

Mega-Flash-Software



Die Megaflash-Software läuft auf allen [MEGA-Modulen](#), [MEGA-Flash-Modulen](#), dem [64K-SRAM-Modul von U. Zander](#), dem [Kombi-Modul](#) u.a. Der Einfachheit halber ist hier nur von Megamodul die Rede.

Das Modul wird anstelle eines ggf. vorhandenen BASIC-Moduls gesteckt. Je nach Ausführung werden auch keine zusätzlichen 16K-RAM-Module benötigt.

Download

Das Softwarepaket umfasst alle enthaltenen Programme (meist im TAP-Format), die Quelltexte der Modulsoftware, Makefiles und einige Hilfsprogramme. die Software ist universell für alle Megamodul-Varianten kompilierbar. Bitte die LIESMICH-Datei beachten!

- [mega_flash.zip](#) Softwarepaket incl. aller Quellen (Stand 7.7.2020)

Im Paket sind enthalten:

- megarom.bin für JKCEMU
- ROM_1.bin ..ROM_5.bin für Mega-Flash-Modul v. buebchen
- kombimodul.bin für 64K-SRAM-Modul sowie Kombi-Modul von U.Zander
- komplette Quellen zum Selbstkompilieren und Anpassen an eigene Wünsche (s.u.)

Für das ältere Megamodul werden nur 3 ROMs benötigt. Der ROM-Inhalt ist derselbe wie fürs Mega-Flash-Modul v. buebchen.

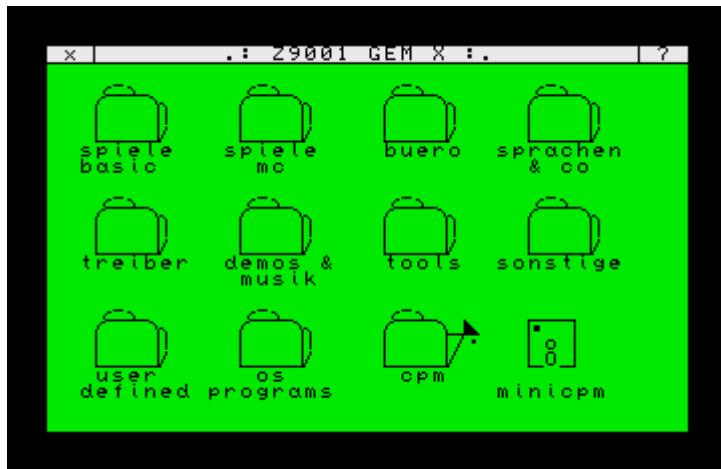
rom_1 und rom_2 sowie rom_3 und rom_4 sind jeweils hintereinander in einen 1-MByte-ROM zu brennen. rom_5 wird unverändert in den dritten ROM gebrannt.

Einstieg

```

robotron Z 9001      MEGA-FLASH-ROM
OS
>DIR
ASGN      TIME      CLLOAD      #
CCP       DIR        DUMP        FILL
TRANS     RUN        IN          OUT
MEM       EOR        CLS         C
EDIT      ASM        IDAS        #
ZM        EEPROM     BASIC       WBASIC
R80       WR80       ZSID        ZSIDW
EPROMMER EPROMMEP  GRAF        CRT40P
CRT80P   GLOBUS87   GRAF14     GRAFP
N-ECK    SWITCH-2   UHR         #
BOOT     BITEX      K6311      K6312
>

```



Im Megamodul sind Hunderte Programme enthalten, die direkt gestartet werden können, ohne dass man sie umständlich von Kassette laden muss.

Die Programme können zum einem ganz normal über die Kommandozeile aufgerufen werden. Selbst Basicprogramme starten automatisch. Zum etwas komfortableren Umgang mit den vielen Programmen gibt es diverse Tools für besagte Kommandozeile. Wer will, kann sich mit **DIR** die komplette Programmliste anzeigen lassen bzw. falls einem der Name nur so olala bekannt ist, kann man auch auf dem Modul danach suchen lassen. „Jokern“ (*) hilft dabei ungemein 😊.

Wer es lieber etwas 'moderner' möchte, kann es auch einmal mit **MENU** probieren. 'Maus', Fenster, anim. Icons usw. sind mit Bordmitteln realisiert.

Software gibt es reichlich: Spinne, Buggy, Pegasus, KC Pascal , Pretty C, Plotter- und Druckertreiber, Script, Text, ... sind nur wenige aus einer breiten Programmpalette. Alles ist soweit wie möglich aufeinander abgestimmt. Basicprogramme können beispielsweise im Speicher belassen werden und mit der IDAS ROM Version im Speicher verändert werden oder Plottertreiber werden beim Aufruf der Plotterdemos ebenfalls im Speicher belassen bzw. die Speicherkonfiguration entsprechend angeglichen. Man kann also gleich losplotten.

Daneben darf man auch mit CP/M etwas herum spielen. Es gibt eine kleine CP/M-Version, die kein Floppymodul und auch nicht das spezielle 64K-RAM-Modul benötigt. Dafür bekommt man dann auch gleich eine, wenn auch winzige, Ramfloppy an die Hand, in der bsw. Pascalprogramme gespeichert und im KC87 Modus auf Kassette ausgelagert oder auch eingeladen werden können. Zum anderen liegen auch schon 1,5 Diskettenabbilder (die 2te Disk hat dann doch nicht mehr ganz gepasst) mit reichlich KC87 CP/M Software auf dem Modul, so dass man auch mit diesem Betriebssystem ein wenig herum spielen kann.

