Monitor

Der Monitor des Amateurcomputers meldet sich mit der Überschrift „AC 1 U 880 - MONITOR V 3.1“, dem Promptsymbol # (Doppelkreuz) und dem dahinter blinkendem Cursor am Anfang der Eingabezeile. Ein Promptsymbol soll dem Bediener zeigen, daß der Computer bereit ist, eine Eingabe oder einen Befehl entgegenzunehmen und darauf entsprechend seinem Programm zu reagieren. Zur besseren Unterscheidung der Programme, in denen man sich gerade befindet, verwendet man meist verschiedene Promptsymbole. So benutzt der Mini-Basic-Interpreter z.B. ein > (größer als) als Promptsymbol. Jeder Befehl wird dem Monitor in Form einer Kommandozeile übergeben, die die Form # X aaaa bbbb cccc CR (Wagenrücklauf) hat. X ist hierbei ein Zeichen aus dem ASCII-Zeichensatz, mit dem der Befehl abgekürzt wird. Diese Abkürzung basiert meist auf einem Schlüsselwort in englischer Sprache. aaaa, bbbb und cccc stellen drei maximal vierstellige hexadezimale Argumente zum Befehl dar. Führende Nullen in den Argumenten dürfen weggelassen werden. Die einzelnen Argumente sind durch mindestens ein Leerzeichen voneinander zu trennen. Zwischen dem Kennzeichen und dem ersten Argument braucht kein Leerzeichen zu stehen; es schadet aber nicht. Werden weniger Argumente angegeben, als der Befehl erfordert, so wird für die fehlenden der Wert 0 angenommen. Das Kommando gelangt erst durch Drücken der CR-Taste zur Analyse und Ausführung zum Rechner: das heißt also, bis dahin kann die Eingabe mit Hilfe der Backspacetaste (Rückschritt) noch korrigiert bzw. verändert werden. Ist der Befehl falsch oder nicht im Monitor enthalten, so quittiert der Computer die Eingabe mit der Ausschrift WHAT #. Fügt man dem Kennzeichen anstatt der Argumente einen : (Doppelpunkt) an, so kommen die letzten zwischengespeicherten Argumente zur Anwendung. Die einzelnen Befehle sind:

Befehle

A aaaa bbbb c (Arithmetik)

Es werden Summe, Differenz, wenn möglich das Displacement, (relative Distanz) für einen Sprungbefehl aus den ersten beiden Argumenten sowie der dezimale Wert des ersten Arguments, berechnet. c gibt die Länge des Sprungbefehls an. Für die relativen Sprungbefehle des U880-Befehlssatzes ist dann c z.B. gleich zwei.

B aaaa (Breakpoint)

Dieser Befehl ist nützlich für das Testen von bzw. die Fehlersuche in Programmen. Er setzt ein Softwarehaltepunkt auf die Adresse aaaa. aaaa muß dabei immer auf das erste Byte eines Befehls zeigen. Ist nach dem Starten des Programms diese Adresse erreicht, erfolgt die Ausschrift BREAK AT aaaa # und die Kontrolle geht wieder an den Monitor zurück. Zuvor werden alle Registerinhalte der CPU in die RSA (Register Save Area) gerettet, so daß sie sich z.B. mittels des R-Befehls anzeigen lassen. Das zu testende Programm wird durch den Break- Befehl nicht zerstört. Man kann es dann beispielsweise mit dem Go-Befehl fortsetzen.

C aaaa bbbb cccc (Compare)

Dieser Befehl vergleicht die beiden Speicherinhalte ab aaaa und bbbb für die Länge von cccc Bytes miteinander. Wird eine Ungleichheit gefunden, erscheinen die beiden nicht übereinstimmenden Bytes mit den jeweils zugehörigen Adressen auf dem Schirm. Mit der CR-Taste kann man die Suche fortsetzen. Jede andere Taste führt zum Abbruch.

D aaaa bbbb (Display Memory; Dump)

Der Speicherinhalt von der Adresse aaaa bis zur Adresse bbbb wird als Hexdump auf dem Bildschirm ausgegeben. Zu Beginn jeder Zeile erfolgt die Ausgabe der jeweiligen Anfangsadresse, es folgen 16 Datenbytes.

E aaaa (Execute Machine Programm)

Das Maschinenprogramm ab Adresse aaaa läuft unter Breakpointkontrolle. Das heißt, die Register der CPU werden entsprechend der aktuellen Werte der RSA geladen, der Softwarebreakpoint wird aktiviert und dann wird zur Adresse aaaa gesprungen.

