist nur noch innerhalb der Directory-Anzeige zugänglich. Damit können Daten und Programme über die Baugruppe CASneo von einer SD-Karte geladen werden. Ein Bedienteil ist nicht mehr notwendig, da die Daten wie auf einer Festplatte auf der SD-Karte gespeichert werden.

Nach dem Start werden 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.

Im Falle von Fehlern wird der Fehlercode des VDIP-Moduls am Bildschirm angezeigt.

S = speichern SD

Mit dieser Funktion können Daten und Programme aus dem Speicher des NKC auf eine SD-Karte der Baugruppe CASneo gespeichert werden.

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 die SD-Karte der CASneo dauert etwa eine Sekunde pro zu speicherndem acht Kilobyte. Ein begonnener Vorgang kann nicht abgebrochen werden.

Ist der gewählte Dateiname bereits auf der SD-Karte vorhanden, wird gefragt, ob die bestehende Datei vor dem Speichern gelöscht werden soll. Für „Ja“ wird nur das großgeschriebene „J“ akzeptiert. Jede andere Taste führt zum Abbruch der Funktion.

Im Falle von Fehlern wird der Fehlercode des VDIP-Moduls am Bildschirm angezeigt.

D = Directory SD-Card

Mit dieser Funktion kann der Inhalt der SD-Karte in der CASneo-Baugruppe angezeigt werden.

Die Anzeige der auf dem Datenträger vorhandenen Dateien ist einspaltig, bei mehr als 16 Einträgen kann mit der Eingabetast-Taste zur weiter geblättert werden.

L=Lade	Dient zum direkten Laden einer Datei
D=Lösche	Dient zum Löschen einer Datei vom Datenträger
T=Type	Dient zu Anzeigen einer kurzen ASCII-Datei
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

E = EXE starten SD

Mit dieser Funktion können Programme über die Baugruppe CASneo von einer SD-Karte auf die dem Programm mitgegebene Speicheradresse geladen und dort sofort gestartet werden. Damit kann man neue Programmroutinen quasi direkt aufrufen und ausführen.

Nach dem Start wird der Dateiname (8 Zeichen!) abgefragt, wobei die zwingende Dateinamenerweiterung „.EXE“ vorgegeben ist und nicht mit eingegeben werden darf.

Nach der Eingabe des Dateinamens wird die Datei unmittelbar an die vorgegeben Adresse geladen. Bei der Eingabe eines leeren Dateinamen wird die Funktion abgebrochen.

> Die Datei muss in ihren ersten beiden Bytes mit der Ladeadresse beginnen. Diese beiden Bytes werden nach dem Laden mit NOP (NoOPeration = Leerbefehl) überschrieben. Die Datei muss beim Speichern die Dateinamenserweiterung „.EXE“ erhalte.

W = Banklader SD

> Diese Programmroutine ist nur im EPROM/EEPROM der BankBoot lauffähig!
> In einer RAM-Bank wird diese Programmfunktion nicht angezeigt.

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 (booten). Das neue Betriebssystem muss bei der Adresse 0000h beginnen.

Zunächst wird die Banknummer (0...F) eingegeben. Anschließend muss der Dateiname (Format 8.3) eingegeben werden, unter dem das System im USB-Speicher abgelegt ist. Das zu ladende Programm wird immer in die ausgewählte RAM-Bank 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.

Nach Auswahl der Bank wird zunächst vom GP die neue Bank ab Adresse 8000h „hinter“ dem Speicher 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 des Betriebssystem auf der BankBoot ist nur mit RESET möglich.

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.

l = laden USB

Die Funktion ist nur noch innerhalb der Directory-Anzeige zugänglich und 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.

d = USB Directory

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
T=Type Dient zu Anzeigen einer kurzen ASCII-Datei
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.

? = Read.Me USB

Mit dieser Funktion wird auf dem USB-Stick eine Datei „Read.Me“ gesucht. Wenn diese gefunden wurde wird deren Inhalt nach 6800h geladen und sofort mittels der Funktion TYPE auf dem Bildschirm angezeigt.

Die Datei **Read.Me** darf neben ASCCII-Zeichen nur 0Dh und 0Ah enthalten und muss mit $$$ enden. Sie darf bis zu 20 Zeilen mit maximal je 42 Zeichen enthalten.

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 = Banklader USB

Diese Programmroutine ist nur im EPROM/EEPROM der BankBoot lauffähig!
In einer RAM-Bank wird diese Programmfunktion nicht angezeigt.

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 die BankBoot ist nur mit RESET möglich.

b = Bank einblenden

Nach Eingabe der gewünschten Banknummer 0...F wird im Hintergrund der BankBoot-Baugruppe die ausgewählte RAM-Bank im Bereich 8000h bis 0FFFFh eingeblendet.

> Diese Programmroutine ist nur im EPROM/EEPROM der BankBoot lauffähig!
> In einer RAM-Bank wird diese Programmfunktion nicht angezeigt.

B = Bank wechseln

Mit dieser Funktion kann ein anderes Betriebssystem „gebootet“ werden. Dieses muss bereits in einer anderen RAM-Bank lauffähig ab der Adresse 0000h gespeichert sein. Damit ist es nach entsprechender Vorbereitung möglich, in beliebige andere Betriebssysteme zu wechseln.

Nach Eingabe der gewünschten Banknummer 0...F wird auf die Adressen 0FFFEh und 0FFFFh 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.

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
N=Eingabe eines neuen Ausgabewertes
M=Menü Zurück zum Grundprogramm-Menü
1-4=Seite Umschaltung der Bildschirmseite

T = TYPE

Mit dieser Funktion kann eine Textdatei von einer bestimmten Adresse auf dem Bildschirm angezeigt werden. Die Datei darf neben ASCCII-Zeichen 0Dh und 0Ah enthalten und muss mit $$$ enden.

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

Verdeckter Befehl „A“ =aktivieren des Uhren-Chip’s:

Am Ende der Funktion wird angezeigt:
- Break = Uhrenchip war schon aktiviert
- OK = Uhrenchip ist jetzt aktiviert
- ERROR = Aktivierung fehlgeschlagen

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

C = BASIC *(falls vorhanden)*

Dieser Menüeintrag wird nur angezeigt, wenn das 8kB-BASIC-Programm - RDK'83 - ab der Speicheradresse 4000h erkannt wird.

So kann der BASIC Interpreter schnell aus dem Grundprogramm gestartet werden.

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.

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

C = CP/M *(falls FLOMON4000 vorhanden)*

Dieser Menüeintrag wird nur angezeigt, wenn das 8kB - FLOMON 4000 – von Jens M. ab der Speicheradresse 4000h erkannt wird.

So kann CP/M schnell aus dem Grundprogramm gestartet werden.

Für die Funktion ist es allerdings Notwendig, dass ein Speichermedium Floppy / CASneo mit entsprechendem Datenträger im System vorhanden ist. Weitere Informationen sind hier zu finden: https://ntxdhxgzadrdathx.myfritz.net/ndr/software/casneocpm/index.html
(Dort ist auch FLOMONCN4000 zu finden.)

