

Modul-1

Downloads

- modul_1.zip

ROM-Inhalte und die extrahierten Programme von 7 verschiedenen Zusammenstellungen außerdem reass. Quellcode der Softwareversion „Paket X.2 (ROM - Disk Verwaltung)“.

- ac1-modul-1-roms.zip

Meine Paket-X-Software incl. angepasster JKCEMU-AC1 zum testen Stand 11.08.2026

- ac1-modul-1.zip

Die Quellen zu meiner Software incl. Softwaresammlung, Doku etc. Stand 11.08.2026

Hinweis zum Nutzen des angepassten JKCEMU-AC1:

- bnkanf EQU 8000h, nextbnkprc equ 1
- ROM-Datei für Programmpaket X: paket_x.bin
- ROM-Datei für ROM-Disk: packedroms.bin
- ROM-Disk einblenden ab: 8000h

oder:

- bnkanf EQU 8000h, nextbnkprc equ 2
- ROM-Datei für Programmpaket X: paket_x.bin
- ROM-Datei für ROM-Disk: jkcemu_xrom.bin
- ROM-Disk einblenden ab: 8000h
- ROM-Disk für physisches Modul-1 am AC1-2010: packedroms.bin

ggf vorab mit testrom.bin die zu nutzende Verschachtelung testen!

Technik

Das originale [SCCH-Modul-1 \(ROMDISK\)](#) stellt zusätzlichen ROM zur Verfügung, und zwar:

- 8 KByte für Grafik/Sound-BASIC-Interpreter „BASIC“
- 8 KByte für Programmpaket X (Software f. ROM-Disk) „PAKET-X“
- je nach Ausbau 256 oder 512 KByte für ROM-Disk „ROM1“ + „ROM2“

Gesteuert wird das Modul über Port 14h:

- Bit 0: Programmpaket X im Bereich E000-FFFF einblenden
- Bit 1: Grafik/Sound-BASIC im Bereich 4000-5FFF einblenden
- Bit 2: CP/M-Modus (Monitor und BWS aus, alles RAM, AC1-2010: Signal C22)
- Bit 3: Ein Segment der ROM-Disk im Bereich C000-FFFF bzw. 8000-FFFF einblenden,

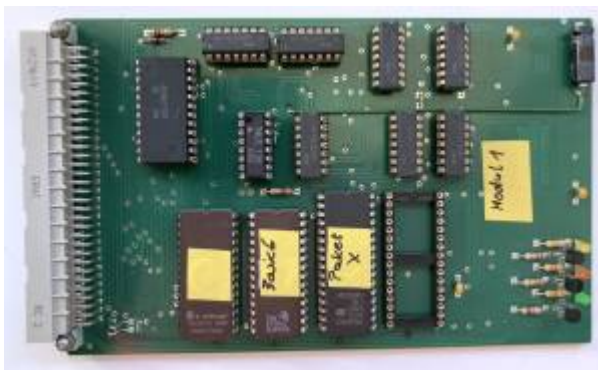
- Bit 0 aktiviert dabei ROM1 bzw. ROM2
- die Bits 4, 5 und 6 legen das entsprechende 16K- bzw. 32K-Segment fest.

Oder (andere Hardware, nur 1 ROM):

- Bit 3: Ein Segment der ROM-Disk im Bereich C000-FFFF bzw. 8000-FFFF einblenden,
 - die Bits 0,4,5,6 legen das entsprechende 16K- bzw. 32K-Segment fest.

Das [Modul-1 des AC1-2010](#) ist gegenüber dem Original erweitert:

- 16 KByte (8 KByte) für Grafik/Sound-BASIC-Interpreter
- 8 KByte für Programmpaket X
- 1 MByte für ROM-Disk, verteilt auf zwei EPROMs 27C4001 (512kx8).



Gesteuert wird das Modul über Port 14h:

- Bit 0: Programmpaket X im Bereich E000-FFFF einblenden
- Bit 1: Grafik/Sound-BASIC im Bereich 4000-5FFF bzw. 2000-5FFF einblenden (je nach Jumper JP1)
- Bit 3: Ein Segment der ROM-Disk im Bereich 8000-FFFF einblenden
 - Bit 0 aktiviert dabei ROM1 bzw. ROM2
 - die Bits 4-7 legen das entsprechende 32K-Segment fest.

