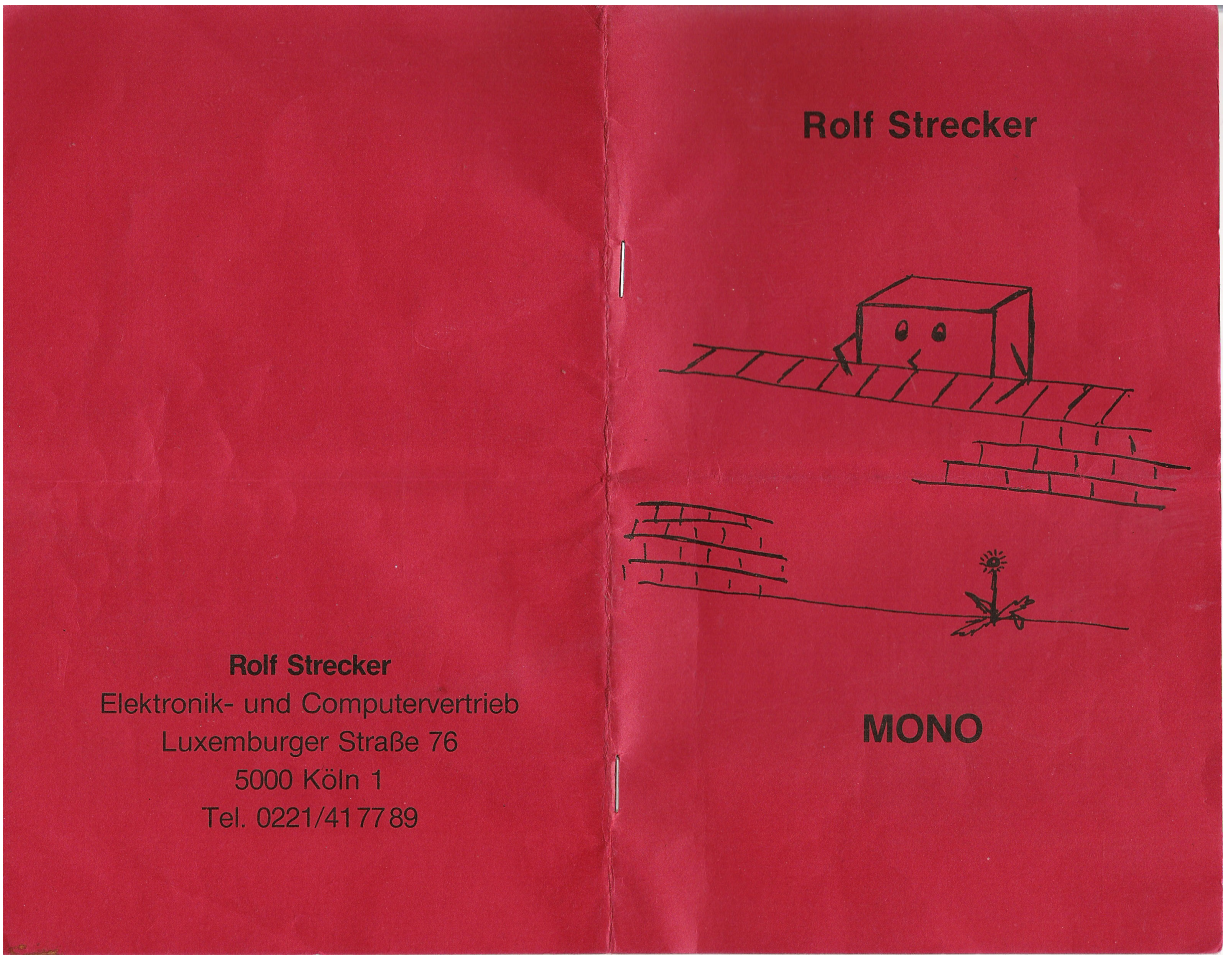
MONO

2005 OCRed by Wilko Schröter

Rolf Strecker

Elektronik- und Computervertrieb
Luxemburger Straße 76
5000 Köln 1
Tel. 0221/41 77 89