F aa bb ce dd … nn (Find String)

Hier ist ab Adresse 00aaH die Datenfolge aa bb cc dd … nn zu suchen. Die zu suchende Datenfolge oder Zeichenkette kann dabei maximal so lang sein, daß das gesamte Kommando gerade noch auf den Bildschirm paßt. Wird die angegebene Zeichenkette vollständig gefunden, springt der Monitor an den M-Befehl, wobei die Adresse auf das erste Byte der Datenfolge zeigt. Soll die Suche nach dem M-Befehl weitergehen, ist die Folge F: aa bb cc dd … nn einzugeben. Wird die Datenfolge im gesamten Speicher nicht gefunden, erfolgt die Ausschrift NOT FOUND #.

G (Go on)

Dieses Kommando funktioniert analog dem E-Befehl. Die Programmausführung wird hierbei ab der Adresse im PC fortgesetzt. Falls ein Breakpoint zuvor eingegeben wurde, wird dieser aktiviert.

I (Initialize)

Hiermit ist ein Löschen aller Anwenderregister möglich. Alle Speicherzellen der RSA werden (bis auf die für den Stackpointer) Null gesetzt; letzterer so, daß das jeweilige Ende des RAM-Speichers im Grundmodul vorsorglich als Anwenderstack genutzt wird. Dies deshalb, weil es z.B. für den E-, J- und G-Befehl erforderlich ist. Sollte dieser Bereich für den einen oder anderen Anwendungsfall nicht günstig sein, kann man den Stack natürlich mit Hilfe des R- Befehls auch in einen anderen RAM-Bereich legen. Auf dem Bildschirm erscheint die Ausschrift CLR/RSA.

J aaaa (Jump)

Dieser Befehl führt ebenfalls zum Ansprung eines Anwenderprogramms ab der Adresse aaaa analog dem E-Befehl, jedoch ohne Breakpointaktivierung.

L aaaa ± (Load from Cassette)

Der Computer versucht, ein Programm oder eine Datei, kurz File genannt, vom Bandgerätinterface in den Speicher zu laden. Das Argument aaaa± bewirkt, das File um diesen Offset verschoben vom ursprünglichen Speicherbereich einzulesen, zum Beispiel dann, wenn der Speicherbereich, von dem das File abgespeichert wurde, im eigenen Computer nicht vorhanden ist. Sonst, also in der Regel, kann man dieses Argument weglassen. Wird der Ladevorgang erfolgreich beendet, so trägt diese Routine die Startadresse des Files, die auch auf dem Bildschirm erscheint, in die Speicherzelle für das erste Argument ein. Das geladene Programm läßt sich dann einfach mit J: anspringen.

M aaaa (Modify Memory)

Hiermit kann der RAM-Speicher ab der Adresse aaaa byteweise angezeigt und neu beschrieben

werden. Nach jedem Drücken der CR-Taste erscheinen die aktuelle Adresse und das zugehörige Byte. Nach dem Promptsymbol läßt sich ein neues Byte oder auch eine Bytefolge, durch Leerzeichen getrennt, eingeben. Drückt man laufend nur die CR-Taste, erfolgt die Speicheranzeige Byte für Byte. Bei einer Dateneingabe würde diese nach dem Drücken der CR-Taste in den Speicher übertragen und der Schreibvorgang überprüft. Sollte das neue Byte nicht vom Speicher übernommen worden sein, weil er z.B. im EPROM-Bereich liegt oder gar nicht vorhanden ist, so erfolgt eine Fehlermeldung. Man schließt den M-Befehl durch die Eingabe eines. (Punkt) ab. Dabei gelangt die zuletzt bearbeitete Speicheradresse in die Speicherstelle für das zweite Argument. Dadurch läßt sich der bearbeitete Bereich mit D: noch einmal betrachten.

P aaaa bbbb cc (Pattern)

Dieser Befehl füllt den Speicher von der Adresse aaaa bis zur Adresse bbbb mit dem Bitmuster cc auf.

R XX (Register Display/Modify)

Dieses Kommando ermöglicht analog dem M-Befehl die Anzeige und das Verändern der Inhalte aller Register der CPU. Hierbei steht XX für das jeweilige Registerpaar. Nach dem Drücken der CR-Taste erscheint der Wert des jeweiligen Doppelregisters, gefolgt vom # auf dem Bildschirm. Mit der nachfolgenden Eingabe eines Wertes läßt sich das Registerpaar dann neu setzen. Folgt dem R ein Doppelpunkt, so werden der gesamte Registersatz, der Breakpoint, die Breakpointsequenz (3 Bytes ab Breakpointadresse) angezeigt sowie die gesetzten Flaggs zusätzlich anhand ihres Symbols dargestellt.