Die ROM-Disk wird in Bänken zu je 32 kB in den Bereich 8000h...FFFFh eingeblendet. Die oberen (beim AC1 nicht verwendeten) 4 Bytes des ROM- Konfigurationsregisters #14 steuern die dabei einzelnen Bänke. Wird mit Bit3 des Registers die ROM-Disk aktiviert, so steuert Bit0 (wie im Original) die Auswahl des EPROMs. Folgende Zuordnungen ergeben sich damit:

```

00  RAM
01  Programm "X",      E000-FFFF
02  BASIC,            2000-5FFF (4000-5FFF bei JP1 offen (8k) )
n8  Bank Nr. n ROM1    8000-FFFF
n9  Bank Nr. n ROM2    8000-FFFF
also
08h EPROM1 Bank0 09h EPROM2 Bank0
18h EPROM1 Bank1 19h EPROM2 Bank1
... ..
F8h EPROM1 Bank15 F9h EPROM2 Bank15, jeweils 8000-FFFF

```

Modul-1 - Ergänzungen:

- neben dem 8k-Basic war auch die Hardwaremöglichkeit für ein 16k-Basic ab #2000 vorgesehen.
- Die Rombank wurde ursprünglich ab #C000 eingeblendet, als die ROM's größer wurden, kam die Einblendung schon ab #8000 hinzu.
- Es gab mehrere Versionen des Programmpaket X, die aber vom E-Ludwig abgeleitet wurden: (E. Ludwig / Beisler / Fey = optisch schönste Variante)

Software

Voraussetzung für die Nutzung des Modul-1 ist ein AC1 mit SCCH-Monitor V8 oder 10/88 oder kompatibel.

Mit dem Kommando **X** wird mit Port 14h, Bit 1 der „PAKET-X“-Eprom aktiviert und die Modul-Software namens „Paket X“ auf dem auf Adresse E000 gestartet. Die Software listet die in der ROM-Bank abgelegten Programme auf. Nach Auswahl wird das entsprechende Programm in den RAM geladen, die ROMBANK deaktiviert und das Programm gestartet. BASIC-Programme können als Systemabzug gespeichert werden. Ein modifiziertes BASIC 3.2 auf dem Modul wird dazu benötigt. Je nach Softwareversion des „Paket X“ sind auch gepackte Programme möglich, die im ROM weniger Speicherplatz benötigen.

Das Hinzufügen weiterer Programme in die ROMBANK ist etwas komplexer. Der reine Binärabzug muss in ROM1 oder ROM2 so abgelegt werden, dass er vollständig in ein Segment passt. Zusätzlich muss im „PAKET-X“-Eprom ein Eintrag für das Programm geschrieben werden sowie passender Umlade- und Startcode. Es stehen dafür diverse Routinen bereit (je nach Softwareversion des „Paket X“ unterschiedliche Parameter, Funktionsadressen, Funktionen, ...)

Zum Zerlegen vorhandener ROM-Bänke und der Extraktion der darauf liegenden Software habe ich ein paar Perl-Programme geschrieben. Die Programme erwarten 2 oder 3 Parameter

```
perl splitrom_xx.pl paketx.bin rom1.bin [rom2.bin]
```

splitrom_x1.pl ist die Version f. Programmpaket X Version 1.0, 8/89 H.Fey und einem ROM (Bit0 zählt Banken hoch) splitrom_x2.pl ist für Paket X V2 mit 2 ROMs (Bit0 wählt den ROM aus) und splitrom_x2b.pl für Paket X V2 mit einem ROM ab C000 (Bit0 zählt Banken hoch).

neue Software

Aufgrund meiner Erfahrungen mit ROM-Modulen für Z9001, Z1013 und den TINY habe ich eine „Paket X“-Version programmiert, die nicht mehr angepasst werden muss, wenn Programme in der ROM-Bank ausgetauscht werden. Der Eprom „PAKET-X“ muss nur einmal programmiert werden. Die Zahl der erlaubten Programme hat sich auf 999 vergrößert. Programme können mit ZX7 optimal gepackt werden und verbrauchen dann ca.30% weniger Speicherplatz im ROM. Ein erster Test-ROM enthält über 300 Programme; und es ist noch Platz!

```

*** ROM - Disk Verwaltung VP 02/2023 ***

001 P CP/M-2.2          0000-FE00,E600 JE600
002 P HRCPM-12         0000-FE00,E600 JE600
003 P DVU-21           4000-531F,4000 d
004 P DVHD-31          4000-57EB,4000 d
005 P Gide-Check       2000-2FE7,2000 g
006 P EDAS-4           4000-544F,4000 J4000
007 P ASSEMBLER        4000-5447,4000 a
008 P COMPILER         5600-6000,5600 s
009 P DEBUGGER         4000-5861,4000 J4000
010 P DISASSEMBLER-3   5800-5FFF,5800 J5800
011 P RAM-TEST         2000-26BF,2000 g
012 P DISK-1F         3000-3BF1,3000 d
013 P OCEAC1           4000-5D44,4000 J4000
014 P FDC-iA           2000-27FF,0000
015 P minibasic        4000-46B4,4000 AC1 Minibasic
016 P AC1-TurboSaveLoad 1900-1C90,1900 g
017 P AC1-V24SaveLoad  1900-1B96,1900 g
018 P AC1-V24Testfool  1900-1BE8,1900 Auto
019 P Check+CPM-Umgebung 2000-25FF,0000 t
020 P DVHD3            4000-575E,4000 d,t
021 P DVU21            4000-531F,4000 d,t
022 P GIDE-CHECK       2000-2AD9,2000 g
023 P HRCPM12-AC2010   0000-FE00,E600 J E600
024 P HRCPM12-SCCH     0000-FE00,E600 J E600

Cr = weiter
Z = zurück
M = Monitor
oder Kennzahl ☐

```

s.o. [Downloads](#) oder github: <https://github.com/hcddr/ac1/tree/master/modul1/romdiskverwaltung-neu>

JKCEMU

der JKCEMU 0.9.8.2 emuliert das Paket X nur in der Modellversion AC1-SCCH. Es werden die ROM-Bänke 08,09..78,79 unterstützt, das ROM-Image darf maximal 256 kByte groß sein und muss die einzelnen ROM-Bänke in der Reihenfolge 08,09..78,79 enthalten (wie im originalen SCCH-Modul-1 mit einem ROM). Es sind 16k- und 32K-Bänke möglich.

