Um den Fortran Compiler zu benutzen, laden Sie das Programm wie folgt in den Monitor:

```
** MONITOR 01Z **
*L
PLAY
LOADING   BBG-FORTRAN
```

Im MZ-800 kann das Programm auch über IPL (Taste C) eingeladen werden. Vorher 700 MODUS einstellen.

Nach dem Laden meldet sich das Programm selbsttätig
Sie befinden sich jetzt in dem "Compiler-Mode".

Die verfügbaren Befehle im "Compiler-Mode" sind:

EDIT	(E)	- Sprung in den "Editor-Mode"
COMPIL	(C)	- Compilieren des im Speicher befindlichen Programms
EXEC		- Compilieren eines Programms direkt von Tape, ohne das Textfile in den Editor zu laden.
BSAVE		- Abspeichern des compilierten Programms auf Cassette.
RUN	(R)	- Starten des compilierten Programms
LIST	(L)	- Der Befehl COMPIL liefert ein Source Listing
LISTN	(LN)	- COMPIL liefert kein Source Listing
LISTE	(LE)	- Nur Programmzeilen mit einem Fehler werden bei COMPILE ausgegeben.
BYE		- Rückkehr zum MONITOR

Der Text-EDITOR:
Nach Eingabe des Befehls EDIT oder nur E befinden Sie sich im EDITOR-MODUS. Nur in diesem MODUS können Programme eingegeben oder verändert werden.
Der Editor hat folgende Befehle:

BEFEHL &

Löscht das im Speicher befindliche Source-Programm.

BEFEHL:!!
Rücksprung zum Compiler-Mode

BEFEHL: I
Zufügen von neuen Programmzeilen am Programmende

BEFEHL: L
Listen des Source-Programmes.
L -Listen des gesamten Programmes
L,n - Listen ab Zeilennummer n
L,P - Listen auf dem Drucker
Mit SHIFT & BREAK stoppt man das Listing, und kehrt zum Editor Modus zurück.

BEFEHL: Bn
Einfügen von Zeilen. Vor der Zeile n wird eine Zeile eingefügt. Nach der Eingabe des Befehls wird der gewünschte Text eingegeben. Die Eingabe wird mit CR abgeschlossen.

BEFEHL: R
Laden eines Source Programmes von Cassette.

BEFEHL: Wfilename
Schreiben eines Programms auf Cassette.

Im Editor Modus steht Ihnen ein Full-Screen Editor zur Verfügung, d.h. Sie können Programmzeilen wie in Basic korrigieren und etwas einfügen. Die Eingabe einer neuen Zeile wird immer mittels der Taste "I" vorgenommen. Vorhandene Zeilen können nach dem List(L-Befehl) direkt auf dem Bildschirm durch Bewegung der Cursortasten usw. geändert werden.

Fortran Befehle und Programmaufbau:
Jedes Fortran-Programm ist folgendermaßen aufgabaut:

```
IZn:   Label      Befehl
 l:    222        WRITE ($16,A1,"avg")
```

IZn: ist die interne Zeilennummer, die vom Compiler selbsttätig erzeugt wird. Sie darf <u>nicht</u> mit eingegeben werden.
Zwischen Zeilennummer und dem Programmteil müssen 6 Leerzeichen sein. In diesem Bereich darf nur eine bis zu 4-Stellen lange Zahl stehen. Diese Zahl nennt man LABEL und dient als Ansprungadresse.
MUSTERPROGRAMM:

```
1 241 WRITE ( ($16,A1)
2       W R I T E    ("Maike")
3       DO 20 I=1,255
4       MEM (53248)-l
5
```

5 20 CONTINUE

Nur ein Befehl pro Programmzeile ist erlaubt.
Variablen können in FORTRAN integer oder real sein. Wenn ein Variablenname mit den Buchstaben I-N beginnt, dann ist die Variable eine Integer Variable. Alle anderen Variablen sind REAL.
Integer-Variable müssen im Bereich von -32768 bis +32767 liegen.
Real-Variable liegen im Bereich von 1.67E-19 bis 1.67E+18 und können negativ, positiv oder null sein.
Die Zahl PI ist bereits auf 3.14159 definiert.
Die Variablennamen können von beliebiger Länge sein. Vom Programm werden jedoch nur die ersten vier Stellen berücksichtigt.

Die Rechenoperatoren sind wie in BASIC (+,-,*,/).
Zu beachten ist jedoch, daß eine Rechenoperation nicht ohne weiteres mit einer Real und einer Integer-Variable ausgeführt werden können. (Siehe Befehl FLOAT).
Zuweisungen sind wie in einem Basicprogramm, jedoch ohne LET.

