

TI99/4A

INTERN

Heiner Martin

The Operating System
of TI 99/4A internal
ROM and GROM Listing with
Commentary and Directions for GPL



Verlag für Technik und Handwerk GmbH

Heiner Martin
The Operating System of TI 99/4A intern



TI99/4A

INTERN

Heiner Martin

**The Operating System
of TI 99/4A internal
ROM and GROM Listing with
Commentary and Directions for GPL**



Verlag für Technik und Handwerk GmbH

**Translated by
Peter Coates**

© 1985 by Verlag für Technik und Handwerk GmbH,
D-7570 Baden-Baden, Postfach 11 28, West-Germany
(ISBN 3-88180-009-3)

Copyright

All rights reserved including translation, reprint and
photomechanical reproduction in whole or in part.

The publisher has taken care to publish right and
complete information. The publishing house
„ Technik und Handwerk GmbH „ cannot be held responsible
for the use for this information for violation of
patent rights or other rights of a third party resulting
from this use.

Note: TI/994A is trademark of Texas Instruments INC, Houston/USA

Printing: F. W. Wesel, Baden-Baden

Index:

Preface	8
The ROM	9
The ROM Listing	11
Graphic Programming Language	80
The GPL Commands	81
The GPL Command formats	95
The GROM 0	98
The GROM 0 Listing	99
The Hexdump of Sample GROM	126
The Basic GROM'S	132
The Basic Value Stack	134
The Basic Symbol Table	135
Listing of GROM 1	136
Listing of GROM 2	171
References to Extended Basic	209

Preface

The TI99/4A is a Home Computer about which little is known.

This is due primarily to the fact that the manufacturer has published very little information about its inner structure. Nothing could be obtained from Texas Instruments about the operating system called " Monitor " and the Programming Language GPL (Graphic Programming Language) used for the TI99/4A.

Therefore the operating system of the TI99/4A (ROM and GROM'S) has been analyzed in detail and the results are summarized in this book. We gave more attention to the parts not well known up to now, than to the program parts which, in most cases, are only used as support to assembler programs i.e. the mathematical routines. Since the information of the manufacturer has been very limited we cannot guarantee the accuracy of all commentaries. We apologize for any mistakes.

We recommend reading two publications by Texas Instruments (Handbook Editor/Assembler and TI 99 Console and Peripheral Expansion System Technical Data) before reading this book. Knowledge of 9900 Assembler is necessary.

This book has been written on a TI 99/4A equipped with two Disk Drives using the TI-Writer and a Mannesman/Tally MT 140SF Printer. My acknowledgement goes to Dipl Ing Michael Weiland who helped me write this book by developing two Disassembler Programs and by giving me support.

ULM January 1985

Heiner Martin

Addendum

After several requests of TI 99/4A enthusiasts in the past years, I rebuilt the layout of this book with the original files transferred from a TI 99/4A to an Windows 95 PC.

Due to the limited demand, this version of TI 99/4A Intern is published in Acrobat Reader format. This file is distributed as freeware, but any upload to a BBS or an online service is only allowed with permission of the author.

ULM, September 1997

Heiner Martin
email: Heiner_Martin@compuserve.com

THE ROM

The operating system of the TI99/4A has several versions. In the ROM area between >0000 and >1FFF only minor changes have been evident up to now. Therefore the listing of only one version shall be enough. How little Texas Instruments has been able to make change in the ROM, can be seen for example, by the fact that the Extended Basic Modul accesses directly into some subroutines. The ROM contains the GPL Interpreter, the interrupt routines, the Cassette routines, part of the Basic Interpreter and several Utility Subroutines.

The GPL Interpreter

GPL (Graphics Programming Language) is a language used for many purposes in the TI99/4A. Programs in this language are contained in the GROM'S. In the second part of this book we will talk more about details of this language. The GPL Interpreter is located at addresses >0024 through >08FF and in some other small areas. There is little difference between the GPL Interpreter and the Basic Interpreter, since there are GPL commands, which are only of concern to the Basic Interpreter i.e. PARSE, CONT and RTNB.

The Interrupt Routine

The interrupt routine is located at address >0900 through >0ABE and tries first to locate the cause of the interrupt. If the cause is not a console interrupt released by VDP, the corresponding DSR interrupt routines are scanned. If the cause is a cassette interrupt, then you jump into a cassette interrupt routine. In case of a console interrupt, which happens every fiftieth of a second in consoles with PAL systems and every sixtieth of a second in consoles of American standards, the automatic Sprite movement and the sound process are checked and the QUIT key is pressed. Finally you check to see if the screen can be turned off and if the "User defined Interrupt Pointer " at address >83C4 is busy. If the screen can be turned off, the routine will be executed with BL *R12. If the screen cannot be turned off, the interrupt is finished.

