EPROM-Floppy unter ML-DOS

Softwareversion: M048_M049_ROM_1.5.0.exe 01.10.2022

Teil 1 - Modul M048 / M049 als EPROM-Floppy unter ML-DOS

Die Ergänzung des RAM-Floppy unter Verwendung eines EPROM-Moduls mit 256 kiByte wurde bereits in den **KC-News 02/1996** beschrieben.
Die Version 2.3 des Systemgenerierungsprogramms SYSGEN.COM für MLDOS unterstützt inzwischen zwei Modultypen, das Modul M048 mit 256 kiByte und das Modul M049 mit 512 kiByte. In der Voreinstellung von **SYSGEN23.COM** sind beide Modultypen aktiv. Bei Bedarf kann während der Systemgenerierung diese Einstellung auch verändert werden.

Nun noch ein paar Worte zum Erstellen der EPROMs. Zunächst sollte man sich 256 kiByte bzw. 512 kiByte der Dateien heraussuchen, die man am häufigsten benötigt und ständig verfügbar haben möchte. Die tatsächlichen Dateigrößen sind dabei auf volle 4ki-Gruppen aufzurunden, da bei RAM-Floppys ab 1 MiByte 4ki-Gruppen verwendet werden. Im EPROM-Modul soll ein Verzeichnis untergebracht werden, die letzte Datei im Modul muss deshalb mindestens 4 bzw. 8 freie Sektoren in der letzten 4ki-Gruppe enthalten.

Eine der zu verwendenden Dateien erfüllt sicher diese Bedingung und kann an das Ende des Moduls geschrieben werden. In diese 4 bzw. 8 Sektoren ist das Modulverzeichnis einzutragen, das vom Startprogramm gelesen und in das Verzeichnis des RAM-Floppys übertragen wird.

Das Modulverzeichnis wurde ähnlich dem Verzeichnis im CP/M-System gewählt, mit dem Unterschied, dass noch keine Blocknummern vergeben wurden. Die Blocknummern sind ja abhängig von der RAM-Floppy-Größe und vom Steckplatz des Moduls. Im Modulverzeichnis steht neben dem Dateinamen nur der relative Dateibeginn (zum Modulanfang) und die Dateigröße. Ein Verzeichniseintrag belegt 16 Byte, somit sind in den 4 Sektoren insgesamt 32 Einträge möglich und in den 8 Sektoren beim M049 64 Einträge.

Jeder Eintrag hat den folgenden Aufbau:

Byte 0	USER-Bereich der Datei (E5h, wenn nicht belegt!)
Byte 1–11	Dateiname (mit entsprechend gesetzten Attributen, siehe nächste Seite)
Byte 12	letzte Extentnummer (Anzahl 16 kiByte-Blöcke = 128 Sektoren a 128 Byte)
Byte 13,14	Startsektor im Modul
Byte 15	Anzahl Sektoren im letzten Extent

Auch bei Dateien größer als 16 kiByte ist nur ein Eintrag erforderlich, im Byte 12 steht die letzte Extentnummer und im Byte 15 wie viele Sektoren der letzte Extent belegt. Mit diesen Angaben berechnet das Startprogramm den vollständigen Directory-Eintrag für das Laufwerk „A:“. Als Beispiel hier die ersten vier Einträge eines Modulverzeichnisses:

```
00  41  53  4D  20  20  20  20  20  C3  CF  4D  01  00  00  20   .ASM_____COM...
00  43  4F  50  59  20  20  20  20  C3  CF  4D  00  A0  00  20   .COPY____COM...
00  44  55  20  20  20  20  20  20  C3  CF  4D  00  C0  00  50   .DU______COM...
00  4C  4E  4B  20  20  20  20  20  C3  CF  4D  00  20  01  60   .LINK____COM...
```

- Der erste Eintrag gehört der Datei ASM.COM, Extent 0 ist voll und belegt 128 Sektoren, im Extent 1 sind weitere 32 Sektoren. Insgesamt ist die Datei also 160 Sektoren groß und belegt genau 20 kiByte. Als letzte Datei ist diese somit nicht geeignet.
- Der zweite Eintrag gehört der Datei COPY.COM. Startsektor im Modul ist 00A0h, also direkt nach ASM.COM.
  Auch diese Datei ist mit 32 Sektoren und genau 4 kiByte nicht als letzte Datei geeignet.