Sehr geehrter Kunde,

mit dem vorliegenden Programm haben Sie ein Produkt aus unserem reichhaltigen Angebot erhalten.
Wie Sie sehr schnell feststellen werden, ist dieser Macro-Assembler ein unentbehrliches Werkzeug für jeden ernsthaften Programmierer.
Als SPECTRUM-Besitzer sollten Sie sich jedoch hiermit nicht zufrieden geben.
Die einzelnen Programme aus unserem Angebot zu besprechen, würde den Rahmen dieser Beschreibung sprengen. Daher verweisen wir Sie an dieser Stelle auf unseren Katalog, der ständig aktualisiert wird. Gegen eine Schutzgebühr von DM 3,80 können Sie diesen bei uns anfordern. In der Anlage zu diesem Programm erhalten Sie 2 Kurzbeschreibungen von SPECTRUM Utilities, die ebenfalls für den ernsthaften Benutzer geschrieben wurden.
In eigener Sache möchten wir noch hinzufügen, daß die Beschreibung mit TASWORD II, dem Gemini 10X Drucker und dem SPECTRUM erstellt wurde.

Wir weisen Sie darauf hin, daß Sie dieses Programm nur für den eigenen Bedarf kopieren dürfen. Der Verleih, das Kopieren sowie Vermieten dieses Produktes ist aus urheberrechtlichen Gründen strengstens verboten. Wir werden gegen jeden uns bekannten Schwarzkopierer gerichtlich vorgehen.

Köln, den 28.11.1984

MONO

(c)1984 by Martin Gergeleit

MONO ist ein leistungsfähiger Macroassembler/Disassembler für den 48K Spectrum. Das Programm belegt den Speicherbereich von A000h bis FFFFh und wird mit LOAD "Mono" oder LOAD "" von der Cassette geladen. MONO startet selbständig, kann jedoch auch später immer wieder mit RANDOMIZE USR 42421 ohne Programm- oder Datenverlust neu gestartet werden. Das Programm besitzt einen vollständigen Bildschirmeditor, der alle 24 Zeilen des ZX Spectrum ausnutzt. Der Editor kennt folgende Befehle:

TRUE VIDEO	Bildschirm löschen
INV. VIDEO	Cursor auf erste Bildschirmposition
Pfeiltasten	Cursorsteuerung
GRAPHICS	Einsetzen einer Zeichenstelle
DELETE	Löschen einer Zeichenstelle
ENTER	Übergabe einer Zeile zur Bearbeitung
CAPS + SPACE	Abbruch aller Rechnungen und Rückkehr zum MONO-Editor
CAPS + SPACE + SYMBOL	Abbruch aller Rechnungen und Rückkehr zum Basic

Jede andere Taste bringt das auf ihr angegebene Zeichen auf den Bildschirm. Dabei können zwar alle Einzelzeichen, die über SYMBOL SHIFT angesprochen werden, benutzt werden, nicht jedoch die Basic-Keywords und die Kleinbuchstaben. Mit diesem Editor können nun alle MONO-Kommandos eingegeben werden. Wird eine Eingabe nicht als Kommando erkannt, so führt dies zu einem SYNTAX ERROR. Ebenso verursacht eine Adresse, die nicht aus vier Hex-Ziffern besteht, eine Fehlermeldung, in diesem Fall wird ein PARAMETER-ERROR gemeldet. Überhaupt sind alle in MONO verwendete Zahlen Hexadezimale Größen.

MONO kennt die folgenden Kommandos:

EXIT
MQNO wird verlassen und der Spectrum kehrt zum Basic zurück. EXIT bewirkt also dasselbe wie der Druck auf die drei Tasten CAPS, SPACE und SYMBOL.

CALL >Adresse<
Die Hauptregister der Z80-CPU werden mit den gespeicherten Werten geladen; dann wird die Routine mit der angegebenen Adresse angesprungen. Der Aufruf entspricht einem CALL der Maschinensprache, d. h. eine Rückkehr zu MONO ist mit dem Z80-Befehl RET möglich. Bei der Rückkehr werden die Registerinhalte wieder gespeichert. Sofern die Routine nicht die Interrupts verbietet oder ändert, ist auch hier immer ein sofortiger Programmabbruch mit CAPS und SPACE mög1ich, was zur Meldung RESET IN ADDRESS X führt, wobei X die Abbruchadresse darstellt. Innerhalb einer Maschinencoderoutine dürfen alle Z80-Register bis auf die Alternativregister verändert werden, also auch die IX- und IY-Register. Aber Vorsicht: Im Basic darf keine Routine aufgerufen werden, die mit eingeschalteten Interrupts das IY-Register ändert, weil dies mit Sicherheit zum Absturz führt.