Die beiden Dateien prog_x_e000_001.bin und x_rom_001.bin von http://www.ac1-info.de/ac1_scch/ac1_scch_baugruppen.htm können genutzt werden; die ROMs anderer Pakete müssen angepasst werden!

Die Emulation des 1MB-Modul-1 nach Hänke/Weidlich ist mit dem originalen JKCEMU nicht möglich. Man muss erst den Quellcode ändern:

- src\jkcemu\emusys\AC1.java: in writeIOByte 32 Bänke erlauben: `int bank = (value & 0x01) | ((value >> 3) & 0x0E) → ...((value >> 3) & 0x1E);`
- src\jkcemu\emusys\ac1_llc2\AbstractSCCHSys.java: in loadScchROMs die ROM-Größe anpassen: `this.scchRomdiskBytes = readROMFile(this.scchRomdiskFile, 0x40000, „SCCH-Modul 1 ROM-Disk“ ); → 0x100000 (1 MB)`
- Das ROM-Image muss aus den beiden ROM1 und ROM2 in 64K-Teilen verschachtelt werden: Reihenfolge 08,09..78,79,...,F8,F9. Oder man modifiziert den Zugriff in src\jkcemu\emusys\AC1.java in loadScchROMs, so dass beide ROMs einfach hintereinander kopiert werden können
- src\jkcemu\emusys\AC1.java: Hier sollte dem AC1 auch eine NMI-Taste gegönnt werden: `case KeyEvent.VK_F3: this.emuThread.getZ80CPU().fireNMI();`

Test-ROM

Verschiedene Varianten des Modul-1 blenden den ROM unterschiedlich ein. Um die Variante bei

unbekannter Hardware zu ermitteln, bietet sich ein Test-ROM an.

Der Test-ROM ist ein 1MB-ROM. Er ist fortlaufend in 16K große Teile (Bänke) zerlegt. In jeder Bank stehen die laufende Bank-Nr. und die Anfangsadresse der Bank im ROM.

Mit Out-Befehlen wird eine beliebige X-Bank aktiviert und geschaut, welcher Teil des ROMs aktiv ist:

```
AC 1 * MONITOR * U880 * (10/88)

# W 14 8
# D 8000
*
<8000> <H> 8000> 00 00 00 00 00 00 00 00 * ..... *
# W 14 9
# D 8000
*
<8000> <H> 8000> 02 00 00 00 00 00 00 00 * ..... *
# W 14 18
# D 8000
*
<8000> <H> 8000> 04 00 00 00 00 00 00 00 * ..... *
# W 14 19
# D 8000
*
<8000> <H> 8000> 06 00 00 00 00 00 00 00 * ..... *
# X
```

Hier erkennt man am letzten Beispiel, dass auf X-Port 19h der ROM-Bank-Teil Nr. 6 (hex) aktiviert wird. Dieser beginnt im ROM ab Adr. 00018000.

Bekannte Varianten der Ansteuerung

```
;-----
---
; Variante 1 laufende Banknummern liegen im ROM hintereinander (orig Modul1,
jkcemu)
; nächste Bank 08,09,18,19,...F8,F9
;-----
---
db      08h,09h,18h,19h,28h,29h,38h,39h
db      48h,49h,58h,59h,68h,69h,78h,79h
db      88h,89h,98h,99h,0A8h,0A9h,0B8h,0B9h
db      0C8h,0C9h,0D8h,0D9h,0E8h,0E9h,0F8h,0F9h

;-----
---
; Variante 2 Banknummern erst ROM1 x8 dann ROM2 x9 (1-MByte-Modul-1
AC1-2010, 2 ROMs)
; nächste Bank 08...F8, dann 09..F9
;-----
---
db      08h,18h,28h,38h,48h,58h,68h,78h
db      88h,98h,0A8h,0B8h,0C8h,0D8h,0E8h,0F8h
db      09h,19h,29h,39h,49h,59h,69h,79h
db      89h,99h,0A9h,0B9h,0C9h,0D9h,0E9h,0F9h
```

```
;-----  
--  
; Variante 3  
;-----  
---  
      db      08h,09h,28h,29h,88h,89h,0a8h,0a9h
```

From:

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

Permanent link:

<https://hc-ddr.hucki.net/wiki/doku.php/homecomputer/ac1/modul-1?rev=1786443314>

Last update: **2026/08/11 10:15**

