Mega-Flash-Modul

Die Hardware des Mega-Flash-Modul wurde von W.Harwardt entwickelt (MEGA-Flash-ROM-RAM, http://buebchen.jimdo.com/8-bit-selbstbau/kc87-z9001/). Die Idee basiert auf dem ursprünglichen Megamodul vom A.S.

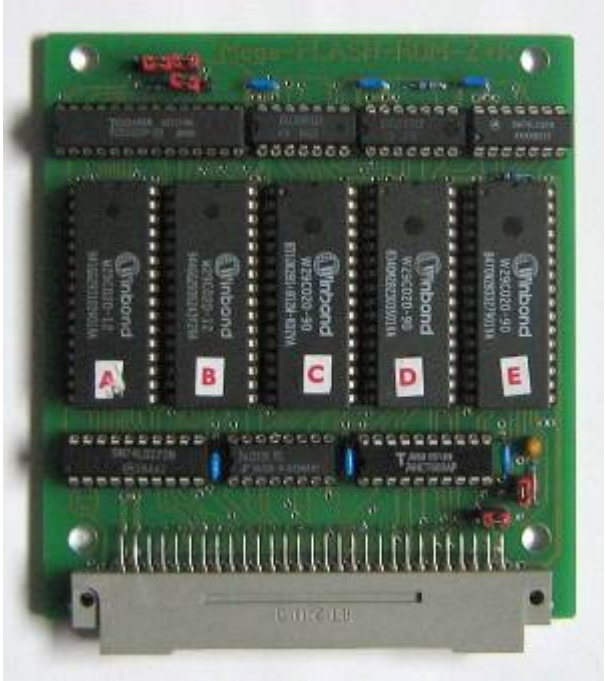
Hardware

Das Mega-Flash-Modul ist äquivalent zum älteren Megamodul, kommt aber im Gegensatz zu diesem ohne GALs aus und ist sowohl für den Z9001 bzw. KC87 als auch für den Z1013 ausgelegt. Das Modul basiert auf dem Funktionsprinzip des Megamodul von Alexander Schön und Speichererweiterungen für den Amstrad CPC, jedoch mit völlig neu entwickelter und erweiterter Schaltung, aber ohne die RAM-Speichererweiterung.

Die 2012- und 2013-Serie des Mega-Flash-Moduls enthalten die 32K-RAM-Speichererweiterung.

Schaltungs- und Aufbauunterlagen s. http://buebchen.jimdo.com/8-bit-selbstbau/z1013/ und http://buebchen.jimdo.com/8-bit-selbstbau/kc87-z9001/

Vorder- und Rückseite Modul Serie 1.10.2011. Hier mit 5 EPROMs bestückt. Bei Nutzung von FLASH-Speicher muss das Steckfeld oben links anders belegt werden. In dieser Serie war noch eine kleine Änderung nötig (Drahtbrücke auf Leiterseite von Jumper an Steckverbinder B9 statt an B27).



Serie 2013 mit 32K-RAM

Vorteile:

- 5 gleiche 512KByte ROMs
- alternativ 5 FLASH-ROMs, z.B. Winbond W90C040 o.a.
- auch kleinere ROMs o. FLASHs nutzbar!
- alternativ 5 RAMs, z.B. zur Nutzung als RAM-Disk
- frei wählbare I/O-Adresse
- abschaltbar (über X1:27A in Verbindung mit dem 64K-RAM-Modul)
- 32K statischer RAM, abschaltbar

Gegenüber der Megamodul-Version von Rauscher/Honi enhält die 2011-Serie keinen 32K-RAM. Es werden hier zusätzlich 2 normale 16K-RAM-Module, ein 64K-RAM-Modul oder ein 128K-RAM-Modul benötigt, falls der Grundspeicher für ein Programm nicht ausreichend ist.

Download

- mega_flash.zip Softwarepaket incl. aller Quellen (Stand 1.4.2013)
- 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 Paket. bitbuster_extreme spart sich nur den 4 Byte großen Header; ist ansonsten unverändert.

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. Mann 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.

Software

Ende 2011 habe ich eine 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. Diese Software kann ohne Änderung auch am Megamodul nach A.S. genutzt werden.

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.

Zur komfortablen Arbeit mit dem Modul gibt es den Kommandozeilenbefehl **DIR** und die graphische Oberfläche **MENU** (s.u.).