REG
Die Inhalte der Z80-Register A, BC, DE und HL werden angezeigt. Die angezeigten Registerinhalte entsprechen dem Stand nach dem letzten CALL-Kommando. Es ist nun möglich, die Inhalte zu verändern, indem man mit dem Cursor in die entsprechende Zeile wandert, die dort angegebene Hex-Zahl ändert und dann die Zeile mit ENTER abschließt. Eine solche Zeile kann auch einzeln eingegeben werden.

DUMP >Adresse<
erstellt einen Hex-Dump ab der angegebenen Adresse auf dem Bildschirm. Es werden immer acht Zeilen auf einmal ausgegeben, dann wartet MONO eine Eingabe ab. Ist diese ‚Eingabe ENTER, so wird ein weiterer Achterblock ausgegeben, sonst wird mit der Meldung READY abgebrochen. Die von DUMP erzeugten Zeilen beginnen mit einem ">"-Zeichen. Wird nun eine Zeile, die mit ">" beginnt mit ENTER übernommen, so wird dies als eine hexadezimale Eingabe verstanden. Dabei muß die erste Zahl eine Adresse angeben. Die folgenden 8-Bit-Zahlen werden dann als Bytes in diese Adresse und die folgenden eingeschrieben. Eine solche Hex-Eingabezeile kann also auch unabhängig vom DUMP-Kommando erstellt werden. Das bedeutet, daß Sie auch über das DUMP Werte verändern können.

LDUMP >Adresse<
erstellt wie DUMP ein Hex-Listing, nur wird dieses auf dem Drucker ausgegeben.

DUMP ASC >Adresse<
gibt auf dem Bildschirm eine Ascii-Dump des Speicherinhaltes aus. In diesem DUMP ist es nicht möglich, mit dem Cursor Änderungen vorzunehmen. Die Ausgabe erfolgt, wie üblich in Achterblöcken.

LDUMP ASC >Adresse<
erstellt einen Ascii-Dump auf einem angeschlossenen Drucker.

MOV >Von Adresse< >Bis Adresse< >Zur Adresse<
verschiebt einen Speicherblock, der von der ersten Adresse bis zur zweiten reicht, zur dritten Adresse. Dabei werden Überlappungen berücksichtigt und somit Fehler vermieden.

DIS >Adresse<
erstellt ein Disassemblerlisting ab der angegebenen Adresse auf dem Bildschirm. Auch hier erfolgt der Ausdruck wieder in Achterblöcken. Da im ganzen System nicht zwischen Macros und original Z80-Befehlen unterschieden wird, übersetzt auch der Disassembler eine Bytegruppe, die ein Macro darstellen kann, nicht als eine Zusammensetzung aus Z80-Befehlen, sondern als einen Macrobefehl. Aufgrund dieser Tatsache wurde im Disassembler auf eine Ausgabe der Bytes im Hexformat verzichtet, da dies bei einem Macro von bis zu 255 Bytes Länge zu einem sehr unübersichtlichen Listing führen würde.

LDIS >Adresse<
erstellt wie DIS ein Disassemblerlisting, das jedoch auf dem Drucker ausgegeben wird.

LIST (>Anfangszeile<)
erstellt ein Listing des augenblicklich im Arbeitsspeicher befindlichen Programms, das entweder mit der angegebenen Zeile oder, wenn keine Angabe hinzugefügt wurde, mit dem Programmanfang beginnt. Der Ausdruck erfolgt ebenfalls in Achterblöcken. Auch hier können natürlich die angezeigten Programmzeilen mit dem Bildschirmeditor verändert werden, da MONO alle Zeileneingaben, die mit einer gültigen Zeilennummer beginnen, als Programmzeile interpretiert. Als solche werden sie in den Speicher einsortiert, wobei eine neue Zeile mit einer bereits vorhandenen Zeilennummer die alte Zeile überschreibt.

LLIST (>Anfangszeile<)
erzeugt ein Programmlisting auf dem Drucker.