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
```

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 (6000h…6800h) 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

h = Hexdump

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 werde, 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 = Speicher Abbild

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

v = vergleichen

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

u = Mustersuche

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.

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

r = 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

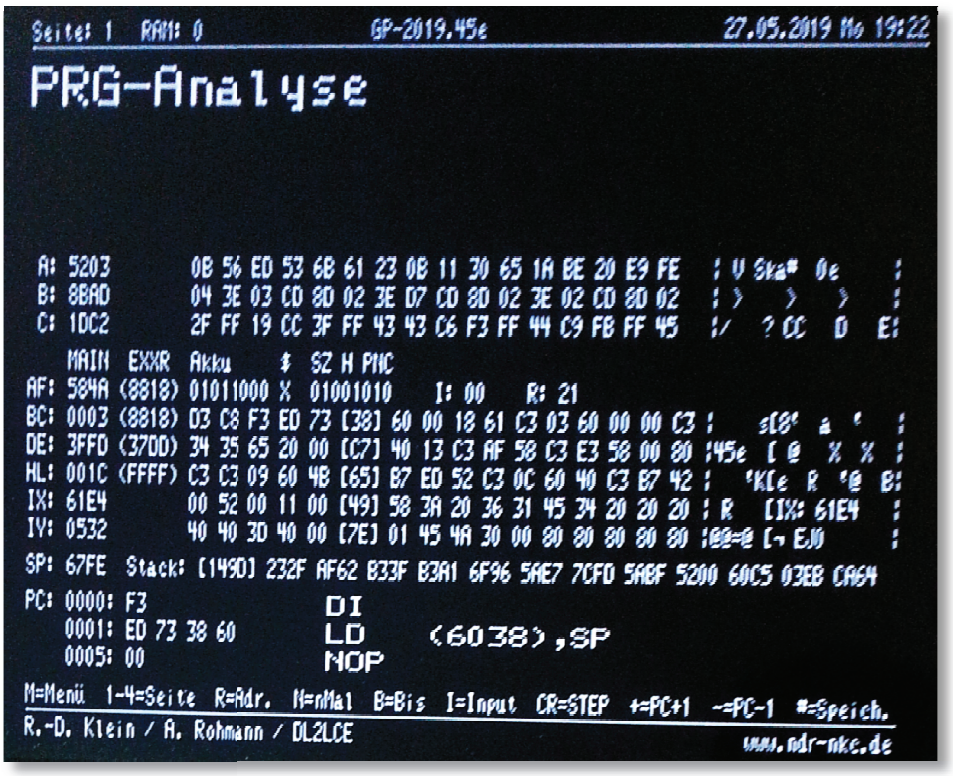
Summe A + B in hexadezimaler Schreibweise
Differenz A – B in hexadezimaler Schreibweise
Differenz B – A in hexadezimaler Schreibweise
Dez A Dezimalwert A und Zweierkomplement von A
Dez B Dezimalwert B und Zweierkomplement von B

Bedienmöglichkeiten innerhalb der Funktion:

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

x = Programmanalyse

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



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

Drei getrennte **Speicherbereiche (A, B, C)** von jeweils 16 Byte Umfang incl. deren ASCII-Interpretation die einzeln auf beliebige Adressen eingestellt werden können
Die **internen Register der Z80** incl. in Klammern deren Zweit-Register, gefolgt von 16 Byte zusammen mit deren ASCII-Interpretation. Dabei stellt der Wert in eckigen Klammern den

Inhalt der mit dem Doppelregister adressierten Zelle dar. Links davon 5 Zellen vor dieser Adresse, rechts davon die auf die Adresse folgenden Zellen. Damit ist z.B. der Erfolg von LD (HL),xxxx Befehlen leicht zu kontrollieren.

Der **Akkumulator und das Flag-Register** werden in Bitdarstellung gezeigt. Dabei stehen die entsprechenden Kürzel über den Bits-des Flag-Registers. Unter dem „$"-Zeichen wird der Inhalt des Akkus als ASCII-Interpretation gezeigt.

Der Inhalt des **Interrupt- und des Refresh-Register** werden in der gleichen Zeile, wie der Akku angezeigt. Dabei ist zu beachten, dass das Refersh-Register ein interner Zähler der CPU ist und keine sinnvollen Auswertungen im Steppbetrieb ermöglicht.

Der **Stackpointer** gefolgt von dem Inhalt des Stapelspeichers (Stack). Dabei werden die Byteeinträge jeweils paarweise getauscht, so dass in der Darstellung der abgelegte Doppel-Registerinhalt bzw. die gespeicherte Adresse direkt lesbar ist. In eckigen Klammern ist der „oberste" Stapelinhalt dargestellt; das ist die Adresse, die mit dem nächsten POP-Befehl vom Stapel genommen wird oder zuletzt mit PUSH in den Stapel gelegt wurde.

Der **Programm-Counter (PC)** gefolgt von der Darstellung der auf die aktuelle Befehlsadresse folgenden 3 Befehle in Byte und in Assemblerdarstellung. Die oberste Zeile ist der Befehl, der als nächstes ausgeführt werden wird.

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 *(Beschreibung weiter unten)*
CR=Step	Ausführen des aktuellen Befehls
+=PC+1	Programmzähler erhöhen (keine Ausführung)
-=PC-1	Programmzähler verringern (keine Ausführung
#=Speich	direkte Änderung von Speicherinhalten

Nicht sichtbare Menüpunkte

S=Skip	Verdecktes Ausführen eines Unterprogramms
Leertaste/Space =	der nächste Befehl wird ohne Ausführung übersprungen

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

y = 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.

ACHTUNG: der RAM-Test überschreibt die vorherigen Speicherinhalte!

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.

* = 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.

p = prog.EEPROM

Die Funktion dient zum direkten Programmieren (Beschreiben) von EEPROMs vom Typ 28C64 im Speicherbereich. Dafür kann der EEPROM auf der Speicherkarte verbleiben.

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 beim Programmieren die Adresse der soeben beschriebenen Adresse im Ziel-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.

> Zum Test kann mit der Funktion auch ein Bereich im RAM „programmiert“ werden. Dabei werden die Prüfsummen in Quelle und Ziel abweichen, da die Ziel-Zellen x+0AAAh = 55h und x+1555h = A0h aufweisen. Das ist normal und eben nur für den EEPROM fehlerfrei!
>
> HINWEIS:
> Auf den Adressen 0000..3FFFh kann kein EEPROM programmiert werden, da dort die Programmroutinen im Grundprogramm liegen. Um dennoch, ohne die EEPROMS des Grundprogrammes (u.U. öfters) aus den IC-Fassungen reißen zu müssen, wurde eine nachladbare Funktion **ICP.F00** *„System EEPROM programmieren“* erstellt. Diese muss in den RAM ab Adresse F000h geladen werden und steht danach im Extended Menü zur Verfügung. Unbedingt daran denken, dass die ersten 16 kB (2x EEPROM) zueinander passen müssen und daher direkt nacheinander ohne die Funktion zu verlassen programmiert werden müssen!

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 = Menüwechsel

Wechselt vom aktiven Grundprogramm-Monitor–Menü zum Extended Menü.

Funktionen des Extended Menüs

Für dieses Menü sucht ein Programm in Adressbereich der aktuellen RAM-Bank nach einer signifikanten Kennung, die am Anfang eines Programmes stehen muss. Auf der Menüseite werden alle so gefundenen Programme aufgelistet und können wie bei den vorherigen Menüs durch Tastendruck ihres zugehörigen Kennbuchstabens/Kennzeichens gestartet werden.

Programmkennung

Damit das jeweilige Programm im Extended Menü eingetragen wird, muss ihm eine Kennung vorangestellt werden. Dafür bestehen drei Varianten:

1. Ein Programm, das mit einer Titelüberschrift startet:

```
.DEFB    0EDh, 0FFh        ;Kennbyte 1 und 2
.DEFB    'T'               ;Kennbuchstabe „T" zum Starten
.DEFB    00Dh              ;Kennbyte 4

RST      30h               ;Löscht BS und Zeigt Prog.Titel an
.DEFB    'Testprogramm'    ;Menüeintrag und Programmüberschrift
.DEFB    0                 ;Textende Kennzeichen
. . . . . .                ;hier beginnt das eigentliche Programm
RET                        ;Programmende
```

2. Ein Programm ohne Titelüberschrift

```
.DEFB    0EDh, 0FFh        ;Kennbyte 1 und 2
.DEFB    'X'               ;Kennbuchstabe „X" zum Starten
.DEFB    00Dh              ;Kennbyte 4

NOP                        ;Leerbefehl bzw. 00-Byte
.DEFB    'Testprogramm'    ;Menüeintrag
.DEFB    0                 ;Textende Kennzeichen
. . . . . .                ;hier beginnt das eigentliche Programm
RET                        ;Porgrammende
```

3. Ein Menüeintrag ohne, dass ein Programmstart hinterlegt ist

```
.DEFB    0EDh, 0FFh        ;Kennbyte 1 und 2
.DEFB    ' '               ;Leerzeichen/Space-Taste
.DEFB    00Dh              ;Kennbyte 4

NOP                        ;Leerbefehl bzw. 00-Byte
.DEFB    '--------------'  ;Menüeintrag, hier Trennlinie
.DEFB    0                 ;Textende Kennzeichen
```

   Ein Programmende ist nicht nötig, da die Leertaste als Programmauswahl abgefangen wird und zum Menüwechsel dient.

Die Bezeichnung wurde bereits in der Version für den AC1 so gewählt. In der Bedeutung scheint „Re= zurück“ (Duden: „in den Ausgangszustand zurückgeführt“) sogar passender zu sein, als das häufig gebrauchte DisAssembler; „Dis=auseinander“ (lat. „entzwei“). ☺

Der ReAssembler dient zum Rückübersetzen eines Programmcodes in Assembler-Befehle. Dabei wird aus dem Maschinensprache-Code eine Textdatei erstellt, und auf dem Bildschirm angezeigt. Es werden allerdings vom ReAssembler aufgrund seiner geringen Größe keine Labels erzeugt; Sprungziele sind immer die aktuellen Adressen. Soll der erzeugte Assembler-Code jedoch nicht nur analysiert, sondern auch verändert werden, ist eine manuelle Nacharbeitung - *Vergabe von Labels* – in einem Assemblerprogramm unumgänglich.

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.

Beachte: 0100h Binär-Code belegen schon mal 4-6 kB Textspeicher!

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 = Menüwechsel

Wechselt vom aktiven Extended–Menü zum Grundprogramm–Menü.

Druckerfunktionen

Das vorliegende Betriebssystem verfügt über eine einfache Druckerunterstützung über die CEN (Centronics auf IOE) – Baugruppe. Damit stehen Unterprogramme für die Druckerausgabe zur Verfügung.

Mit der Tastenkombination [Strg] + [P] wird der Sofortdruck ein/aus geschaltet. Dabei wird jedes ASCII-Zeichen, das auf dem Bildschirm dargestellt wird, parallel zum Drucker gesendet. Das betrifft auch die Kopf- und Fußzeile des Menübildes. Es können auch keinerlei Formatierungen auf dem Drucker realisiert werden, so dass der Ausdruck nicht immer dem Bildschirmbild entspricht.

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 Einsprüngen 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 verwendeten RST-Aufruffunktionen

Bei den RST-Befehlen handelt es sich um Ein-Byte-Befehle, die sich wie ein normaler Unterprogrammaufruf – Call – verhalten und aufgrund der Einsparung von jeweils 2 Bytes für häufig verwendet Unterprogrammaufrufe verwendet werden.