S aaaa bbbb cccc name (Save to Cassette)

Hiermit kann man eine File ab der Adresse aaaa bis zur Adresse bbbb mit der Startadresse cccc unter dem Namen name (Länge max 6 Zeichen) auf Magnetband abspeichem.

T aaaa bbbb cccc (Transfer)

Mit diesem Befehl kann der Speicherinhalt ab der Adresse aaaa in den Speicherbereich ab der Adresse bbbb für die Länge von cccc Bytes kopiert werden. Ein Überlappen der beiden Speicherbereiche ist zulässig. Beispiel: T 1900 1901 80 verschiebt den Speicherinhalt ab 1900H für 80H Byte um eines nach oben.

V (Verify with Cassette)

Hiermit ist es möglich, ein auf Magnetband abgespeichertes File noch einmal mit dem Speicheroriginal zu vergleichen. Dadurch sind z.B. Datenfehler aufgrund mangelhaften Bandmaterials vermeidbar.

Z

Mit diesem Befehl wird in den Mini-BASIC-Interpreter gesprungen, vorausgesetzt, daß die dafür erforderlichen beiden EPROMs gesteckt sind.

u aaaa bbbb cccc (u -Leerzeichen)

Ein Leerzeichen, gefolgt von Argumenten, speichert deren Werte in den entsprechenden Speicherzellen ab, so daß danach mit dem Operator : gearbeitet werden kann. Damit ist es z.B. möglich, die Startadresse für den F-Befehl frei zu wählen.

Internes

Nach dem Einschalten bzw. jedem RESET-Impuls beginnt der Amateurcomputer ab Adresse 0 mit der Ausführung des Programms. Um hier universell zu bleiben, und nicht bei jedem Programmwechsel auch die EPROMs austauschen zu müssen, bekam dieser Anfangsbereich ein Programm, das nach dem Einstellen eines bestimmten Anfangszustandes (Bildschirmlöschen, Fertigmeldung), Initialisierung genannt, in den Dialog mit dem Benutzer tritt, so daß dieser mittels entsprechender Befehle über die weitere Programmausführung entscheiden kann. In seiner Funktion stellt es sozusagen ein minimales Betriebssystem dar.

Das Monitorprogramm enthält Befehle zum Einlesen und Abspeichern von Programmen bzw. Speicherinhalten auf Magnetband, zur Anzeige und Modifizierung von Speicher- und Registerinhalten, zum Ansprung von Nutzerprogrammen und zur Programmentwicklung und -testung. Damit ist es also bereits möglich, selbst Programme in Maschinensprache zu entwickeln, zu testen und das fertige Programm z.B. auf Magnetbandkassetten abzuspeichem bzw. auch Programme für den „AC1“ einzulesen und abzuarbeiten, die andere Amateure entwickelt haben. Das Einlesen der Programme vom Magnetband in den RAM-Speicher des Rechners und der nachfolgende Ansprung dieser Programme soll den Regelfall bei der Nutzung des „AC1“ darstellen. Auf diese Art ist ein relativ schneller Programmwechsel möglich, und, weil sich bei diesem Prinzip nur das gerade benötigte Programm im Speicher des Rechners befinden muß, braucht dieser auch nur so groß zu sein, wie es das jeweilige Programm erfordert.

Die Möglichkeit, häufig benötigte Programme im restlichen bzw. erweiterten EPROM-Speicher abzulegen, besteht natürlich auch, ist aber weitaus teurer als die Nutzung des externen Magnetband-Speichers.

Das Monitorprogramm für den „AC1“ entstand auf der Grundlage von [10]. Dazu wurde der dort veröffentlichte hexadezimale Speicherausdruck (Hexlisting) rückübersetzt, kommentiert und der Hardware des „AC1“ angepaßt; also mit entsprechenden Routinen für Tastatur, Bildschirm und Tonbandausgang versehen. In der endgültigen Version wird er etwa 2 KByte EPROM und etwa 60 RAM--Zellen belegen. Eine abgerüstete Variante, die nur das Einlesen und Anspringen von Programmen ermöglicht und dann etwa 1 KByte EPROM benötigt, ist ebenfalls denkbar.