ADIS >Anfangsadresse< >Endadresse< >Anfangszeile< >Abstand<
ist ein sehr mächtiges Kommando, das den Speicherbereich, der durch die Anfangs- und Endadresse angegeben wird, disassembliert und den erstellten Text in Programmzeilen ablegt. Die Nummern der Zeilen, die erzeugt werden, beginnen mit der angegebenen Anfangszeilennummer und steigen mit dem im Kommando definierten Abstand. Wenn bereits ein Programmteil im Speicher vorhanden ist, so können die Zeilennummern so gewählt werden, daß der neue Programmteil den alten nicht überschreibt.

SAVE >Name< (>Anfangsadresse<) (>Endadresse<)
speichert im Spectrum-Format den angegebenen Speicherblock unter dem angegebenen Namen auf Band ab. Wird weder End- noch Anfangsadresse angegeben, so speichert sich MONO selbst mit Programm und Macros ab.

LOAD >Name< (>Anfangsadresse<) (>Endadresse<)
lädt das Programm mit dem angegebenen Namen. Werden sonst keine Angaben gegeben, legt MONO das Programm in dem Bereich ab, aus dem es gesichert wurde. Ein Name muß angegeben werden. MONO ist in der Lage eine abgespeicherte MONOversion über sich selbst zu laden.

VERIFY >Name< (>Anfangsadresse<) (>Endadresse<)
bewirkt dasselbe wie LOAD, nur werden die Speicherblöcke hier miteinander verglichen.

ASS >Anfangsadresse<s
ist das wichtigste Kommando. Es assembliert den Text des Listings in Z80-Machinencode. Wenn im Programmtext nicht anders angegeben, so wird der fertige Code ab der angegebenen Adresse abgelegt. Ist der Code an dieser Stelle im Speicher lauffähig, so kann er nach dem Assemblieren mit CALL >Adresse< gestartet werden.

(L)LIST MACRO
gibt eine Liste aller bekannten Befehle aus, wobei die zuletzt definierten als erste angezeigt werden.

NEW
löscht das gesamte Programm.

OFFSET >Differenz<
bewirkt, daß alle folgenden ASS-, DIS-, LDIS- und ADIS-Kommandos sich nicht auf den angegebenen Speicherbereich, sondern auf den Bereich beziehen, der sich ergibt, wenn man die Anfangsadresse und die Differenz addiert. Das hat folgenden Sinn: Wenn ein Programm den Bereich belegt, in dem MONO oder das ROM liegen, so kann es in den freien Speicherbereich gelegt werden, aber behandelt werden, als ob es im Originalbereich läge. Dazu geben Sie als Differenz die wirkliche Adresse minus der Originalanfangsadresse an (z. B. Org.:F000, Wirk.:8000, Diff.:9000). OFFSET 0000 stellt den normalen Zustand wieder her. Vorsicht: Auch wenn das Disassemblerlisting bei einem Offset <> 0000 ein lauffähiges Programm anzeigt, so darf es auf keinen Fall mit CALL aufgerufen werden, denn es liegt nicht im richtigen Speicherbereich.

Im Programmtext sind nun folgende Statements möglich:

ORG >Adresse<
ändert die laufende Position des Assemblerpointers auf die angegebene Adresse. Der Assemblerpointer gibt die Adresse an, in die der nächste Maschinenbefehl eingeschrieben wird. Es ist auf diese Weise möglich, verschiedene Programmteile an unterschiedlichen Stellen im Speicher abzulegen. Aber Vorsicht: mit ORG können Sie auch schon dort abgelegte Programmteile wieder überschreiben.

DEF >Byte< ... >Byte<
legt Bytes an der laufenden Pointerposition ab, auch wenn diese keinen sinnvollen Befehl ergeben (z. B. Variable oder Texte)