- Die dritte Datei im Modul heißt DU.COM, beginnt im Sektor 00C0h im Modul und ist 80 Sektoren groß. Das entspricht einer Dateigröße von 10 kiByte, da aber auf volle 4kiByte-Gruppen aufgerundet werden muss, sind 12 kiByte zu reservieren. Wenn diese Datei am Modulende untergebracht würde, könnte hier das Modulverzeichnis eingetragen werden.
- Die vierte Datei LINK.COM beginnt also jetzt nach den vollen 12 kiByte, die für DU.COM reserviert wurden. Startsektor ist somit 0120h. Bei der Dateigröße mit 96 Sektoren und genau 12 kiByte wäre kein Platz für das Modulverzeichnis.

Dateien wie DU.COM, die bei 2K-Gruppen einen Block weniger belegen, also mehr als 15 freie Sektoren enthalten, lassen logischerweise einen freien Block im EPROM-Bereich offen.
Das EPROM-Modul sollte deshalb auf dem höchsten Steckplatz stecken, da ein Beschreiben dieser Blöcke nicht gesperrt ist.

Das auf **Seite 5** beschriebene Programm, welches die ROM-Datei automatisch erzeugt, legt aus diesem Grunde ein „DUMMY.ROM“ in Userbereich 15 an, wenn die Dateien nicht alle Sektoren belegen. Dies verhindert „freie Sektoren“ und für ML-DOS sind alle Sektoren belegt.

Alle Dateien des EPROM-Floppys befinden sich im USER-Bereich 0 und besitzen das R/O-, Public- und das SYS-Attribut. Das hat den Vorteil, dass die Dateien auch von anderen Laufwerken bzw. USER-Bereichen aufrufbar sind und nicht so einfach zu löschen sind.

Teil 2 - Dateiattribute im ZSDOS – ML-DOS

Dateiattribute werden von ZSDOS benutzt, um den Status von Dateien zwischen den BDOS Rufen zu kontrollieren.
Die Attribute sind in den höchstwertigen Bits (Bit 7) der Dateinamen (8 Bytes Name, 3 Bytes Typ) von FCB und Directoryeintrag enthalten. Folgende Bedeutungen sind definiert, beim EPROM-Floppy sind die farblich gekennzeichneten Attribute gesetzt:

FCB + 1 f1 (für Anwender verfügbar; Attribut „zuerst anordnen“ bei DEFRAG.com)
FCB + 2 öffentliche Datei
FCB + 3 kein Zugriffstempel
FCB + 4 f4 (für Anwender verfügbar; von ML-DOS / MicroDOS nicht genutzt)
FCB + 5 reserviert für interne Benutzung durch ZSDOS
FCB + 6 reserviert für interne Benutzung durch ZSDOS
FCB + 7 reserviert für interne Benutzung durch ZSDOS
FCB + 8 Wheel Schutz
FCB + 9 Nur-Lesen (Schreibschutz)
FCB + 10 Systemdatei
FCB + 11 archiviert

DUMMY.ROM – EPROM-Floppy-Leerdatei
Programme im EPROM-Floppy

Die ***BDOS-Funktion 30*** gestattet dem Programmierer das Setzen oder Löschen von Attributen durch Setzen oder Rücksetzen der entsprechenden Bits im FCB, dessen Adresse in DE übergeben wird. Im FCB dürfen Jokerzeichen enthalten sein.
Der Anwender darf die Attributbits, die für die interne Benutzung reserviert sind, **nicht verändern**. Bereits unter CP/M und ZRDOS waren diese Bits reserviert, so dass dies keine neue Einschränkung darstellt.