Programme werden einfach durch Eingabe des Programmnamens am OS-Prompt gestartet. Das gilt für alle Programmarten. Bei BASIC-Programmen erfolgt automatisch ein spezielles Laden des BASICs mit anschließendem Autostart des BASIC-Programms.

Eine Anleitung für alle Programme des Moduls gibt es nicht. Viele Programme sind in den Dokumentationen von Robotron beschrieben. Im Download-Paket ist eine Übersicht `packeroms.bin_mega.txt` enthalten, die in etwa **DIR L entspricht.**

Kommandos

Im Modul sind einige spezielle Kommandos (Programme) enthalten, die die Arbeit mit dem Modul und mit dem KC ermöglichen. Für den reinen Anwender sind vor allem **MENU**, **DIR**, **HELP** wichtig.

Kommando	Beschreibung
DIR [suchmuster]	Auflisten aller Kommandos
DIR L [suchmuster]	Auflisten incl. Banknummer, Bankadr. und Startadr.
DIR C [suchmuster]	Auflisten incl. Banknummer und Kommentar
HELP [kommando]	Hilfe anzeigen
CLS	Bildschirm löschen
C	Cursor on/off
MENU	graphische Oberfläche als GEM

DIR [suchmuster]

Alle Kommandos(Programme), die auf dem Modul enthalten sind, werden aufgelistet.

Die Anzeige kann mit PAUSE angehalten werden. Eine beliebige Taste setzt die Anzeige fort. Mit STOP wird das Kommando abgebrochen.

Programme mit FA-Rahmen werden in Cyan ausgeschrieben.

DIR L [suchmuster], DIR C [suchmuster]

Mit diesem Kommando erhält man einen Überblick über die Belegung des Mega-ROM-Moduls.

Alle Programme werden aufgelistet. Es erfolgt eine ausführliche (**L** = lange) Anzeige. Mit dem Parameter **C** (comment) wird zusätzlich eine kurze Beschreibung angezeigt.

Die Programme werden in der Reihenfolge angezeigt, in der sie im Modul abgelegt sind und in der sie auch gesucht werden.

Die Anzeige kann mit PAUSE angehalten werden. Eine beliebige Taste setzt die Anzeige fort. Mit STOP wird das Kommando abgebrochen.

Bei normalen Programmen (mir OS-Rahmen) wird die Banknummer, der Kommandoname, die Adresse des Kommandorahmens und die eigentliche Startadresse angezeigt. Bei Programmen mit FA-Rahmen werden angezeigt: Bank, Adr. in Bank, Dateityp, Name, Anfangsadresse, Endadresse, Startadresse, Dateikategorie (s.u.).

```
robotron Z 9001 ** MEGA-FLASH-ROM **
U.Pohlerts 12/24/2012
OS
>DIR R+A R+AFRI1 R+AUTOCR R+AFRI2
>DIR D*MO R+DEMO1 R+DEMO2 R+DEMO3
R+GRDEMO R+DEMO1 R+DEMO2 R+DEMO3
R+PLDEMO MESSDEMO SE-DEMO1 Z90-DEMO
>DIR M*B
0D E600 MC EMONB2 B200 BC43 B200 1
12 D8000 MC RAMBASIC IC 0300 2AFF 2400 1
32 D5000 BAS S+MAULBAS 0401 2044 0000 2
48 D3000 BAS CLIMBEBAS 0401 2B74 0000 2
7D D2000 BAS CLIMBEBAS 0401 25BB 0000 2
84 C2000 BAS FORMEL1B 0401 0FFF 0000 2
89 D8000 BAS KOMBAS 0401 092E 0000 2
A3 E600 MC LAMBADA 0401 092E 0000 2
>
```

Zur Einschränkung der Anzeige kann dem DIR-Kommando ein **Suchmuster** übergeben werden. Ein '*' steht dabei für eine beliebige Anzahl beliebiger Zeichen (auch 0!) und '?' für genau ein beliebiges Zeichen.

Im obigen Bild werden erst alle Kommandos angezeigt, die 'R+A' im Namen enthalten. 'D*MO' sucht nach einem 'D' und irgendwo danach ein 'MO' im Namen.

Das letzte Beispiel 'M*B' zeigt die Suche nach 'M' und danach 'B' im Namen. Bei 'EMONB2' und 'RAMBASIC' sieht man, dass der * zwischen 'M' und 'B' einmal für 2 und einmal für 0 Zeichen steht.

HELP [Kommando] (Hilfe)

```
robotron Z 9001 ** MEGA-FLASH-ROM **
U.Pohlerts 12/24/2012
OS
>HELP
Anzeige einer kurzen Hilfe
Aufruf: HELP Kommando
moegliche Kommandos:
SYSTEM F8S KRT SAVE
ZM IDAS EDIT RSM
CPM DOS
>HELP SAVE
Speichern auf Kassette (OSSAVE robotron)

Aufruf: SAVE name[.typ] von,bis[,sadr]

fehlt typ, wird COM genommen
fehlt saddr, wird addr genommen
>
```

Es wird eine kurze Hilfe zu einem Kommando angezeigt. Ohne Parameter werden alle vorhandenen Hilfetexte aufgelistet.