```
RST 00h      ;=Call 0000h, Verwendung wie Software-RESET
RST 08h      ;=Call 0008h, call UPV – Unterprogrammverteiler
RST 10h      ;=Call 0010h, call zeich – ein ASCII-Zeichen in Puffer
RST 18h      ;=Call 0018h, call textout – Zeichenkette in Puffer
RST 20h      ;=Call 0020h, call prtac – Accu als Hexzahl in Puffer
RST 28h      ;=Call 0028h, call prthl – HL-Reg. als Hexzahl in Buffer
RST 30h      ;=Call 0030h, call TITLout – generiert ein Programmtitel
RST 38h      ;=Call 0038h, ist der IM0 Interrupt und bleibt frei
```

Auf den Adressen der RST-Befehle 0008h, 0020h etc. stehen jeweils Sprünge in den System-RAM-bereich. Erst dort ist der zuvor beschriebene Programmaufruf hinterlegt. Damit ist es möglich die vorgenannten Unterprogramme „umzuleiten“ oder von USER-Programmen anderweitig zu verwenden. Die ursprünglichen RST-Sprung-Vectoren können/müssen anschließend durch den Aufruf des Unterprogrammes „SetRST“ (UP-Nr. 05Ah) zurückgesetzt werden!

RST 08h = JP 6003h
RST 10h = JP 6006h
RST 18h = JP 6009h
RST 20h = JP 600Ch
RST 28h = JP 600Fh
RST 30h = JP 6012h
RST 38h = JP 6015h
NMI = JP 6018h

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.

Der Programmaufruf würde dann z.B. so aussehen:

```
Start:      RST 08h          ;Call Unterprogrammverteiler
            .defb prthl      ;HL HEX in Buffer ab IX schreiben
            . . .
```

Denkbar wäre auch eine Version in der die Nummer des Unterprogrammes zusammen mit dem RST 08h-Aufruf als Word gespeichert wird:

```
prthl:      .equ  013CFh     ;UP-Nr.-prthl + Z80-Code für RST 08h
                             ;…wird in der Reihenfolge CF, 13 abgelgt
Start:      .defw prthl      ;ruft das UP prthl auf
            . . .
```

Für diese Variante ist am Ende dieses Scriptes eine Liste zum Einfügen in ein eigenes Assemblerlisting abgedruckt.

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

```
;----- GP- Unterprogramme -----
upv:        .equ 0CFh   ;Code für den Befehl RST 08h (UP-Verteiler)
```

```
;------KEYBOARD
csts:         .equ 000h   ;Status KEY lesen
ci:           .equ 001h   ;Zeichen von KEY einlesen nach ACCU
ki:           .equ 002h   ;wie CI mit Wandlung nach Großbuchstaben
;
;------GDP:
wait:         .equ 003h   ;Warten bis GDP bereit
waitsync:     .equ 004h   ;Warten auf Vertical Blank
cmd:          .equ 005h   ;Kommandoausgabe an GDP, ADDU=Kommando-Byte
fast:         .equ 006h   ;Schneller unsynchronisierter Schreibmodus
slow:         .equ 007h   ;Normaler synchronisierter Schreibmodus
;
setpen:       .equ 008h   ;GDP Schreibmodus setzen
erapen:       .equ 009h   ;GDP Löschmodus setzen
setviewpage:  .equ 00Ah   ;Anzeigeseite setzen (ohne Aktivierung)
setwrtpage:   .equ 00Bh   ;Schreibseite setzen (ohne Aktivierung)
aktpage:      .equ 00Ch   ;Seite gemäß viewpage und aktpage aktivieren
setaktpage:   .equ 00Dh   ;Seite setzen und aktivieren
clrall:       .equ 00Eh   ;alle Bildschirmseiten löschen
clrakt:       .equ 00Fh   ;aktuelle Bildschirmseite löschen
clrinvis:     .equ 010h   ;Seite loeschen auch unsichtbar
kill:         .equ 011h   ;GP Funktion auf Seite zurücksetzen
;
;------ AUSGABETEXT über Buffer
      ;-- 1.Schritt
getausbuf:    .equ 012h   ;Textbuffer initialisieren -> IX = Pointer
      ;-- 2.Schritte (mögl.)
prthl:        .equ 013h   ;HL HEX in Buffer ab IX schreiben
prthld:       .equ 014h   ;HL DEZ in Buffer ab IX schreiben
prtac:        .equ 015h   ;ACCU HEX in Buffer ab IX schreiben
prtacd:       .equ 016h   ;ACCU DEZ in Buffer ab IX schreiben
prtbin:       .equ 017h   ;ACCU BIN in Buffer ab IX schreiben
print:        .equ 018h   ;Text ab (HL) in Buffer ab IX schreiben
printin:      .equ 019h   ;n. Call folgende Bytes,0 in Buffer
zeich:        .equ 01Ah   ;Zeichen in ACCU in Buffer ab IX schreiben
newline:      .equ 01Bh   ;0dh, 0ah in Buffer schreiben IX
      ;-- 3.Schritt:
printbuf:     .equ 01Ch   ;ld HL=X-Pos, DE=Y-Pos, A=Schrift in Buffer und
                          ;gibt Buffer auf BS aus (incl. 0ah etc)

;------- AUSGABETEXT ohne Buffer
textaus:      .equ 01Dh   ;Text a.BS von HL
textmulti:    .equ 01Eh   ;Mehrere Texte ausgeben wie vor
txtout:       .equ 01Fh   ;Text a.BS ausgeben Parameter+Text nach Call
TXTCSR:       .equ 020h   ;Text a.BS auf CSR-Position weiter (noch
nicht!)
;
;------ EINGABETEXT
textein:      .equ 021h   ;hl->Versorgungsblock c=flag 0,1
textxy:       .equ 022h   ;Texteingabefeld
gethl:        .equ 023h   ;Eingabe als Zahl interpretieren -> HL
gettext:      .equ 024h   ;IX
expr:         .equ 025h   ;Auswertung eines Ausdrucks (rechnet)
getpara:      .equ 026h   ;Parametereingabe mit Bezeichnung (HL)
;
;------ AUSGABEGRAFIK
```

```
moveto:      .equ 027h   ;Grafik Cursor setzen
drawto:      .equ 028h   ;Linie zeichnen
tmove:       .equ 029h   ;Turtle Position setzen
tschreite:   .equ 02Ah   ;Turtle vorwärts bewegen (Symbol SCHREITE)
tschr16tel:  .equ 02Bh   ;Turtle langsam vorwärts bewegen
                         ;(Symbol SCHR16TEL)
tdrehe:      .equ 02Ch   ;Turtle rotieren
thoch:       .equ 02Dh   ;Turtle anheben (unsichtbar bewegen)
trunter:     .equ 02Eh   ;Turtle absenken (sichtbar schreiben)
turtle:      .equ 02Fh   ;Turtle zeichnen (setpen/erapen)
sprite:      .equ 030h   ;Sprite ausgeben (IX=Definition, B=Size)
;
;------ ALLGEMEINES
upper:       .equ 031h   ;Char in ACCU in Großbuchstaben wandeln
waitms:      .equ 032h   ;Wartezeit in Millisekunden
wait100ms:   .equ 033h   ;100 Millisekunden warten
adj360:      .equ 034h   ;HL
sin:         .equ 035h   ;HL
cos:         .equ 036h   ;HL
cplhl:       .equ 037h   ;HL
hexdez:      .equ 038h   ;Umwandlung HL(hex) -> CDE(dez)
dezhex:      .equ 039h   ;Umwandlung CDE(dez) -> HL(hex)
length:      .equ 03Ah   ;Länge eines Z80 Befehls ermitteln
;
;------ UHR
getDate:     .equ 03Bh   ;E=Tag, D=Monat, HL=Jahr
getTime:     .equ 03Ch   ;C=Stunde, D=Minute, E=Sekunde,
                         ;B=Wochentag (1..7)
;
;-------USB
usb_start:   .equ 03Dh   ;USB Initialisieren
usb_isdisk:  .equ 03Eh   ;Laufwerk prüfen
usb_dir:     .equ 03Fh   ;Test ob Datei vorhanden
usb_load:    .equ 040h   ;Datei komplett laden
usb_save:    .equ 041h   ;Datei komplett schreiben
usb_opw:     .equ 042h   ;Datei zum Schreiben öffnen
usb_wr:      .equ 043h   ;Bytes in Datei schreiben
usb_opr:     .equ 044h   ;Datei zum Lesen öffnen
usb_rd:      .equ 045h   ;Bytes aus Datei lesen
usb_clf:     .equ 046h   ;Datei schließen
usb_sek:     .equ 047h   ;Pointer in Datei setzen (DE=Position)
usb_dlf:     .equ 048h   ;Delete File
usb_ren:     .equ 049h   ;Rename File
vnc_read:    .equ 04Ah   ;Byte aus geöffneter Datei lesen
vnc_write:   .equ 04Bh   ;Byte in geöffnete Datei schreiben
vnc_aus:     .equ 04Ch   ;Datenträger auswerfen
;
;--- Drucker
LOINIT:      .equ 04Dh   ;Drucker CER Init
LO:          .equ 04Eh   ;Druckerausgabe Accu
;
;--- Monitor/Menübedienung
TITLout:     .equ 04Fh   ;Titelanzeige
clrcenter:   .equ 050h   ;Mittelbereich löschen
getadr:      .equ 051h   ;eine Adresse abfragen
getvb:       .equ 052h   ;Adressen von-bis abfragen
```