The Utility Subroutines

Several Utility Subroutines can be used in ROM by activating XMR Command of GPL or by activating the XMR Utility Subroutines of the Assembler Module. This can be done by using two tables. The first table starts at >0CFA and contains the addresses of 16 more tables, which include the start addresses of the corresponding routines. Two of these address tables are contained in the ROM of the console, starting at >0D1A and >12A0. >0D1A contains the floating point routines mentioned in the Assembler Handbook and some additional routines to round floating point numbers. 12A0 has been named by Texas Instruments " XTAB " and contains essentially the starting addresses of some Utility Subroutines which are

important for the Basic Interpreter. Some of these cannot be used by Assembler programs since they lead back to the GPL Interpreter.

The Cassette Routine

The Cassette Routine is located at addresses >1346 through >15D3. The input or output of data into the cassette recorder port takes place in this area. Unfortunately these routines and their main parts are not directly usable by the Assembler programs, since they go directly back to the GPL Interpreter.

The Basic Interpreter

The ROM contains only part of the Basic Interpreter. It starts at >18C8. Some of the XML XTAB routines also belong here. The table of the jump addresses for the Basic Routines starts at >1C9C. When the first bit is set in the address, the corresponding routine is then located in Basic GROM'S 1 and 2. The best results can be obtained by starting at >1986 (EXECute Basic).

We also want to mention that the operating system contains a few routines which are only usable by extensions. These extensions, though, are not available at the present time. The GPL interpreter takes into consideration an extension which is located at CRU address >1B00. The operating system also supports, at least partially a better decoding of the writing and reading addresses; thus the development of a specific extension for the module port seems possible.

```

*****
*
*           Analysis of TI 99/4A
*
*****
*
*           Console ROM at >0000 through >1FFF
*
*           20.7.84 H. Martin
*
*****

```

```

0000  83E0  DATA >83E0      RESET Vektor
0002  0024  DATA >0024
0004  83C0  DATA >83C0      INT Level 1
0006  0900  DATA >0900
0008  83C0  DATA >83C0      INT Level 2
000A  0A92  DATA >0A92

000C  30AA  DATA >30AA      Clock frequency and header

000E  0460  B    @>02B2      Keyboard scanning
0010  02B2

0012  0008  DATA >0008
0014  1E00  DATA >1E00

0016  0460  B    @>007A      GPL interpreter, if in R9 GPL-byte
0018  007A

001A  1E00  DATA >1E00

001C  0460  B    @>0078      GPL interpreter without interrupt
001E  0078

0020  0460  B    @>04B2      Keyboard scanning for CLEAR
0022  04B2

Reset and GPL EXIT:
0024  020D  LI   13,>9800    Load system pointer    GROM read data
0026  9800
0028  020E  LI   14,>0100    System flag
002A  0100
002C  020F  LI   15,>8C02    VDP write address
002E  8C02
0030  0200  LI   0,>0020      GROM address
0032  0020
0034  1013  JMP  >005C

0036  1000  JMP  >0038      Turns off GPL extension
0038  1E00  SBZ  >0000
003A  02E0  LWPI >280A
003C  280A
003E  0380  RTWP          and return
0040  280A  DATA >280A      BLWP vectors, turn on GPL extension and
0042  0C1C  DATA >0C1C          start

0044  FFD8  DATA >FFD8      XOP1
0046  FFF8  DATA >FFF8
0048  83A0  DATA >83A0      XOP2
004A  8300  DATA >8300

004C  1100  DATA >1100      CRU value QUIT key

```

```

*****
* Notes to GPL interpreter:
*
* In general the registers are used as follows:
*
* D: R3=Address, R2=Data, R4HByte=Flag VDP-RAM
* S: R1=Address, R0=Data, R4LByte=Flag VPD-RAM
* GPL code is located in R9 HByte, and R5 is flag at word commands
*
*****

```

GPL >F8 SWGR changing to other GROMS

```

004E 06A0 BL @>0864 Push Grom address on substack
0050 0864
0052 06A0 BL @>0864 The same once more
0054 0864
0056 C90D MOV 13,@>8300(4)GROM Read data on substack instead of GROM address

0058 8300
005A C342 MOV 2,13 New GROM read data

005C D11D MOVB *13,4 GROM data in R4
005E C180 MOV 0,6

```

GPL interpreter with GROM address in R6:

```

0060 DB46 MOVB 6,@>0402(13) Write GROM address from R0
0062 0402
0064 DB60 MOVB @>83ED,@>0402(13)
0066 83ED
0068 0402
006A 5820 SZCB @>011B,@>837C Clear condition bit in GPL status
006C 011B
006E 837C

```

GPL interpreter

```

0070 0300 LIMI >0002 Permit interrupt
0072 0002
0074 0300 LIMI >0000
0076 0000
0078 D25D MOVB *13,9 Fetch GPL byte in R9
007A 1105 JLT >0086 Negative? i.e. >80 through >FF
007C D109 MOVB 9,4
007E 09C4 SRL 4,12 1. Nybble
0080 C164 MOV @>0C36(4),5 Trick, address always even number!
0082 0C36
0084 0455 B *5 Execute routine