[10] Krake, H.: ZETBUG - ein komfortabler Z-80-Monitor, Funkschau 52 (1980), H.11

Erweiterbarkeit

Die einzelnen Befehlsroutinen des Monitors werden nicht über einen Sprungverteiler erreicht, sondern vom Monitor anhand eines Kodemusters im Adreßbereich von 0 bis 1FFFH gesucht. Jede Befehlsroutine hat dabei die Form 00 09 XX 0D … Routine … C9. XX ist der ASCII-Kode des entsprechenden Kennbuchstabens. Man braucht eigene, z.B. von Magnetband ladbare Ergänzungen, also nur in diesen Rahmen zu „packen“. Damit ist es ebenfalls möglich, eigene Anwenderprogramme über Kennbuchstaben zu starten. Der derzeitige Monitor belegt die ersten beiden Kilobyte des EPROM-Bereiches und etwa 128 Byte RAM zu Beginn des Arbeitsspeichers. Wenn man noch etwas Spielraum für mögliche Modifikationen späterer Monitorversionen läßt, hat man also den Speicherbereich von 1900H bis 1FFFH für Erweiterungen zur Verfügung. Es ist auch denkbar, auf den Mini-BASIC-Interpreter zu verzichten und diesen Bereich für Erweiterungen zu nutzen. Da der Monitor am Anfang des Speicherbereiches liegt, fallen in diesen Bereich auch die Ansprungpunkte der RST-Befehle und

der Beginn der NMI-Routine.

Um nun trotzdem auch diese Möglichkeiten in Anwenderprogrammen nutzen zu können, ohne dabei die EPROMs zu wechseln, was auf Dauer auch im Interesse der Fassungen nicht zu empfehlen ist, sind die Routine RST8...RST38H sowie der NMI-Ansprung über eine Sprungtabelle im RAM geführt. Diese läßt sich vom Anwender modifizieren.

Systemadressen	
1800H - 1801H	Speicher für Cursorposition
1802H - 1804H	Sprung zur RST 8H-Routine (Eingabe Zeichen)
1805H - 1807H	Sprung zur RST 10H-Routine (Ausgabe Zeichen)
1808H - 180AH	Sprung zur RST 18H-Routine (Ausgabe Zeichenkette)
180BH - 180DH	Sprung zur RST 20H-Routine z.Z. vom Monitor nicht belegt
180EH - 1810H	Sprung zur RST 28H-Routine z.Z. vom Monitor nicht belegt
1811H - 1813H	Sprung zur RST 30H-Routine z.Z. vom Monitor nicht belegt
1814H - 1816H	Sprung zur RST 38H-Routine (Fehlermeldung)
1817H - 1819H	Sprung zur NMI-Routine z.Z. vom Monitor nicht belegt
185BH - 185CH	Speicher für Argument 1
185DH - 185EH	Speicher für Argument 2
185FH - 1860H	Speicher für Argument 3

nutzbare Unterprogramme

RST 08H	holt ein Zeichen von der Tastatur und kehrt mit dem ASCII-Kode des Zeichens im Akku zurück
RST 10H	gibt das im Akku enthaltene Zeichen (ASCII-Kode) auf dem Bildschirm aus und rückt den Cursor um eins weiter
RST 18H	gibt die dem Unterprogrammaufruf folgende Zeichenkette auf dem Bildschirm aus bis einschließlich dem Byte, bei dem Bit 7 gesetzt ist, bewegt den Cursor weiter, kehrt dann zum folgenden Byte zurück
CALL 07EBH	MS30

Akku bei Return = 0 |

CALL 07EEH	OUTHEX

kein Register wird zerstört |

CALL 07F1H	OUTHL

kein Register wird zerstört |

CALL 07F4H	INLINE

kein Register wird zerstört |

CALL 07F7H	INHEX	wandelt eine maximal vierstellige in ASCII-Zeichen angegebene Zahl ab (DE) abwärts in deren hexadezimalen Wert um, der dann in HL steht. DE wird entsprechend dekrementiert, der Akku wird zerstört

CALL 07FAH	TASTE	testet den Tastaturstatus, kehrt bei gedrückter Taste mit dem nach 30 ms anliegenden Kode zurück (wartet nicht auf loslassen der Taste!); wenn keine Taste gedrückt, erfolgt sofortige Rückkehr mit gesetztem Zero-Flag
CALL 07FDH	GETCO1	Sprung zur Monitoreingabeschleife, der Monitorstack wird neu initialisiert