MACRO >Name<
ist ein Befehl, der es Ihnen erlaubt, Ihre eigenen Mnemonics zu definieren; d. h. Sie können einem Namen eine Sequenz von bis zu 255 Bytes zuweisen. Wenn Sie dann im folgenden Programmtext diesen Namen benutzen, so setzt MONO dafür, wie bei einem Zilog-Mnemonic die definierte Sequenz ein. Ein Macro ist also kein Unterprogrammaufruf, sondern ein Befehl, den Sie nach ihren eigenen Bedürfnissen definieren können. Durch die Benutzung von Macros ist es möglich, kürzere und übersichtlichere, aber trotzdem schnelle Maschinenprogramme zu erstellen. Macros werden von MONO in einem gesonderten Speicherbereich erstellt, sind also unabhängig von ORG oder dem sonstigen Code im Speicher. Einem Macro können auch Parameter übergeben werden. Ein Macro wird nun folgendermaßen definiert: In einer Zeile wird der Befehl Macro, gefolgt vom Namen des neuen Macros angegeben. In den folgenden Zeilen schreiben Sie mit bisherigen Befehlen (im Grunde alles Macros für MONO) den Code, den Ihr Macro nachher ersetzen soll. Mit einem END-Statement wird die Definition beendet. Im Code des Macros sind auch DEF-Statements zulässig. Nicht sinnvoll sind in einem Macro direkt adressierte Sprünge innerhalb des Macros und ORG-Statements. Es ist auch nicht möglich ein Macro rekursiv zu definieren, also ein Macro in seiner eigenen Definition zu benutzen. Parameter werden vom Namen in den Code übergeben, d. h. beim Assemblieren eingesetzt, indem Sie im Namen und im Text der Definition Symbole verwenden. Ein >#< steht für einen Ein-Byte-Parameter, ein >%< symbolisiert eine Zwei-Byte-Zahl und ein >$< steht für eine relative Sprungadresse. Erscheinen zwei Parameter des gleichen Typs im Namen des Macros, so werden sie der Reihe nach eingesetzt. Falls Sie noch Fragen zur Macrodefinition haben, so sehen Sie sich bitte die Beispielprogramme an.

END
beendet, wie schon erwähnt, die Definition eines Macros.

Ein-Byte-Parameter können nur direkt angegeben werden, was bei MONO bedeutet, daß Sie ein ">"-Zeichen vor der zweistelligen Hexzahl angeben müssen (z. B. LD A,>FF). Zwei-Byte-Parameter können direkt oder mit einer Zeilennummer, die eine Art Label darstellt, angegeben werden. Die Zeilennummern stehen dabei für die Zwei-Byte-Adresse, ab der ihr Code eingeschrieben wird. Sie werden durch ein "L" gekennzeichnet (z. B. JP L1000, LD HL,(L10), JP >6000). Relative Sprünge können in allen drei Arten angegeben werden, (z. B. JR >02, JR L1000, JR >6000).

Zeilen, die mit ";" beginnen, werden als Kommentarzeilen verstanden und beim Assemblieren übersprungen.

Fehlermeldungen:	Situation:
READY	Kommando wurde erfolgreich ausgeführt.
NO PRINTER	Es ist kein Drucker angeschlossen (gilt nur für Sinclair-Drucker).
SYNTAX ERROR	Zeile kann nicht ausgeführt werden.
RESET IN ADDRESS X	>CAPS< und >SPACE< wurden gedrückt.
PARAMETER ERROR	Ein Parameter hatte nicht die richtige Anzahl an Hexziffern.
MEMORY OVERFLOW	Der Arbeitsspeicher ist voll.
BREAK IN CASETTE-OPERATION	Während SAVE, LOAD oder VERIFY wurde >SPACE< gedrückt.
TAPE LOADING ERROR	Es gab einen Lesefehler.
UNDEFINED MACRO	Das Macro wurde so nicht definiert.
BAD LOCATION	Der Code würde Mono überschreiben.
NO END-STATEMENT	Ein Macro wurde nicht mit END beendet.
NO MACRO-STATEMENT	Ein END zuviel im Programmtext.
MACRO TOO LONG	Ein Macro sollte mehr als 255 Bytes Symbolisieren.
JR IMPOSSIBLE	Ein relativer Sprung sollte mehr als 128 Bytes weit springen.
UNDEFINED PARAMETER	Ein Macroparameter wurde nicht benutzt.
BREAK IN ADDRESS X	Das Break-Macro wurde ausgeführt.

Programmbeispiele:

1)	10 MACRO EX	Der Name des Macros ist: EX BC,HL.
	20 PUSH BC	
	30 PUSH HL	Der Befehl setzt sich aus vier Elemen-
	40 POP BC	ten zusammen.
	50 POP HL	
	60 END	Die Definition ist abgeschlossen.
	ASS 8000	Das Programm wird erfolgreich assembliert
	READY	und Mono kennt nun EX BC,HL.