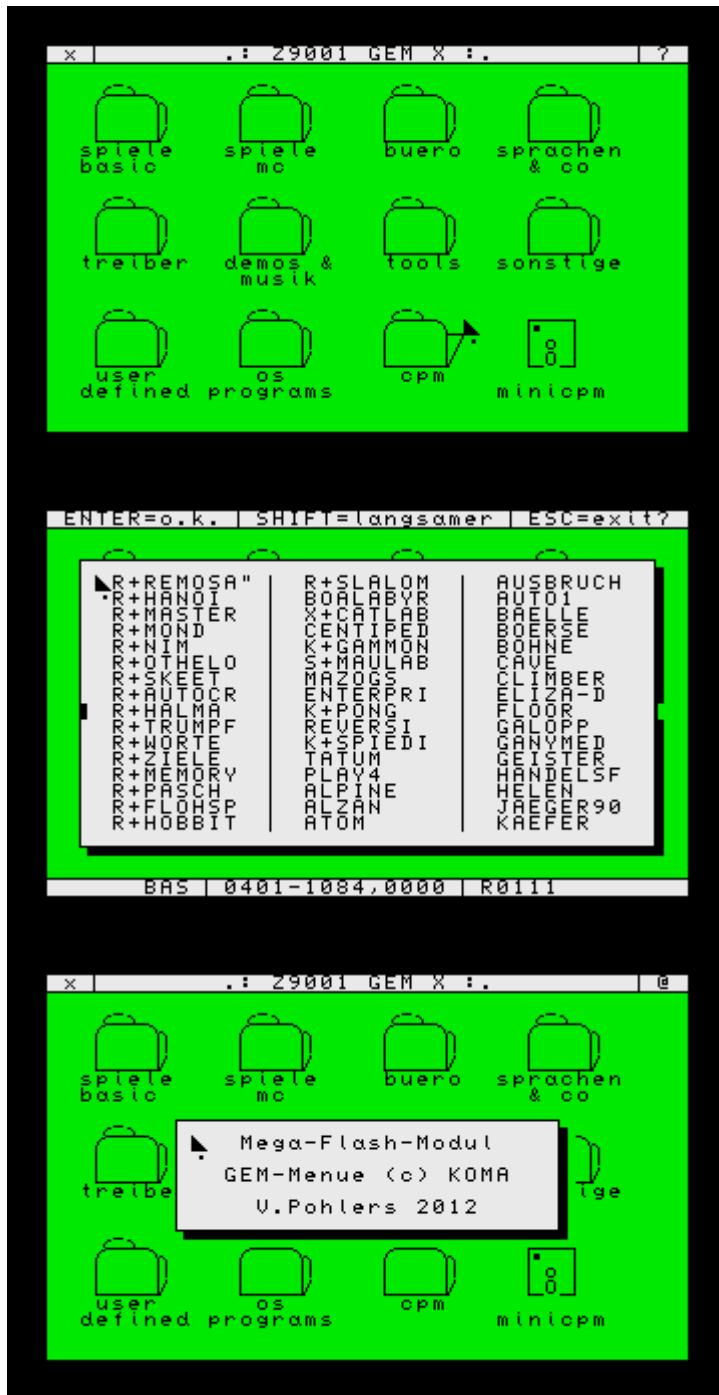
CLS (Bildschirm löschen)

Hiermit wird der Bildschirm gelöscht.

C (Cursor on/off)

Besitzer eines Z9001/KC87 mit Farbmodul, aber nur über Antennenkabel angeschlossenen Fernseher, sehen keinen Cursor, da dieser als blinkender Farbhintergrund ausgegeben wird. Mit diesem Kommando wird die Cursoranzeige auf s/w umgestellt. D.h., der Cursor wird als blinkendes Quadrat angezeigt. Ein nochmaliger Aufruf dieses Kommandos macht dies wieder rückgängig.

MENU (graphische Oberfläche)



Die graphische Oberfläche wurde basierend auf der Software des [Megamodul](#) geschrieben. Die Bedienung erfolgt wie dort.

Nach Eingabe von MENU sieht man eine Oberfläche, die sich an GEM orientiert.

Mit den Cursortasten wird der Mauszeiger bewegt. Wenn gleichzeitig Shift gedrückt wird, bewegt sich der Mauszeiger wesentlich langsamer. Mit ENTER öffnet man einen Ordner, mit ESC schließt man diesen wieder. Bei großen Ordnern ist ein Scrollen möglich.

In der Statuszeile werden ständig Informationen über das Programm unterm Mauszeiger wie z.B. der genutzte Speicherbereich angezeigt. Ist der Mauszeiger über dem gewünschten Programm, muss man einfach nur ENTER drücken, und das Programm startet.

Über das X links oben kann MENU verlassen werden (mit dem Mauszeiger aufs X gehen und ENTER drücken).

Hinter dem Fragezeichen rechts oben verbirgt sich ein About-Fenster.

WINDOW [erste _zeile, letzte _zeile, erste _spalte, letzte _spalte]

Analog zu BASIC: Diese Anweisung gestattet, einen rechteckigen Abschnitt des Bildschirms als Ausgabebereich zu definieren. Innerhalb des Ausgabebereiches erscheinen sämtliche Ausgaben. Ohne Parameter wird der volle Bildschirm eingestellt.

WINDOW entspricht also WINDOW 0,23,0,39 bzw. WINDOW 0,23,0,79 im CRT80-Modus.

Dateiarbeit

Zum Arbeiten mit Kassette, USB-Stick und Diskette stehen dieselben Kommandos zur Verfügung. Je nach geladenem Treiber USBX oder DOSX wird statt Kassette der USB-Stick bzw. Diskette genutzt.