```
getvbn:      .equ 053h   ;Adressen von-bis-nach abfragen
FMENU:       .equ 054h   ;Fussmenue
endERR:      .equ 055h   ;Programmende mit ERROR
endOK:       .equ 056h   ;Programmende mit OK
endBRK:      .equ 057h   ;Programmende mit BREAK
;
NewMin:      .equ 058h   ;Uhr aktualisieren
crc:         .equ 059h   ;Prüfsumme berechnen  Länge, HL=crc
setRST:      .equ 05Ah   ;Setzt RST-Sprung-Vectoren zurück
```

Beschreibung der Systemfunktionen

Es folgt eine ausführliche Beschreibung der einzelnen Funktionen, die über die Sprungverteiler angesprochen werden können. Dies erfolgt durch Aufruf des Unterprogrammverteilers, gefolgt von der Nummer des aufzurufenden Unterprogrammes.

Zum Beispiel:

```
;sendet ein Fragezeichen an den Drucker
LD    A,"?"       ;Fragezeichen
CALL 000Dh        ;Unterprogrammverteiler
.DEFB 04Eh        ;LO = Druckerausgabe
```

mit der vorigen UP-Bibliothek

```
;sendet ein Fragezeichen an den Drucker
LD    A,"?"       ;Fragezeichen
CALL 000Dh        ;Unterprogrammverteiler
.DEFB LO          ;LO = Druckerausgabe
```

oder als RST-Befehl

Der Unterprogrammverteiler lässt sich auch über den RST-Befehl: RST 30h aufrufen.

```
;sendet ein Fragezeichen an den Drucker
LD    A,"?"       ;Fragezeichen
RST   08h         ;Unterprogrammverteiler
.DEFB LO          ;LO = Druckerausgabe
```

oder in Kurzform

```
;sendet ein Fragezeichen an den Drucker
LD    A,"?"       ;Fragezeichen
.DEFB UPV, LO     ;= RST 08h und UP.Nr. für UP- „LO"
```

bzw. besser

```
LO:   .equ 4ECFh  ;UP-Nr. + RST 08h-Befehl
;
LD    A,"?"       ;Fragezeichen
.DEFW LO          ; = 0CFh, 4Eh im Speicher
```

Siehe unter „**Unterprogramm Tabelle zum Einfügen**“ am Ende des Scriptes.

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.

UP.NR.: 00H **CSTS** ABFRAGE DES TASTATURSTATUS

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CSTS              ; Aufruf CSTS
            JR Z,WAITKEY            ; Warten, wenn keine Taste gedrückt
            RET
```

UP.NR.: 01H **CI** EINLESEN EINES ZEICHENS VON DER TASTATUR

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Zeichen einlesen
            CP 0DH                  ; Taste CR gedrückt?
            JR NZ,KEYLOOP           ; wiederholen bis CR gedrückt
            RET
```

UP.NR.: 02H **KI** EINLESEN EINES ZEICHENS VON DER TASTATUR

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB KI                ; Zeichen einlesen
            CP 'M'                  ; Taste M oder m gedrückt?
            JR NZ,KEYLOOP           ; wiederholen bis M oder m gedrückt
            RET
```

UP.NR.: 03H **WAIT** WARTEN AUF BEREITSCHAFT DER GDP BAUGRUPPE

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB WAIT              ; Warten auf Bereitschaft
            LD A,41H                ; Buchstabe A
            OUT (70H),A             ; Kommando an GDP senden
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 04H **WAITSYNC** WARTEN AUF SYNCHRONISATION

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB WAITSYNC          ; Warten auf Bildschirmrücklauf
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CMD               ; Buchstabe ausgeben
            INC A                   ; nächster Buchstabe
            CP 5BH                  ; Vergleich auf Buchstabe Z
            JR NZ,LOOP              ; bis alles ausgegeben ist
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 05H **CMD** AUSGABE EINES KOMMANDOS AN GDP

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CMD               ; Kommando ausführen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 06H **FAST** SCHNELLE BILDSCHIRMAUSGABE

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

UP.NR.: 07H **SLOW** NORMALE BILDSCHIRMAUSGABE

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

UP.NR.: 08H **SETPEN** SCHREIBMODUS AKTIVIEREN

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

Parameter	keine
Rückgabe	keine
Register	ACCU

```
START:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CLRAKT            ; Bildschirm löschen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB SETPEN            ; Schreibnodus aktivieren
            LD A,41H                ; Buchstabe A
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CMD               ; ausgeben
            LD HL,1000              ; 1000 Millisekunden
```

```
RST 08h                 ; Unterprogrammverteiler
.DEFB WAITMS            ; warten
RST 08h                 ; Unterprogrammverteiler
.DEFB ERAPEN            ; Löschmodus aktivieren
LD A,41H                ; Buchstabe A
RST 08h                 ; Unterprogrammverteiler
.DEFB CMD               ; Löschend überschreiben
RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 09H **ERAPEN** LÖSCHMODUS AKTIVIEREN

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

Parameter	keine
Rückgabe	keine
Register	ACCU

UP.NR.: 0AH **SETVIEWPAGE** BESTIMMT DIE ANZUZEIGENDE BILDSCHIRMSEITE

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	

UP.NR.: 0BH **SETWRTPAGE** BESTIMMT DIE ZU BESCHREIBENDE BS-SEITE

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	

UP.NR.: 0CH **AKTPAGE** AKTIVIEREN VON ANZEIGE- UND SCHREIBSEITE

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

UP.NR.: 0DH **SETAKTPAGE** SETZEN UND AKTIVIEREN DER BILDSCHIRMSEITE

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,3                  ; Bildschirmseite 4
LOOP:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB WAITSYNC          ; Warten auf Synchronisation
            LD A,B                  ; Bildschirmseite holen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB SETAKTPAGE        ; Bildschirmseite aktivieren
            INC B                   ; nächste Seite
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; warten bis Taste gedrückt?
            DJNZ LOOP               ; wiederholen bis BS-Seite 1
            RET
```

UP.NR.: 0EH **CLRALL** LÖSCHT DEN INHALT ALLER BILDSCHIRMSEITEN

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.

UP.NR.: 0FH **CLRAKT** LÖSCHT DEN INHALT DER AKT. SCHREIBSEITE

Die Funktion löscht den Inhalt der aktuellen Bildschirmseite.