Wird das Attribut-Bit für **öffentliche Dateien** auf 1 gesetzt, dann kann diese Datei von jedem Nutzerbereich der gleichen Diskette gefunden werden, wenn eine eindeutige Dateiangabe zur Suche benutzt wird. Es liegt in der Verantwortung des Programmierers dafür zu sorgen, dass auf einer Diskette keine zweite Datei existiert, die mit einer öffentlichen Datei namensgleich ist.

Das Attribut **Wheel-Schutz** verhindert in einem Z-System, dass die Datei überschrieben, gelöscht, umbenannt oder eines ihre Attribute verändert wird, solange das Wheel-Byte nicht gesetzt ist. In einer Systemumgebung ohne ZCPR hat dieses Attribut im Allgemeinen keine Wirkung. ZSDOS geht dann davon aus, dass der Anwender alle Nutzungsrechte besitzt.

Mit dem Attribut **Nur-Lesen** wird uneingeschränkter Schutz vor dem Überschreiben, Löschen oder Umbenennen einer Datei erreicht.

Das **System-Attribut** hat zwei Funktionen. Zum einen werden Systemdateien von den meisten Directoryprogrammen nicht angezeigt und bleiben dem Anwender dadurch „verborgen“. Zum anderen werden derartige Dateien beim Öffnen unter ZSDOS entlang des Pfades gefunden, wenn der Pfad-Dateizugriff aktiv ist.

Das **Archiv-Attribut** zeigt an, ob eine Datei verändert wurde. Dieses Bit muss von einem Anwendungsprogramm gesetzt werden (typische Vertreter sind Programme für Sicherheitskopien). Wird in die Datei geschrieben, dann löscht ZSDOS das Archiv-Attribut des ersten Extents.

Siehe: **ZSDOS 1.1** Programmer's Manual

Maßeinheiten für Speicherdaten nach „IBM Spectrum Control“

https://www.ibm.com/docs/en/spectrum-control/5.4.2?topic=concepts-units-measurement-storage-data

Teil 3 - Aufbau Module M048 / M049

Modul M048 als EPROM-Floppy

Strukturbyte 73h
Steuerbyte = AASSSSxM
16 Blöcke zu je 16 kiByte

Blockgröße **4 kiByte** für Programme, immer volle Blockgröße verwenden!

Verzeichnisaufbau: 4 Sektoren je 128 Byte → 512 Byte **200h**

Aufbau EPROM:	**00000h**	Programme / Dateien
	3FE00h	Beginn der max. 32 Verzeichniseinträge
	3FFFFh	Ende Speicher