Eigene Programme können einfach ins Modul integriert werden (s.u.).

Zusätzlich zu den Kommandos der Modul-Verwaltung stehen einige vom Z1013 inspirierte Kommandos zur Speicherarbeit bereit.

Kommando	**Beschreibung**
DIR [suchmuster]	Auflisten aller Kommandos
DIR L [suchmuster]	Auflisten incl. Banknummer, Bankadr. und Startadr.
HELP [kommando]	Hilfe anzeigen
DUMP von bis	Speicher anzeigen HEX/ASCII
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 ;)
CLS	Bildschirm löschen
C	Cursor on/off
MENU	graphische Oberfläche ala GEM

DIR [suchmuster]

Alle transienten Kommandos werden aufgelistet. Die Anzeige erfolgt absteigend von FF00 bis 100h, anschließend werden alle nachfolgenden Bänke bis Bank FF jeweils von C000 bis E700 durchsucht. Angezeigt wird nur der Kommandoname.

Programme mit FA-Rahmen werden in Cyan ausgeschrieben.

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

DIR L [suchmuster]

Alle transienten Kommandos werden aufgelistet. Die Anzeige erfolgt absteigend von FF00 bis 100h, anschließend werden alle nachfolgenden Bänke bis Bank FF jeweils von C000 bis E700 durchsucht. Angezeigt wird die Banknummer, der Kommandoname, die Adresse des Kommandorahmens und die eigentliche Startadresse. Mit diesem Kommando erhält man einen Überblick über die Belegung des Mega-ROM-Moduls.

Bei Programmen mit FA-Rahmen werden angezeigt: Bank, Adr. in Bank, Dateityp, Name, Anfangsadresse, Endadresse, Startadresse, Dateikategorie (s.u.).

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

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)

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

Hilfetexte sind einfache FA-Dateien vom Typ ft_HELP (2). Sie können Farbe u.a. Steuercodes enthalten.

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 Fehleraussohrift: 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 Komandoausführung 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 Uberlappung 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.

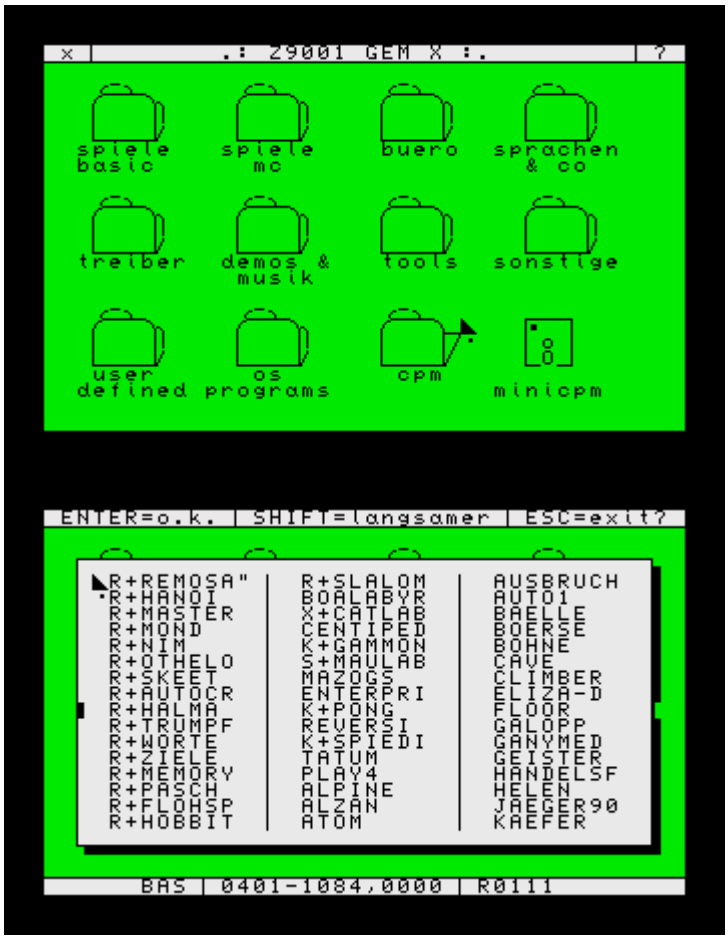
CLS (Bildschirm löschen)