Abkürzungen:

exp	-	Integer Expression
fxp	-	Floating Point Expression
v	-	Variablenname
n	-	Integer Konstante
hx	-	Hexadezimale Zahl

DIMENSION: Dimensionieren von Variablen, z.B. DIMENSION A(5), I(7) liefert zwei Array's: Ein REAL-Array mit den Elementen A(O)-A(5). und ein Integer Array I(0)-I(7). Der DIMENSION-Befehl muß ganz am Programmanfang stehen.

GOT0 label: sprung zum angegebenen Label.

IF (exp) label 1, label2, labe13: Wenn der Integer Ausdruck negativ ist, verzweigt das Programm zum labell, wenn der Ausdruck null ist, verzweigt das Programm zum labe12, wenn der Ausdruck positiv ist, verzweigt das Programm zum labe13. Label1, 2,,3 können ggf. weggelassen werden, z.B. IF(1-5), 1000. In diesem Falle springt'das Programm zum LABEL 1000 falls I größer als 5 ist.

DO label v=exp 1, exp 2, exp 3: Dies ist ein Schleifenbefehl und zählt von exp 1 nach exp 2 step exp 3 kann ggf. weggelassen werden.
Wenn das Programm die Programmzeile mit dem label erreicht, wird die schleife ggf. erneut durchlaufen. In dieser Programmzeile mu8 ein CONTINUE stehen. (Siehe Musterprogramm).

CALL label: Unterprogrammaufruf des Labels

RETURN: Rückkehr von der Subroutine zur nachfolgenden Zeile des entsprechenden CALL's.

PAUSEn: (n ist Integer). Das Programm wird angehalten und es wird PAUSE n ausgegeben. Bei drücken einer Taste wird das Programm fortgesetzt.

STOPn: Das Programm wird unterbrochen. Auf die Frage RESTART? kann man mit Y oder N antworten. Bei Y wird das Programm erneut durchlaufen, bei N springt das Programm zurück in den Compiler.

BREAK: Erst nach setzen eines BREAK kann die Programmausführung mit SHIFT & BREAK unterbrochen werden.

USR (exp): Aufruf einer Maschinenspracheunterroutine (wie in BASIC)

$ML hx1,hx2,.... : Mit diesem Befehl kann man direkt Maschinensprachebefehle in das Fortranprogramm integrieren.

$ML 11,A3,11,CD,30,00 spielt Music mit der Frequenz, die in $11A3 abgespeichert ist.

END: Alle Programme müssen mit einem END beendet werden. Alle nachfolgenden Zeilen werden vom Compiler ignoriert.

READ (v1.xx, v2.xx, ...): Dem READ Befehl entspricht im BASIC der INPUT-Befehl. Die möglichen Eingaben sind folgende:

v.I Dezimaler-Interger-Wert

v.B Integerwert in Hexadezimal

v.E REAL-Wert in Dezimal

v.A1 Ein einzelner Buchstabe wird eingelesen und im ASCII-Code abgespeichert.

v.A2 Zwei CHARAKTER werden eingelesen und in High- und Low Bytes abgespeichert.

Beispiel: READ ("Eingabe von X:" ,X.I) Nach Ausgabe des Textes wird X mit dem eingegebenen Integer Wert gefüllt.

WRITE (exp1.xx,exp2.xx,...): Der Write-Befehl wird benutzt für Ausgaben auf dem Bildschirm oder auf dem Drucker. Folgende Formate sind möglich:

exp Ausgabe Dezimal

exp.In Der Integer-Ausdruck I wird in einem Feld der Länge n-1 Ziffern ausgegeben.

fxp.E Ausgabe einer REAL-Expression.*

exp.B2 Hexadezimale Zahl wird in einem 2-ziffrigen Feld ausgegeben.

exp.B4 Hexadezimale Zahl wird in einem 4-ziffrigen Feld ausgegeben.
exp.X X Leerzeichen werden ausgegeben.
exp.A1 Ein Charakter wird ausgegeben im ASCII-Format
exp.A2 Zwei Charakter werden ausgegeben.
exp.V Vertical Position der Ausgabe auf dem Bildschirm
exp.H Horizontal-Position der Ausgabe.
Beispiel: WRITE (/,12.H,16.V, "BBG-SOFTWARE"). Das Zeichen / bedeutet Zeilenvorschub.
WRITE ($16.A1) entspricht CLEAR-HOME
WRITE ($15.A1) entspricht HOME
WRITE (1.A1) bedeutet Ausgabe auf dem Drucker
WRITE (0.A1) bedeutet wieder Ausgabe auf dem Bildschirm
SETG (exp1, exp2) und RESG (exp1, exp2): SET und RESET der Punkte mit den entsprechenden x- und y-Koordinaten.