Parameter keine
Rückgabe keine
Register ACCU

```
START:
            LD A,0                  ; Seite 0
            RST 08h                 ; Unterprogrammverteiler
            .DEFB SETVIEWPAGE       ; Anzeigeseite setzen
            LD A,1                  ; Seite 1
            RST 08h                 ; Unterprogrammverteiler
            .DEFB SETWRTPAGE        ; Schreibseite setzen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB AKTPAGE           ; Seiten aktivieren
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CLRAKT            ; Inhalt Seite löschen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 10H **CLRINVIS** LÖSCHT EINE UNSICHTBARE BILDSCHIRMSEITE

Die Funktion dient zum Löschen einer nicht sichtbaren Bildschirmseite, da das Kommando der GDP nur auf der aktuell sichtbaren Seine 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

UP.NR.: 11H **KILL** BEENDET EINE LAUFENDE FUNKTION DES GRUNDPROGRAMMS

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

UP.NR.: 12H **GETAUSBUF** GIBT ZEIGER A.D.N AUSGABEPUFFER ZURÜCK

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB GETAUSBUF         ; IX = Pufferadresse
            LD HL,1234H             ; Wert nach HL
            .........
```

Das Beispiel wird bei PRTHL fortgesetzt

UP.NR.: 13H **PRTHL** AUSGABE EINER 16 BIT HEXADEZIMALZAHL I.D.PUFFER

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

```
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRTHL             ; HL in Puffer schreiben
            LD HL,100               ; X-Position
            LD DE,200               ; Y-Position
            LD A,21H                ; Schriftgröße
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRINTBUF          ; Puffer ausgeben
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 14H **PRTHLD** AUSGABE EINER 16 BIT DEZIMALZAHL I.D.PUFFER

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	

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 15H **PRTAC** AUSGABE EINER 8 BIT HEXADEZIMALZAHL I.D.PUFFER

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	

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 16H **PRTACD** AUSGABE EINER 8 BIT DEZIMALZAHL I.D.PUFFER

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	

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 17H **PRTBIN** AUSGABE EINER 8 BIT ZAHL I.D.PUFFER

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	

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 18H **PRINT** AUSGABE EINES TEXTES (HL-ADRESSIERET) I.D.PUFFER

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB GETAUSBUF         ; IX = Pufferadresse

            LD HL,TEXT              ; Adresse der Textdefinition
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRINT             ; Text in Puffer übertragen

            LD HL,100               ; X-Position
            LD DE,200               ; Y-Position
            LD A,21H                ; Schriftgröße
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRINTBUF          ; Puffer ausgeben

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 19H **PRINTIN** AUSGABE EINES FOLGENDEN TEXTES I.D.PUFFER

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:

            RST 08h                 ; Unterprogrammverteiler
            .DEFB GETAUSBUF         ; IX = Pufferadresse

            RST 08h                 ; Unterprogrammverteiler
            .DEFB 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
RST 08h                 ; Unterprogrammverteiler
.DEFB PRINTBUF          ; Puffer ausgeben

RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 1AH **ZEICH** AUSGABE EINES ASCII ZEICHENS I.D.PUFFER

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	

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 1BH **NEWLINE** NEUE ZEILE IM AUSGABEPUFFER I.D.PUFFER

Schreibt einen Zeilenwechsel (Carriage Return und Line Feed) in den Ausgabepuffer. Der Puffer wird mit der Kennung 00H abgeschlossen. Die vorherige, an die GDP übermittelte Y-Start-Position wird beibehalten, d.h. es ist möglich in einem Ausgabeblock den Text eingerückt darzustellen.

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

Um eine Bildschirmausgabe zu ermöglichen muss der Ausgabepuffer zuvor mit GETAUSBUF (Up.:Nr.: 12h) initialisiert worden sein. Nachdem der Ausgabepuffer mit den gewünschten Ausgaben gefüllt ist, erfolgt die Ausgabe mit der Funktion PRINTBUF (UP.Nr.: 01Ch).

UP.NR.: 1CH **PRINTBUF** AUSGABE DES PUFFERS AUF DEM BILDSCHIRM

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 Ausgebezeile.

Parameter	HL	X-Position
	DE	Y-Position
	ACCU	Schriftgröße
Rückgabe	keine	
Register	ACCU, HL, DE	

```
TEXT:
            .DEFB "HL: ",0          ; Textdefinition
START:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB GETAUSBUF         ; Adresse Puffer nach IX
            LD HL,TEXT              ; Startadresse des Textes
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRINT             ; Text in Puffer ablegen
            LD HL,1234H             ; HL belegen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRTHL             ; Wert in Puffer ablegen
            LD HL,100               ; X-Position
            LD DE,200               ; Y-Position
            LD A,21H                ; Schriftgröße
            RST 08h                 ; Unterprogrammverteiler
            .DEFB PRINTBUF          ; Puffer ausgeben

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 1DH **TEXTAUS** DIREKTE TEXTAUSGABE AUF DEM BILDSCHIRM

Schreibt einen Text unmittelbar ohne Verwendung des Ausgabepuffers auf den Bildschirm. Die Funktion wird direkt 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:
            .DEFW 10                ; X-Position
            .DEFW 200               ; Y-Position
            .DEFB 21H,0             ; Schriftgröße und Ausrichtung
            .DEFB "NDR-NKC"         ; Text
            .DEFB 0                 ; Ende-Kennung
START:
```

```
LD HL, TEXT             ; Versorgungsblock laden
RST 08h                 ; Unterprogrammverteiler
.DEFB TEXTAUS           ; Ausgabe am Bildschirm

RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 1EH **TEXTMULTI** AUSGABE MEHRERER TEXTE

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

UP.NR.: 1FH **TEXTOUT** DIREKTE TEXTAUSGABE AUF DEM BILDSCHIRM

Schreibt einen Text unmittelbar ohne Verwendung des Ausgabepuffers auf den Bildschirm. Der Versorgungsblock der die Position des Textes und die Schriftgröße enthält und der Text selbst folgen direkt nach dem Aufruf der Funktion. Die mitgegebene Y-Position wird bei einem Zeilenwechsel (0Ah) beibehalten, damit kann ein mehrzeiliger Text auch eingerückt ausgegeben werden.

Zur vereinfachten Handhabung wurde diese Funktion im Grundprogramm auf den RST-Befehl RST 18h gelegt.

Parameter	keine
Rückgabe	keine
Register	keine

```
START:
            RST 18h             ; TEXTOUT
            .DEFW 10            ; X-Position
            .DEFW 200           ; Y-Position
            .DEFB 21h           ; Schriftgröße
            .DEFB 00h           ; Schriftart, 04h kursiv
            .DEFB "NDR-NKC"     ; Text
```

```
.DEFB 0                 ; Ende-Kennung

RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 20H - RESERVIERT -

Diese Unterprogrammnummer ist für einer Weiterentwicklung reserviert und derzeit ohne Funktion.

UP.NR.: 21H **TEXTEIN**

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 eingebbarer Zeichen
BYTE	reserviert für tatsächliche Anzahl eingegebener Zeichen
TEXT	Puffer für einzugebende Zeichen

UP.NR.: 22H **TEXTXY** ABFRAGE VON BENUTZEREINGABEN

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:
            ;--- Bestimmung der Eingabeaufforderung
```

```
LD HL,100               ; X-Position
LD DE,200               ; Y-Position
LD A,21H                ; Schriftgröße
LD IX,TEXT              ; Startadresse Text
RST 08h                 ; Unterprogrammverteiler
.DEFB TEXTPRINT         ; ausführen

;--- Bestimmung des Eingabefeldes
LD HL,200               ; X-Position
LD DE,200               ; Y-Position
LD A,21H                ; Schriftgröße
LD B,15                 ; Feldbreite
RST 08h                 ; Unterprogrammverteiler
.DEFB TEXTXY            ; ausführen

RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 23H **GETHL** EINGABE ALS ZAHL INTERPRETIEREN

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

```
RST 08h                 ; Unterprogrammverteiler
.DEFB GETHL             ; Eingabe interpretieren
.DEFB UVP, GETAUSBUF    ; Pufferadresse holen
RST 08h                 ; Unterprogrammverteiler
.DEFB PRTHL             ; in Ausgabepuffer

LD HL,200               ; X-Position
LD DE,180               ; Y-Position
LD A,21H                ; Schriftgröße
RST 08h                 ; Unterprogrammverteiler
.DEFB TEXTAUS           ; Ergebnis ausgeben

RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 24H **GETTEXT** RÜCKGABE DES EINGEGEBENEN TEXTES

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	

UP.NR.: 25H **EXPR** AUSWERTUNG VON AUSDRÜCKEN

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

UP.NR.: 26H **GETPARA** PARAMETEREINGABE

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

UP.NR.: 27H **MOVETO** SETZT DIE KOORDINATEN FÜR GRAFIKAUSGABE

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB MOVETO            ; Grafikcursor setzen
            ...........
```

UP.NR.: 28H **DRAWTO** ZEICHNEN EINER LINIE ZUR ANGEGEBENEN POSITION

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB DRAWTO            ; Linie zeichnen

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Werten auf Tastendruck
            RET
```

UP.NR.: 29H **TMOVE** SETZT DIE POSITION DER SCHILDKRÖTENGRAFIK

Die Schildkrötengrafik (Turtlegrafik) bietet einfache Funktionen zum Zeichen 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 … 511)
	BC	Winkel im Uhrzeigersinn, 0=rechts, 90=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 verwendet werden.

```
START:
            LD HL,256               ; X-Position
            LD DE,128               ; Y-Position
            RST 08h                 ; Unterprogrammverteiler
            .DEFB TMOVE             ; Position setzen
            . . . . . .
```

Das Beispiel wird bei TSCHREITE fortgesetzt

UP.NR.: 2AH **TSCHREITE** BEWEGT DIE SCHILDKRÖTE VORWÄRTS

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB TSCHREITE         ; Linie nach oben zeichnen