Die Parameter sind hexadezimal anzugeben. Eine Vornull ist nicht nötig.

SAVE aadr eadr [sadr] [1] Datei speichern auf Kassette/USB/Diskette

Speicher von Adresse aadr bis einschließlich eadr

Es erfolgt eine Abfrage von name[.typ] fehlt typ, wird COM genommen

Fehlt eine Angabe der Startadresse sadr, wird die Anfangsadresse aadr genommen
folgt noch eine 1, wird ohne Kopfblock gespeichert (nur DOSX, USBX)

LOAD [aadr] [1] Datei laden

analog SAVE Abfrage von Name+Typ

aadr ist alternative Anfangsadresse

folgt eine 1, wird auch der erste Block normal gelesen (nur DOSX, USBX)

FCB Anzeige des FCB

Aus dem aktuellen FCB (File Control Block, Adresse 005Ch) werden Dateiname, Dateityp, Anfangsadresse, Endadresse und Startadresse angezeigt

EOR [adr] Anzeigen/Setzen EOR

Mit diesem Kommando wird der aktuelle Wert von EOR (end of ram, 0036h) angezeigt. Treiber können den Wert verändern und sich damit vor Überschreiben schützen. Das BASIC der Mega-Software beachtet bei der Ermittlung des memory end genau diesen Wert.

Mit einem Parameter kann der EOR gezielt geändert werden.

Maschinencode

Zur Arbeit mit Maschinencode gibt es ein paar dem Z1013 entlehnte Kommandos. Die Parameter sind hexadezimal anzugeben. Eine Vornull ist nicht nötig.

Kommando	Beschreibung
DUMP von bis	Speicher anzeigen HEX/ASCII

Kommando	Beschreibung
FILL von bis byte	Speicher mit Byte füllen
TRANS von ziel anzahl	Speicherbereich kopieren
IN port	Port einlesen
OUT port byte	Portausgabe
RUN adr [port]	Programmstart von Adr.
MEM adr	Speicher editieren (neue Byte(s) eingeben + Enter, zurück mit R, Ende mit ;)

MEM adr (Modify Memory)

Es ist möglich, mit diesem Kommando einen Speicherbereich ab der angegebenen Anfangsadresse byteweise anzuzeigen und gegebenenfalls zu verändern. Es erfolgt die Ausgabe der aktuellen Adresse und des Inhaltes des zugehörigen Bytes. Anschließend wird mit dem Zeichen „#“ zur Eingabe aufgefordert. Soll der alte Inhalt beibehalten werden, ist nur die Enter-Taste zu betätigen, ansonsten wird vorher eine hexadezimale Zahl eingegeben. Es können auch mehrere Byteinhalte, durch Leerzeichen voneinander getrennt, eingegeben werden.

Nach Betätigung der Enter-Taste wird die aktuelle Adresse erhöht und auf der nächsten Zeile fortgesetzt. Wird versucht, einen nicht vorhandenen Speicherbereich oder einen ROM zu beschreiben, erfolgt eine Fehlerausschrift: ER aerr bb, wobei aerr die Adresse und bb den fehlerhaften Inhalt darstellen. Anschließend wird eine erneute Eingabe erwartet. Diese Fehlerausschrift wird vor allem dann auftreten, wenn versucht wird, nicht vorhandene Speicher oder Festwertspeicher zu beschreiben. Mit Eingabe des Zeichens „R“ kann die aktuelle Adresse bei Bedarf zurückgestellt werden.

Die Kommandoausführung wird beendet durch Eingabe eines Semikolon „;“. Die aktuelle Adresse wird als Endadresse übernommen. Mit dem Kommando 'DUMP' kann der aktualisierte Speicherbereich nochmals auf dem Bildschirm angezeigt werden.

DUMP aadr eadr (Display Memory)



Mit diesem Kommando können beliebige Speicherbereiche zwischen einer Anfangs- und einer Endadresse angezeigt werden. Die Anzeige des Bereiches zwischen FFF8 und FFFF ist mit dem D-Kommando nicht möglich, dafür muss das M-Kommando verwendet werden. Die Anzeige erfolgt zeilenweise in hexadezimaler Form. Zuerst wird die Adresse des jeweiligen Bereiches ausgegeben, danach folgen acht Byte des Speicherinhaltes, gefolgt von der ASCII-Darstellung. Es wird immer eine Zeile vollständig ausgegeben, auch wenn die Endadresse eine andere Anzahl von Bytes verlangt.

Die Anzeige kann mit PAUSE angehalten werden. Eine beliebige Taste setzt die Anzeige fort. Mit STOP wird das Kommando abgebrochen.

FILL aadr eadr bb

Damit ist es möglich, einen angegebenen Speicherbereich zu löschen oder mit dem Byte bb zu füllen. Wird das Kommando ohne Parameter verwendet, wird der gesamte adressierbare Speicher gelöscht. Weiterarbeit ist dann nur nach Betätigen der Resettaste möglich.

TRANS aadr zadr anz (Transfer)

Es erfolgt ein Transport eines Speicherbereiches ab der Anfangsadresse auf eine Zieladresse mit der festgelegten Anzahl von Bytes. Dabei ist eine Überlappung der beiden Bereiche möglich.

IN port (Port einlesen)

Der angegebene Port wird gelesen. Das Ergebnis wird angezeigt.

OUT port byte (Portausgabe)

Es wird eine Datenbyte byte auf den Port port ausgegeben.