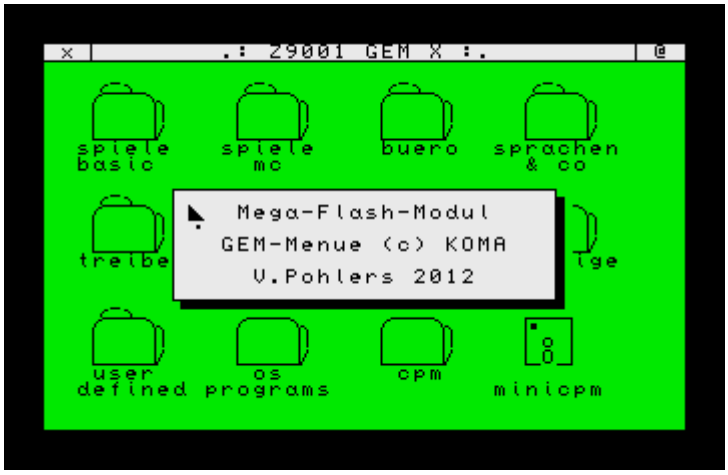
Hiermit wird der Bildschirm gelöscht. Es wird das Zeichen ^L an das aktuelle Konsolengerät gesendet.

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)

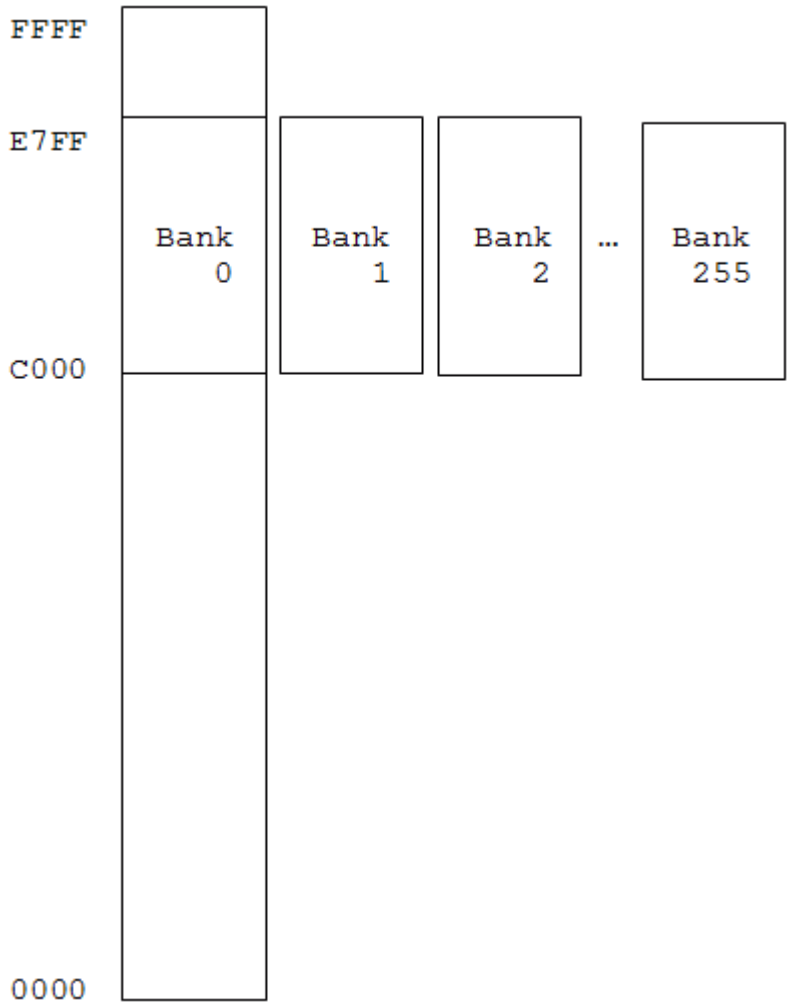




Die graphische Oberfläche wurde basierend auf der Software des Megamodul geschrieben. Die Bedienung erfolgt wie dort. Technisch wurde aber vieles anders als beim Megamodul gelöst. So ist MENU ein normales Programm, die Dateidialoge sind nicht vorgegeben, sondern werden zur Laufzeit gebildet. Außerdem ist bei großen Dialogen ein Scrollen möglich (z.B. bei den BASIC-Spielen). Zusätzlich werden in der Statuszeile Informationen über das gerade selektierte Programm wie z.B. der genutzte Speicherbereich angezeigt.