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 2BH **SCHR16TEL** BEWEHT DIE SCHILDKRÖTE VORWÄRTS

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB TMOVE             ; Anfangsposition setzen
            LD B,72                 ; 72 Liniensegmente
LOOP:
            LD HL,8                 ; 8/16 Pixel
            RST 08h                 ; Unterprogrammverteiler
            .DEFB SCHR16TEL         ; kurze Linie zeichnen
```

Das Beispiel wird bei DREHE fortgeführt

UP.NR.: 2CH **TDREHE** DREHT DIE SCHILDKRÖTE

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
        RST 08h                 ; Unterprogrammverteiler
        .DEFB TDREHE            ; Turtle drehen
        DJNZ LOOP               ; bis Kreis abgeschlossen
        RST 08h                 ; Unterprogrammverteiler
        .DEFB CI                ; Warten auf Tastendruck
        RET
```

UP.NR.: 2DH **THOCH** HEBT DIE SCHILDKRÖTE AN

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

UP.NR.: 2EH **TRUNTER** SENKT DIE SCHILDKRÖTE AB

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
        RST 08h                 ; Unterprogrammverteiler
        .DEFB MOVE              ; Anfangsposition setzen
        LD B,36                 ; 36 Liniensegmente

LOOP:
        LD HL,4                 ; 4 Pixel
```

```
RST 08h                 ; Unterprogrammverteiler
.DEFB SCHREITE          ; Linie zeichnen
LD HL,5                 ; 5 Grad
RST 08h                 ; Unterprogrammverteiler
.DEFB DREHE             ; Turtle drehen
RST 08h                 ; Unterprogrammverteiler
.DEFB HEBE              ; Turtle anheben
LD HL,4                 ; 4 Pixel
RST 08h                 ; Unterprogrammverteiler
.DEFB SCHREITE          ; Linie zeichnen
LD HL,5                 ; 5 Grad
RST 08h                 ; Unterprogrammverteiler
.DEFB DREHE             ; Turtle drehen
RST 08h                 ; Unterprogrammverteiler
.DEFB SENKE             ; Turtle absenken
DJNZ LOOP               ; bis Kreis abgeschlossen
.........
```

Das Beispiel wird bei TURTLE fortgeführt

UP.NR.: 2FH **TURTLE** STELLT DIE SCHILDKRÖTE A.D. AKT. POSITION DAR

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

```
RST 08h                 ; Unterprogrammverteiler
.DEFB TURTLE            ; Turtle anzeigen
RST 08h                 ; Unterprogrammverteiler
.DEFB CI                ; Warten auf Tastendruck
RET
```

UP.NR.: 30H **FIGUR** ZEICHNET EINE VEKTORGRAFIK AUF DEN BILDSCHIRM

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
6	Linie nach links unten	SW
7	Linie nach links	W
8	Linie nach links oben	NW
9	??	
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                 ; Endeekennung
START:
            LD IX,FIG               ; Figurdefinition
            LD B,3                  ; Größe
            RST 08h                 ; Unterprogrammverteiler
            .DEFB FIGUR             ; Figur zeichnen

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

UP.NR.: 31H **UPPER** ZEICHEN IN GROSSBUCHSTABEN WANDELN

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Zeichen von Tastatur holen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB UPPER             ; Zeichen in Großbuchstaben wandeln
            CP 0Dh                  ; Eingabetaste?
            RET Z                   ; Ende

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CMD               ; Zeichen ausgeben
            JR START                ; neu starten
```

UP.NR.: 32H **WAITMS** VARIABLE WARTEZEIT IN MILLISEKUNDEN

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	

UP.NR.: 33H **WAIT100MS** WARTET 100 MILLISEKUNDEN

Unterbricht die Programmausführung für 100 Millisekunden bei 4 MHz Taktfrequenz.

Parameter	keine
Rückgabe	keine
Register	ACCU

UP.NR.: 34H **ADJ360** BEREICHSANPASSUNG FÜR WINKEL

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	

UP.NR.: 35H **SIN** BERECHNET DEN SINUS EINES WINKELS

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	

UP.NR.: 36H **COS** BERECHNET DEN COSINUS EINES WINKELS

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	

UP.NR.: 37H **CPLHL** ZWEIERKOMPLEMENT VON HL

Bildet das Zweierkomplement des übergebenen Wertes.

Parameter	HL	übergebener Wert
Rückgabe	HL	Zweierkomplement
Register	ACCU	

UP.NR.: 38H **HEXDEZ** HL IN DEZIMALZAHL WANDELN

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	

UP.NR.: 39H **DEZHEX** DEZIMAL IN HEXZAHL WANDELN

Umwandlung einer Dezimalzahl in einen 16 Bit Wert.

Parameter	CDE	Dezimalzahl
Rückgabe	HL	umgewandelter Wert
Register	ACCU	

UP.NR.: 3AH **LENGTH** Z80 BEFEHLSLÄNGE ERMITTELN

Berechnet die Länge eines Z80 Assemblerbefehls.

Parameter	HL	Zeiger auf den Befehlscode
Rückgabe	B	Länge des Befehls in Byte
Register	ACCU, DE, HL	

UP.NR.: 3BH **GETDATE** DATUM VON UHR3 LESEN

Liest das Datum aus einer angeschlossenen Baugruppe UHR3 aus.

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

UP.NR.: 3CH **GETTIME** UHRZEIT UND WOCHENTAG VON UHR3 LESEN

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	

UP.NR.: 3DH **USB_START** STARTET DAS USB SYSTEM

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:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_START         ; USB initialisieren
            RET C                   ; Ausgang, keine Hardware
            .........
```

Das Beispiel wird bei USB_ISDISK fortgeführt

UP.NR.: 3EH **USB_ISDISK** TEST AUF VERBUNDENEN DATENRÄGER

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

```
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_ISDISK        ; Test auf Datenträger
            RET C                   ; Ausgang, kein Datenträger
            .........
```

Das Bespiel wird bei USB_DIR fortgeführt

UP.NR.: 3FH **USB_DIR** TEST AUF VORHANDENSEIN EINER DATEI

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
RST 08h                 ; Unterprogrammverteiler
.DEFB USB_DIR           ; Test auf Vorhandensein Datei
RET C                   ; Ausgang, Datei nicht vorh.
.........
```

Das Beispiel wird bei USB_LOAD fortgeführt

UP.NR.: 40H **USB_LOAD** DATEI KOMPLETT LADEN

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_LOAD          ; Ganze Datei laden
            RET
```

UP.NR.: 41H **USB_SAVE** DATEI KOMPLETT SCHREIBEN

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 sichernder Bytes
                RST 08h                 ; Unterprogrammverteiler
                .DEFB USB_SAVE          ; Ganze Datei schreiben
                RET
```

UP.NR.: 42H **USB_OPW** DATEI ZUM SCHREIBEN ÖFFNEN

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	

UP.NR.: 43H **USB_WR** SCHREIBT DATEN IN GEÖFFNETE DATEI

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	

UP.NR.: 44H **USB_OPR** DATEI ZUM LESEN ÖFFNEN

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	

UP.NR.: 45H **USB_RD** LIEST DATEN AUS GEÖFFNETER DATEI

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	

UP.NR.: 46H **USB_CLF** DATEI SCHLIESSEN

Schließt eine zuvor mit USB_OPR oder USB_OPW geöffnete Datei.

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

UP.NR.: 47H **USB_SEK** DATEIPOINTER SETZEN

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	

```
DATEN:
            .DEFB "TEST.DAT"        ; Dateiname
            .DEFB 0                 ; Endekennung

START:
            LD IX,DATEN             ; Pointer auf Dateiname
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_OPR           ; Datei öffnen
            RET C                   ; Ausgang bei Fehler

            LD DE,100H              ; 100H Bytes
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_SEK           ; Dateipointer setzen
            RET C                   ; Ausgang bei Fehler

            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_RD            ; Byte aus Datei lesen
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_CLF           ; Datei schließen
            RET
```

UP.NR.: 48H **USB_DLF** DATEI LÖSCHEN

Löscht eine Datei vom Datenträger.