RUN adr [bank] (Programmstart)

Mit diesem Kommando können Programme gestartet werden, auch wenn sie nicht über einen OS-Kommandorahmen verfügen und somit nicht per Kommandoname ausgeführt werden können.

Ein Programm auf Adresse adr wird gestartet. Mit RET kehrt das Programm zum OS zurück. Optional kann eine Bank angegeben werden. Ist dies der Fall, wird zuerst die Bank aktiviert, ehe das Programm gestartet wird. Dadurch können Programme gestartet werden, die in einer anderen Bank als der Systembank liegen.

Mit **RUN F000 bank** wird das Megamodul hart auf eine andere Bank als die Systembank umgeschaltet. Das Megamodul verhält sich dann wie ein normales 10K-ROM-Modul; das OS-Verhalten bzgl. Programmsuche und -start ist unverändert original Z9001. Erst nach einem Hardware-Reset ist die Modul-Systemsoftware wieder aktiv.

spezielle Programme

Zuätzliche (für mich) wichtige Programme sind:

Kommando	Beschreibung
CHKROM	Test des Moduls. Für jede Bank wird eine Prüfsumme errechnet und mit dem Sollwert verglichen. Fehler werden angezeigt.
EPSON	Druckertreiber für verschiedene Druckertypen
V24	Der ROM BM116, enthält die Robotron-Druckertreiber V24A1, V24A2, V24A3
TR_SAMML	Treiber-Paket aus mp 10/87, umfasst SIFE,SIFA,CENTR,LX86,TD40,BEEP
ZM30, ZMA8	angepasste Versionen des Zusatzmonitors, kooperativ zu anderen Treibern
DEVICE	ähnlich ASGN, aber mit mehr Möglichkeiten
SYSINFO	Analyse der Hardware

Kommando	Beschreibung
CRC	CRC-Berechnung
CRT40	CRT-Treiber für schnelles Bildschirmlöschen
INITKC	Diskettenformatieren im OS. Damit kann man eine Bootdiskette nun selbst erstellen: initkc, cpm-48k, pip a:=c:@cpmz9.com
DOSX	Treiber für Diskette statt Kassette. Im Basic gibt es dazu den Befehl DIR.
USBX	Treiber für USB statt Kassette. Im Basic gibt es dazu die Befehle DIR und CD.
RAMTEST	RAM-Test Y21SO (FA 12/86), an Z9001 adaptiert (vp)

Software ins Modul integrieren

Ende 2011 habe ich diese alternative quelloffene Software für das Mega-Modul und das Mega-Flash-Modul geschrieben. Diese besteht i.W. aus einer OS-Erweiterung; damit Programme in allen Bänken gesucht und von dort gestartet werden können. Es ist **keine** Änderung des OS nötig. Eigene Programme können leicht ins Modul aufgenommen werden:

Das OS des Z9001 ist analog zum CP/M aufgebaut. Die oberste Schicht, die Kommandoeingabe CCP, kann durch ein eigenes Programm ersetzt werden. Dazu dient das Kommando „#“. Die Mega-Flash-Software nutzt genau dies aus, um das CCP zu erweitern.

Außerdem wurde eine **Bankrückschaltung** integriert; so das Programme beim Beenden wieder die Bank mit der Systemerweiterung (kurz Systembank) aktivieren.

Dadurch kann jede Software, z.B. originale ROM-Modul-Software, unverändert bleiben. Es muss keine spezielle Enderoutine o.a. gepatcht werden.

Der Z9001 kann verschiedene Programme gleichzeitig im Speicher halten. Das Betriebssystem OS findet und startet das jeweilige OS-Programm anhand eines speziellen Codebereiches, dem Kommandorahmen (OS-Rahmen genannt). Das erweiterte CCP in der Mega-Flash-Software durchsucht nicht nur den sichtbaren Speicherbereich 100h-E7FFh, sondern alle Bänke des Mega-Moduls nach solchen OS-Kommandrahmen¹⁾.

Aus moderner Computersicht ist der Kommandorahmen so eine Art Dateiname; das Dateisystem entspricht dem Speicher. Die Position im Filesystem (der Pfad) wäre damit das Analogon zur Startadresse.

Dieser **OS-Rahmen** muss auf einer xx00h-Adresse liegen und sieht so aus

```
org xx00h
jp start
db "NAME      "      ; genau 8 Zeichen
db 0          ; Ende eines Kommandos
db 0          ; Ende der Liste
```

Details s. OS-Handbuch. Die hier stehenden Programmnamen können im CCP eingegeben werden. Das CCP sucht den Programmnamen in allen Kommandorahmen und startet bei gefundenem Programmnamen das Programm. Andernfalls erscheint die Ausschrift „start tape“.

Der OS-Kommandorahmen ist im Modul für Programme nutzbar, die im Speicherbereich von C000h-

E7FFh arbeiten (also z.B. Inhalte originaler ROM-Module), oder die eine eigene Umladeroutine besitzen, die das eigentliche Programm erst an die Zieladresse im RAM kopieren und dort starten.

Beispiel für ein Programm mit OS-Rahmen. Hier stehen 3 Kommandos im OS-Rahmen. Die xx00-Adresse im EPROM muss mit der korrekten Lage im Z9001 korrespondieren (hier wäre das C000h).