Modul M049 als EPROM-Floppy

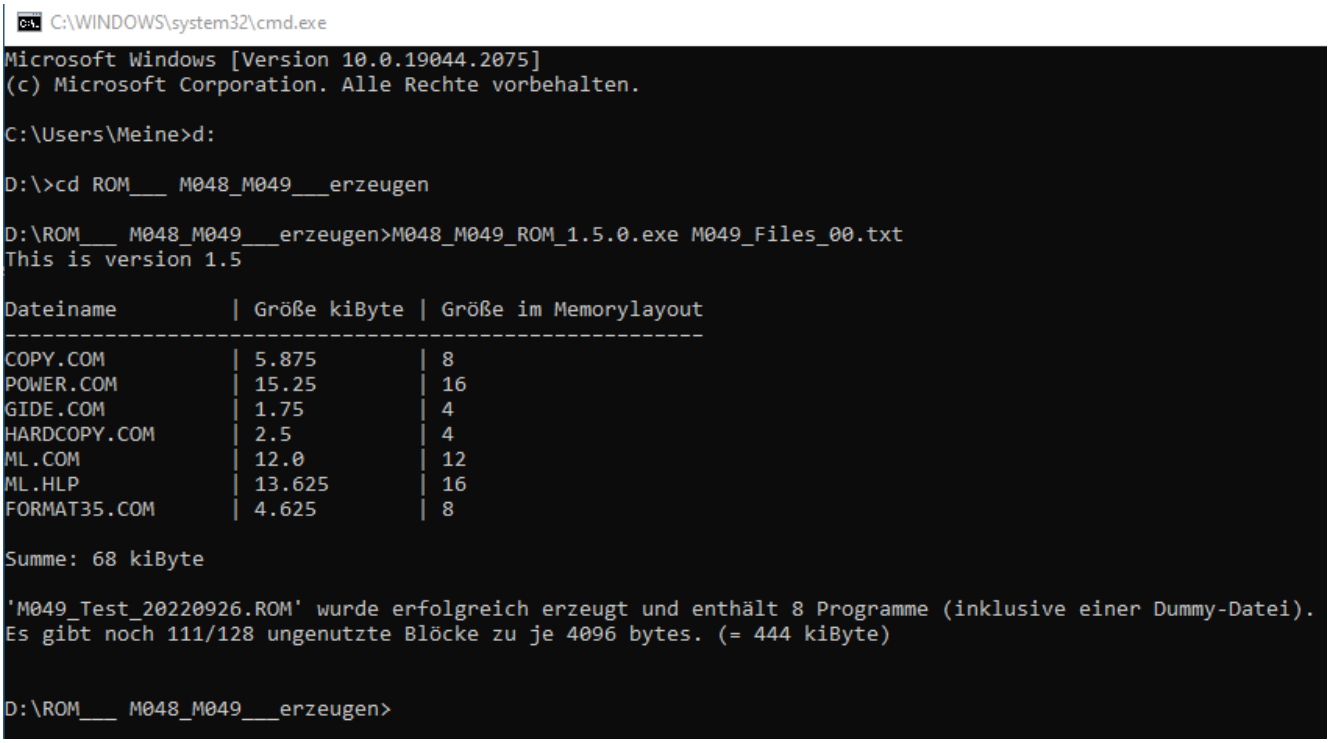
Strukturbyte 74h
Steuerbyte = AASSSSSM
32 Blöcke zu je 16 kiByte

Blockgröße **4 kiByte** für Programme, immer volle Blockgröße verwenden!

Verzeichnisaufbau: 8 Sektoren je 128 Byte ⟶ 1024 Byte **400h**

Aufbau EPROM:	**00000h**	Programme / Dateien
	7FC00h	Beginn der max. 64 Verzeichniseinträge
	7FFFFh	Ende Speicher

Teil 4 - EPROM-Floppy - ROM-Datei für M048 / M049 erzeugen



Die Erzeugung der ROM-Datei erfolgt auf Konsolenebene.

Zur Erzeugung der ROM-Datei werden zwei Dateien benötigt:

- M048_M049_ROM_1.5.0.exe — Hauptprogramm
- M049_Files_00.txt — Textdatei

Das Hauptprogramm dient zur Erzeugung der ROM-Datei für das M048 und M049.
Es liest die Textdatei ein (Name kann frei festgelegt werden) und erstellt aus dem Inhalt die ROM-Datei, abhängig vom Inhalt.

Aufbau Textdatei:

M049_Test_20220926

Dateiname der ROM-Datei mit Unterscheidung:
M048 = 256 kiByte **M049** = 512 kiByte