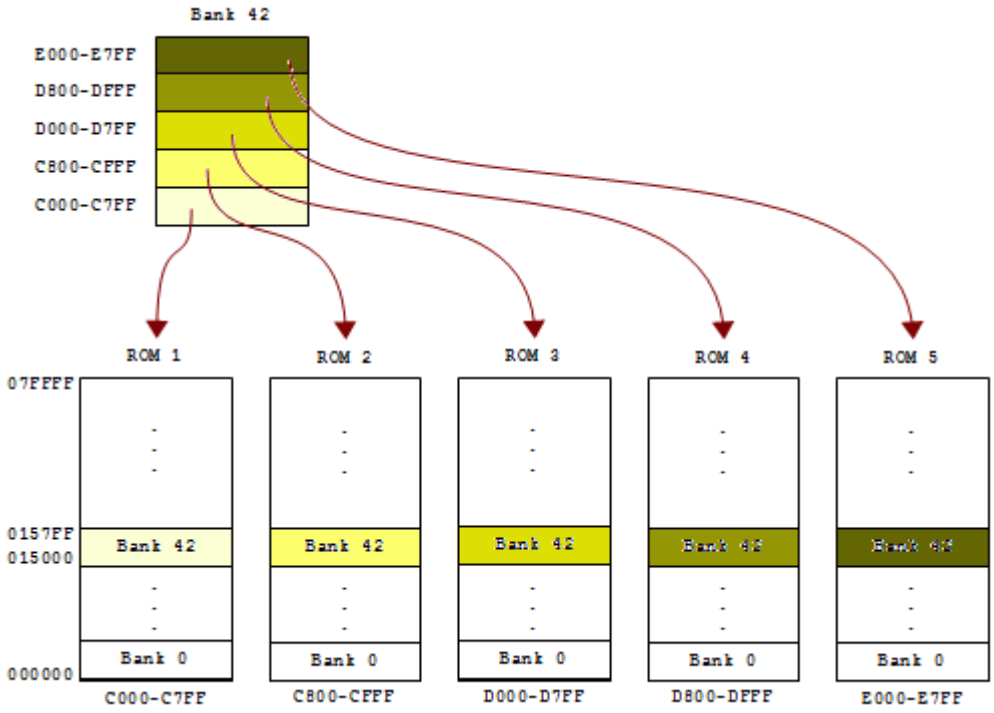
Funktionsweise

Das Modul besteht logisch aus Sicht des Z9001 aus 256 Speicherbereichen (sogenannten Bänken) im Speicherbereich C000-E7FFh. Durch Ausgabe der Banknummer auf Port FFh wird die gewählte Bank aktiv, d.h. im Adressraum des Z9001 ist im Bereich C000-E7FFh der Inhalt dieser Bank zu sehen. Es ist immer nur genau eine Bank aktiv und nutzbar.



Physisch sind die Bänke auf die 5 ROMs wie folgt dargestellt verteilt. Jede Bank wird in 5 2K große Teile aufgeteilt und auf die 5 ROMs aufgesplittet.

(Beim Megamodul sind es nur 3 ROMs; 1+2 und 3+4 sind jeweils zusammen in einem 1-MByte-ROM enthalten. Ansonsten ist es das gleiche Verfahren)



Jeder ROM enthält jeweils 2K große Abschnitte aus ALLEN Bänken.

Wenn man selbst Software in das Mega-Flash-Modul integrieren will, muss man diese Aufteilung in 2K-Häppchen unbedingt beachten!

eigene Software

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 Systemerweitertung (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-Kommandrahmen[1)].

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.