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

```
FILE:
            .DEFB "TEST.DAT"        ; Zu löschende Datei
            .DEFB 0                 ; Endekennung
```

```
START:
            LD IX,FILE              ; Pointer auf Dateiname
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_DLF           ; Datei löschen
            RET
```

UP.NR.: 49H **USB_REN** DATEI UMBENENNEN

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB USB_REN           ; OLD.DAT umbenennen
            RET
```

UP.NR.: 4AH **USB_READ** BYTE AUS DATEI LESEN

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	

UP.NR.: 4BH **USB_WRITE** BYTE IN DATEI SCHREIBEN

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

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

UP.NR.: 4CH **USB_AUS** DATENTRÄGER AUSWERFEN

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

UP.NR.: 4DH **LOINIT** DRUCKER INITIALISIEREN

Mit dieser Funktion wird die Centronics-Schnittstelle initialisiert.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	immer 01h

UP.NR.: 4EH **LO** ZEICHEN AN DRUCKER SENDEN

Mit dieser Funktion wird ein ASCII-Zeichen an die Centronics-Schnittstelle gesendet

Parameter	ACCU	auszugebendes ASCII-Zeichen
Rückgabe	CARRY	wenn Druck mit CTRL+C abgebrochen wurde
Register	ACCU	

UP.NR.: 4FH **TITLOUT** AUSGABE EINER TITELSEITE A.D. BILDSCHIRM

Löscht den aktuellen Bildschirm, generiert die Kopf- und die Fußzeile und schreibt den Titel in Großschrift oben auf den Bildschirm.

Zur vereinfachten Handhabung wurde diese Funktion im Grundprogramm auf den RST-Befehl RST 30h gelegt.

Parameter	keine	
Rückgabe	keine	
Register	keine	

```
START:
            RST 30h                 ; Unterprogrammverteiler
            .DEFB TITLOUT
            .DEFB "Funktion"        ; Überschrift-/Titeltext
            .DEFB 0                 ; Ende-Kennung

            RST 08h                 ; Unterprogrammverteiler
            .DEFB CI                ; Warten auf Tastendruck
            RET
```

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 50H **CLRCENTER** LÖSCHT DEN MITTLEREN BILDSCHIRMBEREICH

Löscht den aktuellen Bildschirm zwischen Kopf- und Fußzeile.

Parameter	keine	
Rückgabe	keine	
Register	ACCU	

UP.NR.: 51H **GETADR** EINGABE EINER ADRESSE

Erzeugt eine Eingabeaufforderung für eine Speicher-Adresse auf dem Bildschirm.

Parameter	keine	
Rückgabe	HL	Eingegebener Wert (Expression)
	IX	Startadresse eingegebener Text
	CARRY	Fehlerhafter Ausdruck
Register	ACCU	

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 52H **GETVB** EINGABE ZWEIER ADRESSEN VON, BIS

Erzeugt eine Eingabeaufforderung für zwei Adressen mit der Bezeichnung „von ADR“ und „bis ADR“ auf dem Bildschirm.

Parameter	keine	
Rückgabe	HL	„von“-Adresse
	DE	„bis“-Adresse
	CARRY	Fehlerhafter Ausdruck
Register	ACCU	

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 53H **GETVBN** EINGABE DREIER ADRESSEN VON, BIS, NACH

Erzeugt eine Eingabeaufforderung für drei Adressen mit der Bezeichnung „von ADR“, „bis ADR“ und „nach ADR“ auf dem Bildschirm.

Parameter	keine	
Rückgabe	HL	„von“-Adresse
	DE	„nach“-Adresse
	BC	Byteanzahl zwischen „von“ und incl. „bis“
	CARRY	Fehlerhafter Ausdruck

Register ACCU

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 54H **FMENU** MENÜ ÜBER DER FUSSZEILE

Erzeugt in der Zeile über der Fußzeile ein Menü für die Bedienung innerhalb oder für das Ende einer Funktion.

Parameter ACCU Bitmaske für Optionen
Rückgabe keine
Register ACCU

Bitmaske:

```
D0    R=Adr
D1    N=Neu
D2    L=Laden
D3    D=Delete
D4    frei
D5    CR=weiter
D6    M=Menü
D7    nicht verwenden!
```

```
START:
            LD A,01000010b          ; N=Neu M=Menü
            RST 08h                 ; Unterprogrammverteiler
            .DEFB FMENU                   ; Menü anzeigen

LOOP:
            RST 08h                 ; Unterprogrammverteiler
            .DEFB KI                      ; Tastenabfrage
            CP 'N'                  ; N=Neu
            JR Z,FU_NEU             ; zur Funktion „NEU“
            CP ‚M‘
            RET Z                   ; zurück zum Grundprogramm
            JR LOOP
```

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen.

UP.NR.: 55H **ENDERR** PROGRAMMABSCHLUSS ERROR

Erzeugt über der Fußzeile eine Statusmeldung „ERROR“ zum Programmende und erwartet die Tastaturquittung mit der Taste ‚M‘.

Parameter keine
Rückgabe keine
Register ACCU, HL

```
START:
            . . . . .               ;Programm
            RST 08h                 ; Unterprogrammverteiler
            .DEFB ENDERR
            RET
```

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 56H **ENDOK** PROGRAMMABSCHLUSS OK

Erzeugt über der Fußzeile eine Statusmeldung „OK“ zum Programmende und erwartet die Tastaturquittung mit der Taste ‚M‘.

Parameter	keine
Rückgabe	keine
Register	ACCU, HL

```
START:
            . . . . .               ;Programm
            RST 08h                 ; Unterprogrammverteiler
            .DEFB ENDOK
            RET
```

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 57H **ENDBRK** PROGRAMMABSCHLUSS BREAK

Erzeugt über der Fußzeile eine Statusmeldung „BREAK“ zum Programmende und erwartet die Tastaturquittung mit der Taste ‚M‘.

Parameter	keine
Rückgabe	keine
Register	ACCU, HL

```
START:
            . . . . .               ;Programm
            RST 08h                 ; Unterprogrammverteiler
            .DEFB ENDBRK
            RET
```

Schriftposition, -größe und –art sind vorgegeben und lassen sich nicht beeinflussen

UP.NR.: 58H **NEWMIN** AKTUALISIERUNG DER UHRZEIT

Aktualisiert die Uhrzeit in der Kopfzeile.

Sollte eingefügt werden, wenn das eigen Programm in einer Warteschleife auf ein externes Ereignis wartet, damit auch dann die Uhrzeit auf dem Bildschirm weiter läuft.

Parameter	keine
Rückgabe	keine
Register	keine

UP.NR.: 59H **CRC** PRÜFSUMMENBERECHNUNG

Berechnet eine CRC-CCITT (CRC-16) Prüfsumme (x^16 + x^12 + x^5 + 1).

Parameter	DE	„von“-Adresse ab der berechnet werden soll
	BC	Länge/Byte-Anzahl
Rückgabe	HL	CRC-Prüfsumme
Register	ACCU, BC, DE, HL	

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.

UP.NR.: 5AH **SETRST** RÜCKSETZEN DER RST-SPRUNG-VECTOREN

Setzt die Sprung-Vectoren (Adressen 6003..6018h) für die RST-Befehle auf die für das Grundprogramm benötigten zurück.

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

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
            RST 08h                 ; Unterprogrammverteiler
            .DEFB 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 3FFEh am Ende des zweiten ROMs eingetragen werden. Als Standard ist die Adresse 40h vorgegeben.

Steuerung des GP

An der Adresse 3FFFh 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 =1    Einblenden der Routinen zur Bankumschaltung
Bit 1 =1    Anzeigen der Versionsnummer in der Kopfzeile
Bit 2 =1    im ReAssembler Hexzahlen immer normieren
Bit 3 =1    BANKBOOT2 ist vorhanden
Bit 4 =1    Baugruppe UHR3 vorhanden
Bit 5       reserviert
Bit 6 =1    Drucker aktiv
Bit 7       Aktivieren des Interrupt Mode 2
```

Als Standardwert ist hier 00010011b entsprechend 13h eingetragen, somit ist die Bankumschaltung, die Anzeige der Versionsnummer und die Zeitanzeige aktiv.

Interrupt Mode 2

Im Interrupt Mode 2 wird die Adresse der auszuführenden Interrupt-Routine aus einer Adresstabelle gelesen, auf deren Eintrag ein Zeiger, der durch den Inhalt des Registers I (HwB) und dem Interruptvektor der auslösenden Hardware (NwB) gebildet wird, zeigt.

Wenn der Interrupt Mode 2 des Grundprogramms aktiv ist, wird das I-Register mit dem Wert 61h geladen. Die auslösende Hardware ergänzt das I-Register (HwB) mit einem niederwertigeren Byte zu einer Zeiger-Adresse in den RAM-Bereich von 6100h bis 6180h. Dieser ist für Interrupt-Routinen-Adressen 2 freigehalten und kann durch den Anwender beliebig belegt werden. Die dort hinterlegten Adressen werden dann sofort angesprungen, ohne dass ein weiterer Befehl nötig wäre.

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.

Der Adressraum 7000..7FFFh wird für System-Erweiterungen frei gehalten. Das bedeutet Grund-programmfunktionen, die aus Platzmangel in dieser Version entfallen mussten, wie z.B. für die serielle Ein-/Ausgabe und für den PROMER können mit der Lade-Funktion in diesen Bereich geladen werden. Die genannten Funktionen werden für diesen Bereich angeboten werden und sind nach dem laden über das Extended Menü aufrufbar.