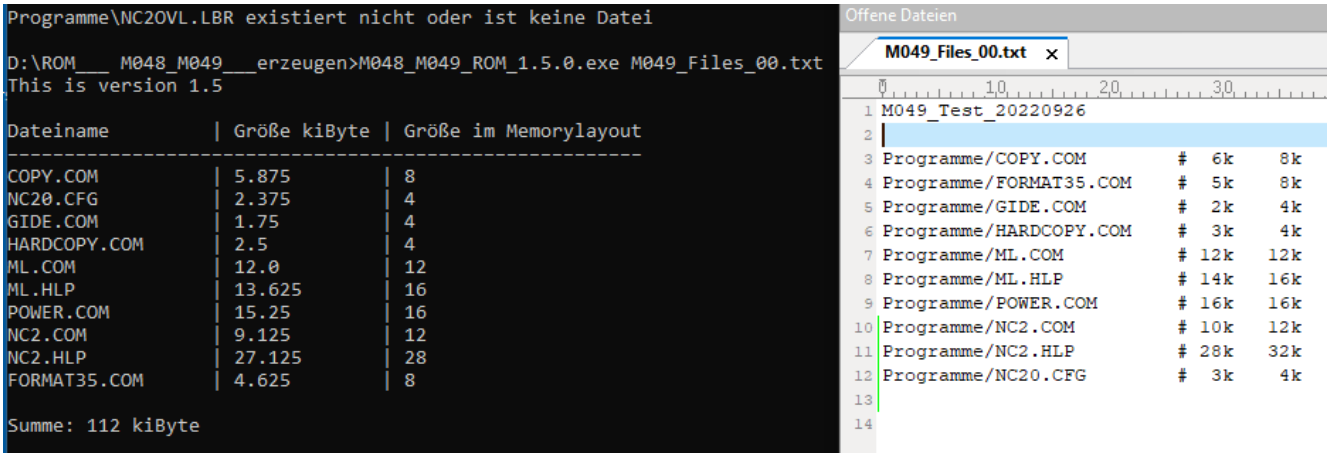
```
Programme/COPY.COM          #  6k   8k
Programme/FORMAT35.COM      #  5k   8k
Programme/GIDE.COM          #  2k   4k
Programme/HARDCOPY.COM      #  3k   4k
Programme/ML.COM            # 12k  12k
Programme/ML.HLP            # 14k  16k
Programme/POWER.COM         # 16k  16k
#Programme/NC2.COM          # 10k  12k
#Programme/NC2.HLP          # 28k  32k
#Programme/NC20.CFG         #  3k   4k
```

Leerzeilen werden nicht beachtet und alles ab „#“ wird als Kommentar angesehen. Damit können auch ganze Zeilen von der Abarbeitung ausgeschlossen werden.

Name der erzeugten ROM-Datei **= M049_Test_20220926**

Das Programm „KC85_M048_M049...“ erzeugt eine ROM-Datei aus den in der Textdatei angegebenen Dateien. Dabei wird automatisch das ROM-Verzeichnis am Ende der ROM-Datei erstellt. Falls die 256/512 kiByte von den Dateien nicht komplett ausgenutzt werden, wird der verbleibende Speicherplatz (Sektoren) in einer Datei „DUMMY.ROM“ im Userbereich 15 eingetragen und dieser Speicher damit als belegt markiert.

Teil 5 - Technische Hintergründe für die Auswahl der Directory-Datei:



```
Programme\NC2OVL.LBR existiert nicht oder ist keine Datei

D:\ROM___ M048_M049___erzeugen>M048_M049_ROM_1.5.0.exe M049_Files_00.txt
This is version 1.5

Dateiname         | Größe kiByte | Größe im Memorylayout
-------------------------------------------------------
COPY.COM          | 5.875        | 8
NC20.CFG          | 2.375        | 4
GIDE.COM          | 1.75         | 4
HARDCOPY.COM      | 2.5          | 4
ML.COM            | 12.0         | 12
ML.HLP            | 13.625       | 16
POWER.COM         | 15.25        | 16
NC2.COM           | 9.125        | 12
NC2.HLP           | 27.125       | 28
FORMAT35.COM      | 4.625        | 8

Summe: 112 kiByte
```

Offene Dateien

M049_Files_00.txt

```
1  M049_Test_20220926
2
3  Programme/COPY.COM          #  6k    8k
4  Programme/FORMAT35.COM      #  5k    8k
5  Programme/GIDE.COM          #  2k    4k
6  Programme/HARDCOPY.COM      #  3k    4k
7  Programme/ML.COM            # 12k   12k
8  Programme/ML.HLP            # 14k   16k
9  Programme/POWER.COM         # 16k   16k
10 Programme/NC2.COM           # 10k   12k
11 Programme/NC2.HLP           # 28k   32k
12 Programme/NC20.CFG          #  3k    4k
13
14
```

Die Textdatei wird nur einmal abgearbeitet und davon ausgegangen, dass es maximal 32/64 Dateien gibt. Aus diesem Grund wird sofort nach einem geeigneten Block (4 kiByte) von einer Datei gesucht, wo das Directory noch Platz hat.
Freier Speicherplatz entsteht nur, wenn weniger als 256/512 kiByte an Daten vorhanden sind. Das kann auch bei 32/64 Dateien der Fall sein, dann wird eine DUMMY.ROM benötigt.
In diesem speziellen Fall können nur 31/63 Dateien verwendet werden und der Benutzer wird aufgefordert, eine Datei zu entfernen.