Für die Mega-Flash-Software wurde außerdem ein neuer Kommandorahmen eingeführt: der **FA-Rahmen**²⁾. Dieser Kommandorahmen wird vom erweiterten CCP der Mega-Flash-Software ebenso wie ein normaler OS-Kommandorahmen durchsucht, um ein eingegebenes Kommando zu finden. Der FA-Rahmen ist für Programme nutzbar, die im Speicherbereich von 100h-BFFFh arbeiten.

Der FA-Rahmen ist 32 Byte lang und liegt ebenfalls auf einer xx00h-Adresse. Nach dem FA-Rahmen folgt das Programm. Der FA-Rahmen hat folgenden Aufbau:

```
org xx00h          ; header
db 0FAh, 0FAh      ; +0 Kennbytes
db Dateityp        ; +2 0-MC, 1-BASIC (s. includes.asm)
db "NAME"          ; +3 genau 8 Zeichen
dw aadr            ; +11 Anfangsadresse im RAM (wichtig!)
dw eadr            ; +13 Endadresse im RAM (kann 0 sein oder wie in TAP
angegeben)
dw sadr            ; +15 Startadresse im RAM (oder FFFFh - nichtstartend)
(wichtig!)
dw länge           ; +17 (Datei-)Länge des Programms (ohne Header) (wichtig!)
db Dateikategorie ; +19 0-Standard (s. includes.asm)
db "Kommentar"     ; +20 12 Zeichen, beliebig, z.B. Autor o.ä.
;
binclude programm.bin
```

Im Unterschied zum OS-Rahmen werden also wesentlich mehr Informationen („Dateiattribute“) vorgehalten.

Beispiel für Programm mit FA-Rahmen. Das originale, unveränderte RAM-Programm folgt direkt auf den Rahmen.

Typ (00-MC, 01-Basic)	Kennung	Programmname	Anfangs- adr.	End- adr.	Start- adr.		
	00023500:	FA FA 00 5A 4D 33 30 20 20 20 20 00 30 FF 3F 00				úú ZM36	0ij?
	00023510:	30 00 0D 00 43 4F 4D 20 20 20 20 20 20 20 20 20				0 . COM	

Der Beispiel-FA-Rahmen im Detail. Die tatsächliche Programmlänge entspricht nicht dem im Kassettenheader angegebenen Bereich bis 3FFF, sondern ist kürzer.

Dieser FA-Kommandorahmen ist für beliebige RAM-MC-Programme, aber auch BASIC- Programme nutzbar (geplant ist die Unterstützung weiterer Dateitypen z.B. Forth- Programme).

MC-Programme mit diesem Rahmen werden zuerst an die korrekte Adresse aadr im Speicher umgelagert und dann auf der Startadresse saddr gestartet. programm.bin ist einfach der binäre Speicherabzug des Programms von aadr bis eadr. Praktisch ist das die *.KCC-Datei OHNE den Kopfblock. (Die Informationen aus dem Kopfblock stehen schon alle im FA-Rahmen; damit spart man ein bisschen Speicher im Mega-Modul).

Basic-Programme werden nach 0401h kopiert. Dann wird die BASIC-Bank zugeschaltet, Basic initialisiert und das Programm gestartet. Für BASIC-Programme ist programm.bin einfach die *.KCC-Datei bzw. *.SSS-Datei, aber ohne die ersten 11 Byte (Typ+Name).

Zur einfachen Konvertierung von *.tap-Dateien ins binäre Format kann das Perl-Programm `tap2bin.pl` genutzt werden.

Passt programm.bin nicht mehr komplett in die aktuelle Bank, wird es einfach in der nächsten Bank

fortgesetzt.

Außerdem können alle Programme mit **bitbuster _ extreme** um etwa 30% komprimiert sein, um Platz im Mega-Modul zu sparen. Bei komprimierten Programmen muss im Dateityp das Bit 7 gesetzt sein (also 80h zum originalen Dateityp addiert).

Das Mega-Modul kann somit einfach um eigene Software erweitert werden: Einfach in einer beliebigen Bank (außer der Systembank) in einem freien Bereich auf einer xx00h-Adresse ein Programm mit OS-Rahmen oder mit FA-Rahmen speichern. Fertig!

Kompilieren

Im download-Paket sind alle Quellcodes, Programme, Tools etc. enthalten, um die ROMs zu erstellen. Das Kompilieren erfolgt unter Windows, sollte sich aber relativ leicht auf Linux anpassen lassen.

Benötigt werden zusätzlich der [Arnold-Assembler AS](#) und Perl (z.B. [Activeperl](#) oder [strawberryperl](#)).

Im **makefile** sind die Pfade zum Assembler und zu Perl anzupassen.

Die Datei **00liesmich** beschreibt alle notwendigen Schritte des Kompilierens.

Die Datei **includes.asm** ist anzupassen. Sie enthält alle Parameter zur Generierung. Hier wird das zu generierende System eingestellt (MEGA oder KOMBI). Mit lastbank wird die ROM-Größe beschränkt.

Die Datei **packedroms.asm** enthält die Zusammenstellung der im ROM enthaltenen Programme. Die Zusammenstellung kann in weiten Grenzen frei erfolgen. Lediglich Systembank und die beiden BASIC-Bänke sollten wie vorgegeben bleiben.

Die Roms werden dann mit

```
make depend  
make
```

und

make mega	für Mega-Modul (3 ROMs)
make flash	für Mega-Flash-Modul (5 ROMs)
make kombi	f. 64K-SRAM-Modul, KOMBI-Modul bzw. Buebchen-Rx3 (1 ROM)

erstellt.