Auch wenn dieses Programm wieder gelöscht wird, bleibt das Macro erhalten. Es kann nur folgendermaßen gelöscht werden:

1a)	10 MACRO EX BC,HL	Hier steht wieder der Name des Macros.
	20 END	Das Macro wird neu, ohne Code definiert.
	ASS 8000	Die Definition wird in den Speicher
	READY	übernommen. Das Macro wurde gelöscht.

2)
```
10 MACRO EX BC,DE
20 PUSH BC
30 PUSH DE
40 POP BC
50 POP DE
60 END
70 EX BC,DE
ASS 8000
READY
```

Macros können auch umdefiniert werden, indem einfach der gleiche Name mit einem neuen Code eingegeben wird. Parameter werden folgendermaßen übergeben:

3)
```
10 MACRO LD (%),#
20 PUSH HL
30 LD HL,%
40 LD (HL) ,#
50 POP HL
60 END
70 LD (>9000),>FF
ASS 8000
READY
```

Das Macro kann eine beliebige Adresse direkt laden.
HL wird mit dem Parameter aus dem Namen geladen. Der Inhalt von HL wird mit der Ein-Byte-Zahl aus dem Namen geladen.
HL wird kurz auf den Stapel gerettet.
Das Macro wird im Programm benutzt.
Das Programm wird assembliert

Auf die gleiche Art können Sie z. B. zwei-Byte Subtraktion OHNE Carry für alle Register definieren:

4)
```
10 MACRO SUB HL,DE
20 XOR A
30 SBC HL,DE
40 END
ASS 8000
READY
```

Weitere Programme aus dem Hause STRECKER KÖLN

M U L T I F I L E

für Spectrum 48k
Dieses hervorragende Adressenverwaltungsprogramm bringt den SPECTRUM auf das Niveau von Personalcomputersystemen.
Durch vollständige Programmierung in Maschinensprache ist es erstmals möglich, eine Maske (natürlich mit 64 Zeichen und deutschen Umlauten) auf dem SPECTRUM ständig zu beschreiben.
Das heißt, jegliche Veränderungen, Kommandos und Eintragungen erfolgen in dieser Maske. Als Features enthält dieses Programm unter anderem:

- Komplette Verwaltung von Adressbeständen
- Ein mit TASWORD II erstellter Brief kann in MULTIFILE eingeladen und wieder abgespeichert werden
- Ausdruck von Rundbriefen bzw. wahlfreier Ausdruck nach vorgegebenen Suchkriterien (mit Einblendung der Anschrift und der Anrede. Definition von Damen, Herren und Damen und Herren für Firmen möglich)
- Sortieren nach Namen, Postleitzahlen oder nach vorgegebenen Kennziffern
- Laden und speichern der Adressbestände, mit Übernahme aller Informationen wie Dateinummer, Anzahl der belegten und freien Files
- Anzeigen des in MULTIFILE enthaltenen Briefes
- Microdrive und Floppykompatibel

MULTIFILE – Das absolute Novum der Adressverwaltung

Bestell.-Nr.: S 0269

M U L T I L A G E

für Spectrum 48k
Das Programm MULTILAGE ist eine Kombination aus Artikel- und Lagerverwaltung. MULTILAGE wurde vom gleichen Autor wie MULTIFILE geschrieben. Aus diesem Grund ist die Syntax der beiden Programme gleich: Es wird ebenfalls mit einer 64 Zeichen Bildschirmmaske gearbeitet.
MULTILAGE enthält unter anderem:

- Komplette Verwaltung von Artikelbeständen
- Ausdruck von Artikel- und Inventurlisten (wahlweise Drucker oder Bildschirm)
- Artikel werden bei der Artikelliste mit ihrem VK ausgegeben. Bei Erstellung einer Inventurliste wird der EK ausgeworfen. Der Gesamtinventurwert wird berechnet
- Separate Liste für Artikel die den Mindestbestand unterschreiten
- Positive und negative Bewegungen werden erfaßt und täglich ausgegeben
- Microdrive und Floppykompatibel

Bestell.-Nr.: S 0278