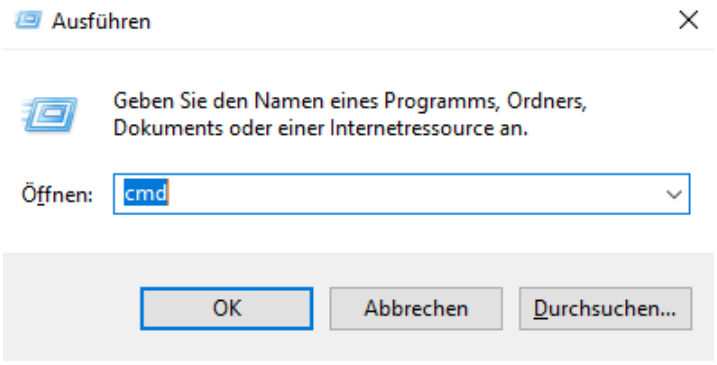
Die Größe wird über „Modulo 4“ Block (4 kiByte) bestimmt und die passende Datei am Ende der Abarbeitung mit der letzte Datei getauscht. Der Austausch erfolgt nur intern und nach außen hat das Directory die gleiche Reihenfolge wie in der Textdatei. Es wird die letzte Datei für das Directory genommen, die sich bei der Suche ergibt. Wird keine gültige Datei gefunden und es werden die vollen 256/512 kiByte an Daten benötigt, bricht das Programm mit einer Fehlermeldung ab.

In dem Beispiel oben ist es „Format35.com“, welches die Kriterien für das Directory erfüllt. Eine Überprüfung, ob wirklich M048 = 512 Byte bzw. M049 = 1024 Byte frei sind, erfolgt beim Schreiben der Directory in die ROM-Datei.

Teil 6 - Fehlerausgaben:

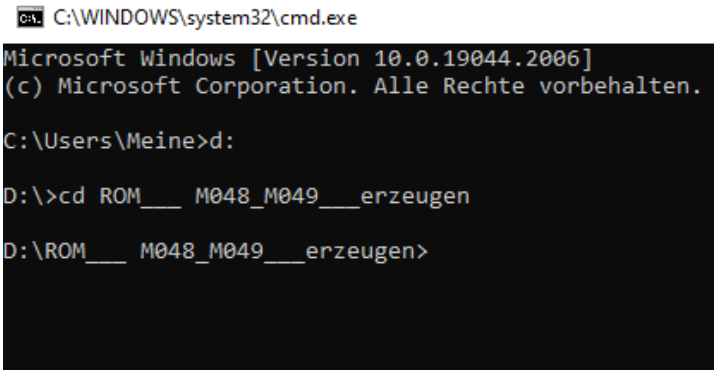
- Zu viele Dateien M048 > 32 M049 > 64
- Dateien belegen zu viel Speicherplatz M048 > 256 kiByte M049 > 512 kiByte
- Maximale Dateienanzahl (32/64) verwendet, belegt aber nicht den gesamten Speicher, es kann keine „DUMMY.ROM“ angelegt werden
- Es wird der gesamte Speicherplatz verwendet, es existiert aber keine Datei für das Directory
- Textdatei enthält keine Daten
- M048 / M049 nicht erkannt – keine Ausgabedatei definiert
- Keine Datei für Directory gefunden
- Angegebene Datei existiert nicht
- Dateiname mehrfach in der Textdatei
- Dateiname + Erweiterung – nur die ersten 128 ASCII-Zeichen sind erlaubt
- Dateiname mehr als 8 Zeichen
- Erweiterung mehr als 3 Zeichen