Interessierte Bastler (und Linuxer) finden hier den Packer zum Verkleinern der Dateigröße:

- <http://www.west.co.tt/matt/speccy/apology/> bitbuster _ extreme-0.1.tar.gz (unten auf der Seite). Der genutzte Packer.
- <http://www.msx.org/downloads/related/development/bitbuster-12> bitbuster1_2.zip. Das ist das originale Bitbuster-Paket. bitbuster _ extreme spart sich nur den 4 Byte großen Header; ist ansonsten unverändert. Dieser originale Packer kann nicht für die Mega-Software verwendet werden, es geht nur die Extreme-Variante!

Historie

9.2.2012: Die Suche mit DIR wurde komplett neu geschrieben. Der neue Algorithmus arbeitet nun ca. 3x so schnell und bietet mehr Optionen (s.u. DIR-Befehl).

Sämtliche Parameter werden an das Programm übergeben. So kann etwa mit **SAVE** test 300,39f ein Programm auf Kassette gespeichert werden, obwohl SAVE (das ist das OS-SAVE der Kassette R0111) selbst als FA-Programm vorliegt und erst in den Speicher geladen und entpackt werden muss, ehe es gestartet wird!

14.2.2012: Es gibt neue Software, z.B. **R+MESSE2**, eine Demo der Leipziger Herbstmesse 1984. Neu ist **CPM**: Damit kann CP/M gestartet werden, ohne dass eine Systemdiskette im Laufwerk liegen muss. Das Programm ist für robotron- und Rossendorf-Hardware geeignet. **BASIC** wurde auf 16 Farben-Unterstützung gepatcht. Der Start von BASIC-Dateien wurde optimiert. Bei eigenen CONS-Treibern wie CRT40P oder CRT80P wird nun automatisch ein angepasstes BASIC gestartet. Man kann z.B. CRT40P und R+INFO hintereinander starten. u.v.a.m.

24.02.2012: **Disk-OS** ist als DOS4 bzw. DOSX mit drin.

10.03.2012: Es gibt 3 original CP/Ms (robotron, Rossendorf, robotron 48k). Das erspart eine Bootdiskette, der Bootvorgang geht auch schneller. **BOOT** und **BOOTZFK** gibt es natürlich auch noch. Als viertes gibt es das **MiniCPM**. In der include-Datei **includes.asm** kann festgelegt werden, ob 1 oder 2 ROM-Floppies genutzt werden sollen. Außerdem sind diverse Testprogramme wie **CHKROM**, **BANKTEST**, **LPRO** etc. hinzugefügt.

20.03.2012: Updates in KRT-Grafiksoftware, CP/M und anderen Programmen. Version zum KC-Treffen 2012.

04.04.2012: **HELP**-Kommando.

23.09.2012: **MENU**-Kommando, bekannt vom Megamodul. Allerdings werden hier die Datei-Dialoge dynamisch erzeugt. Und zum Erstellen der Hilfe-Dateien für das **HELP**-Kommando gibt es nun einen kleinen Editor.

14.10.2012: Korrekturen in der Systemsoftware bzgl. Startverhalten

27.12.2012: Erweiterungen in Vorbereitung auf eine neues Mega-FRR-Modul. Und 100 neue Programme!

11.01.2013: kleines Update, u.a. DISK-OS nun auch für Floppies.

01.04.2013: Korrigiertes DiskOS, geprüfte Installation.

25.04.2013: Anpassung für OS 1.1 (Z9001.84)

19.08.2013: Fehler in MEM behoben; Eingabe eines einzelnen 00-Werts war nicht möglich. Neue Systemkommandos **SAVE** und **FCB**. Kommando **SDX** in System-ROM aufgenommen. Ebenso V24X.

25.01.2015: nach diversen miniänderungen und Anpassungen an andere Moduloptionen ist nun die Bankumschaltung modifiziert. MENU läuft wieder

30.04.2015: Fehlerkorrektur DIR-Wildcard-Routine, MENU Anzeige OS-Namen Beta-Version für UZ-64K-SRAM-Modul mit abwechselnd 10K und 6K-Bänken.

17.06.2015: Version für **UZ-64K-SRAM-Modul:** make ROM

Zusätzlich zum gewohnten Umfang gibt es Kommandos für die Uhr (RTC, DAT, ...), die Entpackroutine ist anders, und ein paar kleine Ergänzungen. Die Suche ist künstlich verlangsamt, damit man die Banknummern durchlaufen sieht (Wunsch von Ulrich, weil er das gut findet, wenn man was durchlaufen sieht). Die Banknummer wird rechts oben mit angezeigt. Uhrzeit und Datum sollten nach Reset zu sehen sein. Der Zusatzmonitor ZM wurde modifiziert, damit er besser mit der KRT zusammenarbeitet.