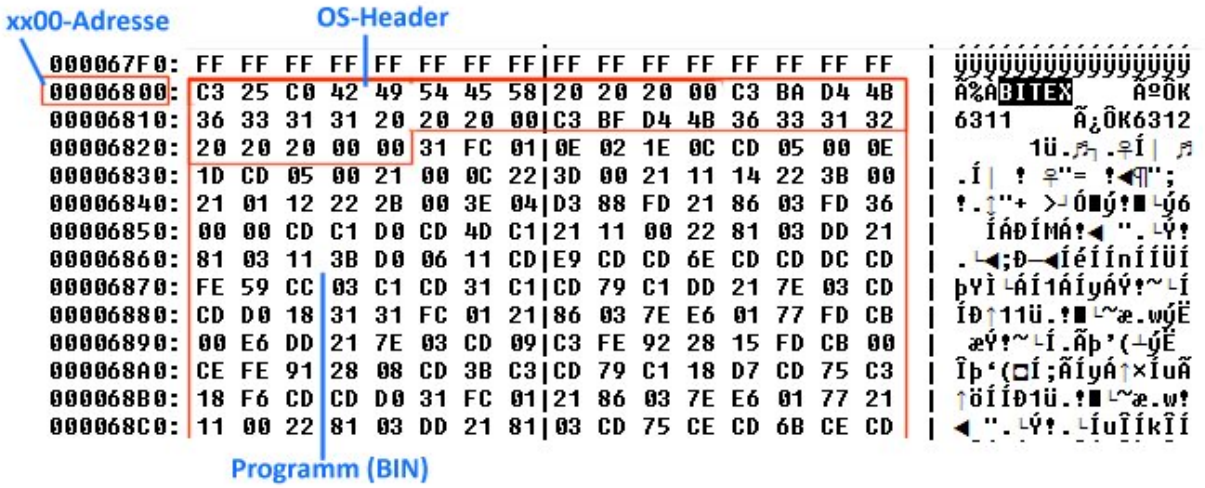
Dieser **OS-Rahmen** muss auf einer xx00h-Adresse liegen und sieht so aus

```
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).

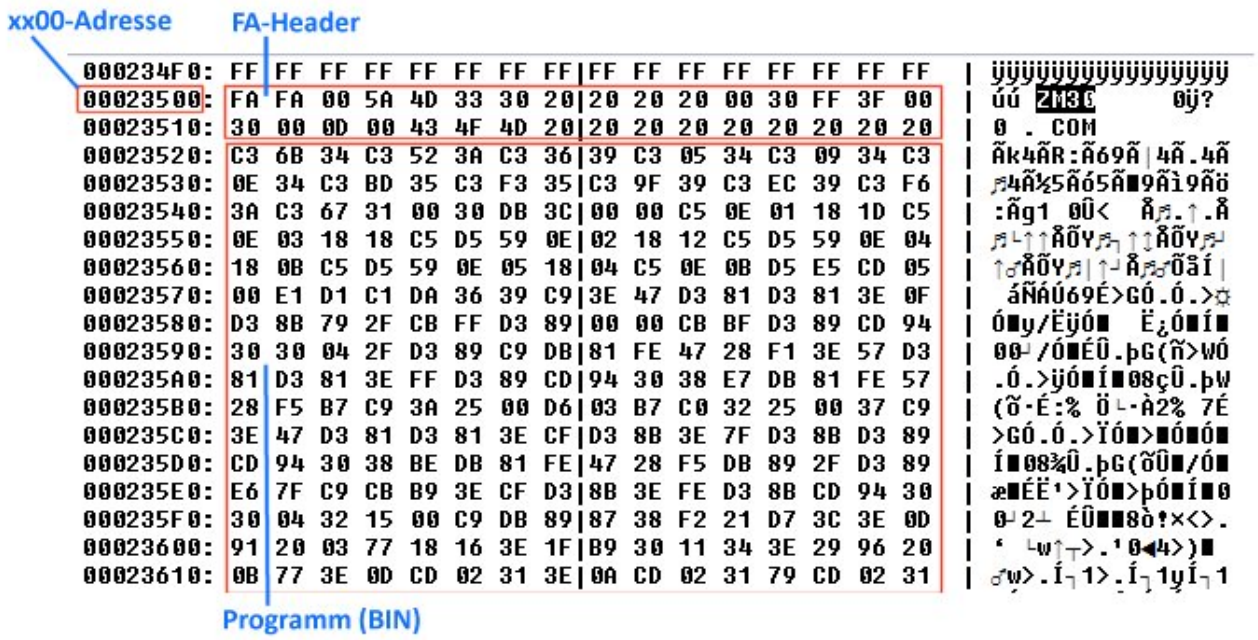
Es wurde außerdem ein neuer Kommandorahmen eingeführt: der **FA-Rahmen**[2]. Dieser Kommandorahmen wird vom erweiterten CCP der Mega-Flash-Software ebenso wie ein normaler OS-Kommandorahmen durchsucht, um ein eingegebenes Kommando zu finden. Der FA-Rahmen ist für Programme nutzbar, die im Speicherbereich 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, bel., 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 unveränderte Programm folgt direkt auf den Rahmen.

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 sadr 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.

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!

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