Teil 7 - EPROM-Floppy - Arbeitsgänge ROM-Datei erzeugen

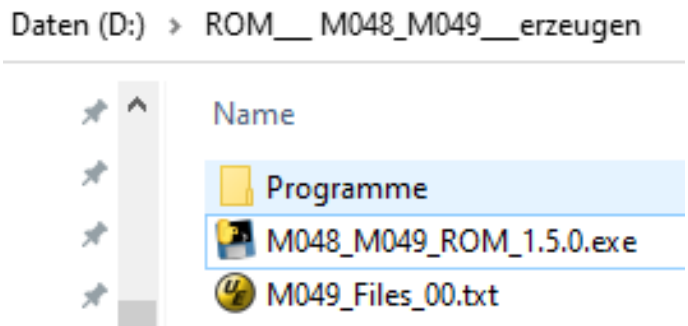


Windows Konsole aufrufen,
„Win-Taste + R“ und „CMD“ ausführen.

in das korrekte Laufwerk wechseln



mit „cd“ das Arbeitsverzeichnis auswählen



Aufbau der Verzeichnisstruktur.

Wegen der Übersicht liegen alle Daten welche in die ROM-Datei sollen im Verzeichnis „Programme“.

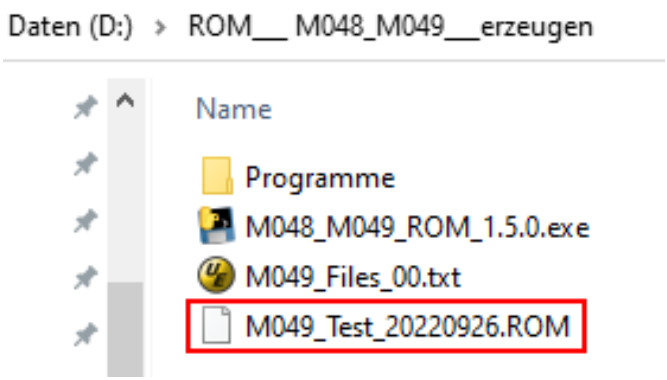
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2075]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Meine>d:

D:\>cd ROM___ M048_M049___erzeugen

D:\ROM___ M048_M049___erzeugen>M048_M049_ROM_1.5.0.exe M049_Files_00.txt
This is version 1.5

Dateiname         | Größe kiByte | Größe im Memorylayout
------------------------------------------------------
COPY.COM          | 5.875        | 8
POWER.COM         | 15.25        | 16
GIDE.COM          | 1.75         | 4
HARDCOPY.COM      | 2.5          | 4
ML.COM            | 12.0         | 12
ML.HLP            | 13.625       | 16
FORMAT35.COM      | 4.625        | 8

Summe: 68 kiByte

'M049_Test_20220926.ROM' wurde erfolgreich erzeugt und enthält 8 Programme (inklusive einer Dummy-Datei).
Es gibt noch 111/128 ungenutzte Blöcke zu je 4096 bytes. (= 444 kiByte)


D:\ROM___ M048_M049___erzeugen>
```

Ausführung vom Programm

```
M049_Files_00.txt
1  M049_Test_20220926
2
3  Programme/COPY.COM          #   6k    8k
4  Programme/FORMAT35.COM      #   5k    8k
5  Programme/GIDE.COM          #   2k    4k
6  Programme/HARDCOPY.COM      #   3k    4k
7  Programme/ML.COM            #  12k   12k
8  Programme/ML.HLP            #  14k   16k
9  Programme/POWER.COM         #  16k   16k
10 #Programme/NC2.COM          #  10k   12k
11 #Programme/NC2.HLP          #  28k   32k
12 #Programme/NC20.CFG         #   3k    4k
```

Inhalt der Textdatei

Die hier gewählte Reihenfolge der Dateien entspricht auch der Reihenfolge in der Directory.



Die erzeugte ROM-Datei liegt dann im Arbeitsverzeichnis und kann ins Modul übertragen werden.