Unterprogramm Tabelle zum Einfügen

In der nachfolgenden Form werden die UP-Aufrufe als Datenwort verwendet. Dabei ist der Programmaufruf RST 08h für den Unterprogrammverteiler bereits enthalten und eigentliche Aufruf erfolgt nur noch mit dem Datenwort.

```
0000            prthl: .equ 13CFh      ;UP-Nr. + Z80-Code RST 08h
                ;
0000 21 00 80   Start: ld HL,8000h
0003 CF 13             .defw prthl     ;ruft das UP prthl auf
0005 00                NOP
0006            ;
```

Bei Bedarf ab hier oder Teile davon in das eigene Assemblerlisting kopieren:

```
;------KEYBOARD
csts:         .equ 00CFh  ;Status KEY lesen
ci:           .equ 01CFh  ;Zeichen von KEY einlesen nach ACCU
ki:           .equ 02CFh  ;wie CI mit Wandlung nach Großbuchstaben
;
;------GDP:
wait:         .equ 03CFh  ;Warten bis GDP bereit
waitsync:     .equ 04CFh  ;Warten auf Vertical Blank
cmd:          .equ 05CFh  ;Kommandoausgabe an GDP, ADDU=Kommando-Byte
fast:         .equ 06CFh  ;Schneller unsynchronisierter Schreibmodus
slow:         .equ 07CFh  ;Normaler synchronisierter Schreibmodus
;
setpen:       .equ 08CFh  ;GDP Schreibmodus setzen
erapen:       .equ 09CFh  ;GDP Löschmodus setzen
setviewpage:  .equ 0ACFh  ;Anzeigeseite setzen (ohne Aktivierung)
setwrtpage:   .equ 0BCFh  ;Schreibseite setzen (ohne Aktivierung)
aktpage:      .equ 0CCFh  ;Seite gemäß viewpage und aktpage aktivieren
setaktpage:   .equ 0DCFh  ;Seite setzen und aktivieren
clrall:       .equ 0ECFh  ;alle Bildschirmseiten löschen
clrakt:       .equ 0FCFh  ;aktuelle Bildschirmseite löschen
clrinvis:     .equ 10CFh  ;Seite loeschen auch unsichtbar
kill:         .equ 11CFh  ;GP Funktion auf Seite zurücksetzen
;
```

```
;------ AUSGABETEXT über Buffer
      ;-- 1.Schritt
getausbuf:   .equ 12CFh  ;Textbuffer initialisieren -> IX = Pointer
      ;-- 2.Schritte (mögl.)
prthl:       .equ 13CFh  ;HL HEX in Buffer ab IX schreiben
prthld:      .equ 14CFh  ;HL DEZ in Buffer ab IX schreiben
prtac:       .equ 15CFh  ;ACCU HEX in Buffer ab IX schreiben
prtacd:      .equ 16CFh  ;ACCU DEZ in Buffer ab IX schreiben
prtbin:      .equ 17CFh  ;ACCU BIN in Buffer ab IX schreiben
print:       .equ 18CFh  ;Text ab (HL) in Buffer ab IX schreiben
printin:     .equ 19CFh  ;n. Call folgende Bytes,0 in Buffer
zeich:       .equ 1ACFh  ;Zeichen in ACCU in Buffer ab IX schreiben
newline:     .equ 1BCFh  ;0dh, 0ah in Buffer schreiben IX
      ;-- 3.Schritt:
printbuf:    .equ 1CCFh  ;ld HL=X-Pos, DE=Y-Pos, A=Schrift in Buffer und
                         ;gibt Buffer auf BS aus (incl. 0ah etc)

;------- AUSGABETEXT ohne Buffer
textaus:     .equ 1DCFh  ;Text a.BS von HL
textmulti:   .equ 1ECFh  ;Mehrere Texte ausgeben wie vor
txtout:      .equ 1FCFh  ;Text a.BS ausgeben Parameter+Text nach Call
TXTCSR:      .equ 20CFh  ;Text a.BS auf CSR-Position weiter (noch
nicht!)
;
;------ EINGABETEXT
textein:     .equ 21CFh  ;hl->Versorgungsblock c=flag 0,1
textxy:      .equ 22CFh  ;Texteingabefeld
gethl:       .equ 23CFh  ;Eingabe als Zahl interpretieren -> HL
gettext:     .equ 24CFh  ;IX
expr:        .equ 25CFh  ;Auswertung eines Ausdrucks (rechnet)
getpara:     .equ 26CFh  ;Parametereingabe mit Bezeichnung (HL)
;
;------ AUSGABEGRAFIK
moveto:      .equ 27CFh  ;Grafik Cursor setzen
drawto:      .equ 28CFh  ;Linie zeichnen
tmove:       .equ 29CFh  ;Turtle Position setzen
tschreite:   .equ 2ACFh  ;Turtle vorwärts bewegen (Symbol SCHREITE)
tschr16tel:  .equ 2BCFh  ;Turtle langsam vorwärts bewegen
                         ;(Symbol SCHR16TEL)
tdrehe:      .equ 2CCFh  ;Turtle rotieren
thoch:       .equ 2DCFh  ;Turtle anheben (unsichtbar bewegen)
trunter:     .equ 2ECFh  ;Turtle absenken (sichtbar schreiben)
turtle:      .equ 2FCFh  ;Turtle zeichnen (setpen/erapen)
sprite:      .equ 30CFh  ;Sprite ausgeben (IX=Definition, B=Size)
;
;------ ALLGEMEINES
upper:       .equ 31CFh  ;Char in ACCU in Großbuchstaben wandeln
waitms:      .equ 32CFh  ;Wartezeit in Millisekunden
wait100ms:   .equ 33CFh  ;100 Millisekunden warten
adj360:      .equ 34CFh  ;HL
sin:         .equ 35CFh  ;HL
cos:         .equ 36CFh  ;HL
cplhl:       .equ 37CFh  ;HL
hexdez:      .equ 38CFh  ;Umwandlung HL(hex) -> CDE(dez)
dezhex:      .equ 39CFh  ;Umwandlung CDE(dez) -> HL(hex)
length:      .equ 3ACFh  ;Länge eines Z80 Befehls ermitteln
```

```
;
;------ UHR
getDate:      .equ 3BCFh  ;E=Tag, D=Monat, HL=Jahr
getTime:      .equ 3CCFh  ;C=Stunde, D=Minute, E=Sekunde,
                          ;B=Wochentag (1..7)
;
;-------USB
usb_start:    .equ 3DCFh  ;USB Initialisieren
usb_isdisk:   .equ 3ECFh  ;Laufwerk prüfen
usb_dir:      .equ 3FCFh  ;Test ob Datei vorhanden
usb_load:     .equ 40CFh  ;Datei komplett laden
usb_save:     .equ 41CFh  ;Datei komplett schreiben
usb_opw:      .equ 42CFh  ;Datei zum Schreiben öffnen
usb_wr:       .equ 43CFh  ;Bytes in Datei schreiben
usb_opr:      .equ 44CFh  ;Datei zum Lesen öffnen
usb_rd:       .equ 45CFh  ;Bytes aus Datei lesen
usb_clf:      .equ 46CFh  ;Datei schließen
usb_sek:      .equ 47CFh  ;Pointer in Datei setzen (DE=Position)
usb_dlf:      .equ 48CFh  ;Delete File
usb_ren:      .equ 49CFh  ;Rename File
vnc_read:     .equ 4ACFh  ;Byte aus geöffneter Datei lesen
vnc_write:    .equ 4BCFh  ;Byte in geöffnete Datei schreiben
vnc_aus:      .equ 4CCFh  ;Datenträger auswerfen
;
;--- Drucker
LOINIT:       .equ 4DCFh  ;Drucker CER Init
LO:           .equ 4ECFh  ;Druckerausgabe Accu
;
;--- Monitor/Menübedienung
TITLout:      .equ 4FCFh  ;Titelanzeige
clrcenter:    .equ 50CFh  ;Mittelbereich löschen
getadr:       .equ 51CFh  ;eine Adresse abfragen
getvb:        .equ 52CFh  ;Adressen von-bis abfragen
getvbn:       .equ 53CFh  ;Adressen von-bis-nach abfragen
FMENU:        .equ 54CFh  ;Fussmenue
endERR:       .equ 55CFh  ;Programmende mit ERROR
endOK:        .equ 56CFh  ;Programmende mit OK
endBRK:       .equ 57CFh  ;Programmende mit BREAK
;
NewMin:       .equ 58CFh  ;Uhr aktualisieren
crc:          .equ 59CFh  ;Prüfsumme berechnen  Länge, HL=crc
setRST:       .equ 5ACFh  ;Setzt RST-Sprung-Vectoren zurück
```

Aktualisierungen

04.01.2020	GP-2019	S. Reimer
14.02.2018	GP2018v2	A. Rohmann
02.02.2018	GP2018v1	A. Rohmann