FORTRAN Standard Routinen:

MEM (exp): entspricht POKE und PEEK in BASIC
MEM (exp1) = exp2 entspricht POKE expl, exp2
v=MEM (exp) entspricht v=PEEK (exp)
GET: Wie in Basic nur wird der ASCII-Wert abgespeichert.
I=GET Die Variable I enthält den ASCII-Wert der gedrückten Taste. Falls keine Taste gedrückt wurde ist I=0.

IOC: Wie MEM, nur auf den Z-80 Ports
IOC (2) = 3
LOW (exp): gibt das Low-Byte der exp
MOD (exp1, exp2): MOD entspricht der mathematischen Modulfunktion und bedeutet eine ganzzahlige Division von expl durch exp2.
IRND (exp): Eine Zufallszahl aus dem Bereich 0 bis exp-1 wird erzeugt.
IABS (exp): Absolutwert von exp (wie in BASIC)
ISIGN(exp1, exp2): Wenn exp1 größer als exp2 ist, liefert diese Funktion exp1. Falls beide gleich sind null, und falls expl kleiner als exp2 ist, liefert sie -exp1.
ABS Wie IABS jedoch für REAL-Zahlen

SORT (fxp): liefert die Quadratwurzel einer REAL-Zahl

SIN (fxp), COS (fxp), TAN (fxp): liefert die jeweilige Winkelfunktion in Radiant.

ATAN (fxp): Arcustangens

ALOG (fxp), EXP (fxp): Natürlicher Logarithmus und Exponentialfunktion.

FLOAT (exp): Wandelt eine Integerzahl exp in eine REAL-Zahl um.

IFIX (fxp): Wandelt eine REAL-Zahl in eine Integerzahl um.

IOR (exp1, exp2), IAND (exp1, exp2), IXOR (exp1, exp2): BOOLEAN- Funktionen.

Fehlermeldungen:

ERROR 1 : falscher Editorbefehl
ERROR 2 : Fehler im Variablennamen
ERROR 3 :
ERROR 4 : DIMENSION-Befehl steht nicht am Programmanfang
ERROR 5 : Syntax-Error
ERROR 6 : Schleifenverschachtelung ist größer als 6
ERROR 7 : Fehler im Array-Variablennamen
ERROR 8 : ERROR beim Dimensionieren (mehr als 2047 Elemente)
ERR DO LOOP: Schleifenfehler
MEMORY SIZE OVER ABORT: Programm ist zu lang
ST NO NOT FOUND: Label für DO,GOTO,IF oder CALL fehlt
FILE ERROR ABORT: kein zu compilierendes Programm vorhanden
OBJECT NOTHING: kein compiliertes Programm für den RUN-Befehl vorhanden.

Folgende Fehler können beim Programmablauf auftreten:
ARRAY ERROR: Eine dimensionierte Variable ist out of range
0 ERROR : REAL-Variable größer als 1.67E+18
U ERROR : REAL-Variable underflow
N ERROR : Falscher mathematischer Wert z.B. SQRT (-1.0)
D ERROR : Division durch null

WARMSTART aus dem Monitor (ohne Löschen des Programms) $ 122A
KALTSTART aus dem Monitor (mit Löschen des Programms) $ 1200

Beispielprogramme:
Hier ein Programm welches eine Division zweier Realzahlen vornimmt. Wir müssen beachten, daß den Realzahlenvariablen nicht die Werte I-N zugewiesen werden dürfen, da diese den Integer Variablen zugewiesen sind. Wir würden keine sinnvollen Werte erhalten.
Wenn wir das Programm listen, erhalten wir folgendes Listing.

```
L
  0:      A=11
  1:      B=22
  2:      C=B/A
  3:      WRITE(C.E)
  4:      KL=IFIX(C)
  5:      WRITE(/,KL)
  6:      END
```

Wir springen dann in den Compiler Mode mit

```
!
```

Nun geben wir C ein und es wird das Programm beim Compilieren aufgelistet. Das Programm müßte eine Länge von 125 Bytes haben. Wenn wir es mit R starten, so müßte zuerst 0.200000E+1 und dann 2 erscheinen. Zuerst wurde die Realzahl als Ergebnis der Division ausgegeben, dann diese in eine Integerzahl mit Hilfe des IFIX Befhles umgeformt und dann ebenfalls ausgegeben..
Nun ein Programm zum Ausprinten aller Zeichen auf dem Bildschirm.

```
 L
    0:     DO2MM=1,255
    1:     DO4KL=53248,54247
    2:  4  MEM(KL)=MM
    3:  2  CONTINUE
    4:     END
```

Jedes Programm muß mit END abgeschlossen sein. Wichtig ist es, daß beim Printen der auszuprintende Wert in Klammern steht, ebenso beim READ Befehl. Am allerwichtigsten ist, daß man Real- und Integer Variablen nicht verwechselt, also Integervaiablen fangen immer mit I-N an, Real mit A-H und O-Z.