Systemerweiterung: auf C02E steht die Nr. der letzten Bank. Patchen auf tatsächliche ROM-Größe: (man kann auch wie bisher in includes.asm die EPROM-Größe vorauswählen)

;UZ:

```
;27010 128K EPROM => 0fh
;27020 256K EPROM => 1fh
;27040 512K EPROM => 3fh
;27080 1M EPROM => 7fh
```

22.05.2016: neue Version für 64K-SRAM- und Kombi-Modul umfangreiche Überarbeitung der Bankrückschaltung, damit die Lade-Bank in der Anzeige stehen bleibt (bei Kombi-Modul aktiv). Unterstützung von 64K-SRAM und Kombi-Modul mit gemeinsamen Code. Unterstützung des USB-Moduls im OS (Kommando USB). Unpacker depack_extreme.exe für gepackte bin-Dateien (ohne Fa-Header). Als Folge der überarbeiteten Bankrückschaltung müssen Programme, die die Systembank nutzen (z.B. Sprungsverteiler), nun unbedingt ft_systembank im FA- Dateityp-Byte enthalten. Jetzt wird eine Textdatei packedroms.bin.txt erstellt, die den Inhalt des ROMs auflistet (ähnlich dem DIR-Kommando im OS). Für 64K-SRAM-Modul und KOMBI-Modul wird nun dieselbe ROM-Datei „kombimodul.bin“ genutzt. Die Software erkennt das korrekte Modul und gibt eine entsprechende Meldung aus.

07.07.2016: MiniCPM für Kombi-Modul, Korrekturen f. Kombi-Modul, GIDE-RTC im Megaflash. GIDE-Treiber, CPM-48K um ROM-Floppy erweitert, auch für Kombi-Modul extra Version, INITKC zum Diskettenformatieren im OS. Damit kann man eine Bootdiskette nun selbst erstellen: initkc, cpm-48k, pip a:=c:@cpmz9.com

07.12.2016: neu LOAD, USB heißt jetzt USBX, kleine Korrekturen im Code. Dank an Rolf W. fürs Testen und Fehlerfinden!

15.03.2017: MiniCPM wieder lauffähig (Dank an M.Bagola fürs Fehlerfinden). Zu ROM-Disk die KCNET-Software TFTP1287.COM und CPMN1587.COM hinzugefügt.

01.08.2017: neue Version vor allem fürs Kombi-Modul (seit April) Nagelneues MINICPM speziell fürs KOMBI-Modul, mit RAM-Disk-Einbindung. Banknummer wird jetzt heruntergezählt (BWS r.o.). Wenn der ZM als Geräte- Treiber oder Debugginghilfe genutzt wird (rst 38), geht das nur mit der RAM-Version.

23.11.2017: Rolf W. hat mich auf eine Idee gebracht. Wenn in (fcb+24) ein 'N' steht, wird in den Treibern DOSX, USBX bei OPENW kein Block geschrieben. Das wird z.B. in BASIC genutzt. Damit sind die erzeugten SSS-Dateien kompatibel zu CP/M usw.

02.10.2018: Neues DOSX fürs Kombimodul. Statt der RAF2008 wird die Modul-RAM- Disk unterstützt, genau wie im MiniCPM. BASIC speichert auf Disk/USB nun analog zur CP/M-Version als „ZBS“ ab. Neuer Parameter in includes für UZ-Version. Wider besseren Wissens wird dann die ROM-Version des ZM sowie Ulrichs Spezial-Version des ZM30 etc. eingebunden.

05.03.2019: Diverse kleine Änderungen in USBX, DOSX, System für bessere Zusammenarbeit. Die

CRTxx-Treiber und Speicher-Treiber DOSX/USBX werden nun nach <STOP> automatisch reinitialisiert. Nach dem Laden eines FA-Programms wird der FCB passend gefüllt. Damit kann man mit SAVE (ohne Parameter) einfach Programme aus dem ROM herauskopieren und auf ein externes Speichermedium übertragen. Zur Arbeit mit Disketten steht zur Erstellung einer Systemdiskette INITKC und CPM-48K bereit. Damit kann eine Diskette formatiert und mit dem OS und weiteren Z9001-spezifischen Treibern beschrieben werden. Danach wird mit BOOT das CP/M gestartet. USBX hat neue Kommandos DDIR und CD (s. Projekt USB-OS).

Die Extra-Version der Kombi-Variante für U.Zander und sein geändertes Betriebssystem wird nur empfohlen, wenn man auch das geänderte Betriebssystem nutzt. Ansonsten gibt es die bekannten Probleme mit den originalen Versionen des ZM. Das automatische Reinitialisieren obiger Treiber ist aufgrund der originalen Versionen des ZM nicht möglich. MINICPM ist auf Wunsch weiterhin drin, aber nicht besonders sinnvoll.

13.03.2020 Mit Build 147 des Arnold-Assemblers musste die Verwendung von Unterverzeichnissen geändert werden (Es wird jetzt relativ zu dem Verzeichnis gesucht, in dem die Datei mit dem Include-Statement liegt, und nicht mehr relativ zum aktuellen Verzeichnis). Das betrifft alle asm-Sourcen und Perl- Tools, die was mit include zu tun haben.

16.03.2020 Basic mit DIR, CD unter DOSX,USBX; sonst Fehlermeldung. DOSX merkt sich das aktuelle Laufwerk

08.04.2020 CHKROM nun auch f. KOMBI, neu DEVICE, neue Version ZM30+ZMA8 Korrektur BASIC

07.07.2020 Korrektur GRAFx (Version passend zu BASIC kompilieren), neu WORD

¹⁾

Und außerdem nach FA-Kommandorahmen

²⁾

Falls es jemand interessiert: Den Namen FA-Rahmen habe ich nach den Kennungsbyte FAh gewählt. Dieses Kennungsbyte ist FLASH ohne die nicht Hexa-Ziffern, also FAh

From:

<https://hc-ddr.hucki.net/wiki/> - **Homecomputer DDR**

Permanent link:

<https://hc-ddr.hucki.net/wiki/doku.php/z9001/software/mega?rev=1594129162>

Last update: **2020/07/07 13:39**