```

Negative interpreter code:

```

0086 04C4 CLR 4
0088 C149 MOV 9,5
008A 0245 ANDI 5,>0100 Set flag for double (word)
008C 0100
008E 06A0 BL @>077A Fetch D
0090 077A
0092 06C4 SWPB 4 Flag in R4 HByte
0094 C0C1 MOV 1,3 D into the right register
0096 C080 MOV 0,2
0098 0289 CI 9,>A000
009A A000
009C 1A09 JL >00B0 Jump for format 5 commands 5
009E 2260 COC @>0030,9 Immediate operand?
00A0 0030
00A2 160C JNE >00BC Not from GROM!
00A4 C04D MOV 13,1 GROM read data in R1
00A6 D011 MOVB *1,0 Fetch data („S“, IMM Value)
00A8 0601 DEC 1

```

```

00AA 06A0 BL @>07AA Fetch 2 bytes
00AC 07AA
00AE 1008 JMP >00C0

00B0 C209 MOV 9,8 Format 5
00B2 0988 SRL 8,8
00B4 0700 SETO 0 Data becomes >FFFF
00B6 C228 MOV @>0BFE(8),8 Fetch address routine
00B8 0BFE
00BA 0458 B *8 Execute

00BC 06A0 BL @>077A Fetch source
00BE 077A
00C0 C209 MOV 9,8
00C2 0998 SRL 8,9
00C4 C228 MOV @>0C4E(8),8 Fetch address routine
00C6 0C4E
00C8 8002 C 2,0 Compare data D and S
00CA 0458 B *8 Execute routine

GPL CGE:
00CC 11CE JLT >006A Cond bit reset, if low
00CE F820 SOCB @>011B,@>837C Set condition bit
00D0 011B
00D2 837C
00D4 10CD JMP >0070 Once again !

GPL CH:
00D6 1BFB JH >00CE Set condition bit, if high
00D8 10C8 JMP >006A Reset and go on

GPL CHE:
00DA 14F9 JHE >00CE Set condition bit if equal or greater
00DC 10C6 JMP >006A Reset and go on

GPL CGT:
00DE 15F7 JGT >00CE Set condition bit if greater
00E0 10C4 JMP >006A Reset and go on

GPL CLOG
00E2 0540 INV 0 AND both data (INV+SZC)
00E4 4080 SZC 0,2
00E6 13F3 JEQ >00CE Set condition bit
00E8 10C0 JMP >006A Reset and go on

GPL CZ:
00EA C082 MOV 2,2 0?
00EC 02C4 STST 4 CPU status in R4
00EE D804 MOVB 4,@>837C CPU status becomes GPL status
00F0 837C
00F2 10BE JMP >0070 And go on

GPL CARRY, OVF, H, GT:
00F4 C009 MOV 9,0
00F6 0AC0 SLA 0,12 Command in R0 last 3 bits
00F8 09D0 SRL 0,13
00FA D160 MOVB @>837C,5 Fetch GPL status
00FC 837C
00FE 0A05 SLA 5,0 Shift to R0!
0100 18E6 JOC >00CE Set condition bit
0102 10B3 JMP >006A Reset condition bit

GPL B:
0104 D19D MOVB *13,6 Fetch new GROM address
0106 1000 JMP >0108
0108 D81D MOVB *13,@>83ED

```

```

010A 83ED
010C 10A9 JMP >0060 Write new address and on with reset

GPL BS:
010E D120 MOVB @>837C,4 GPL statusbyte
0110 837C
0112 0A24 SLA 4,2
0114 1106 JLT >0122 Condition bit set, then execute
0116 D11D MOVB *13,4 Skip address in GROM
0118 10A8 JMP >006A Clear status and go on

GPL BR:
011A D120 MOVB @>837C,4
011C 837C
011E 0A24 SLA 4,2
0120 11FA JLT >0116 Set condition bit, no execution
0122 D81D MOVB *13,@>83F3 Fetch jump address in Lbyte R9
0124 83F3
0126 0249 ANDI 9,>1FFF Eliminate command
0128 1FFF
012A D1AD MOVB @>0002(13),6 Read high bit of actual GROM address
012C 0002
012E 0246 ANDI 6,>E000 The first 3 bits only
0130 E000
0132 E189 SOC 9,6 New address
0134 1095 JMP >0060 Execute and go on

GPL ABS:
0136 0742 ABS 2 Data ABS
0138 107A JMP >022E Execute

GPL NEG:
013A 0502 NEG 2 Data NEG
013C 1078 JMP >022E Execute

GPL CLR:
013E 0702 SETO 2 First >FFFFFF, then invert

GPL INV:
0140 0542 INV 2 Data INV
0142 1075 JMP >022E Execute

GPL FETCH:
0144 C184 MOV 4,6 Save VDP flag
0146 06A0 BL @>0864 Push actual GROM address on substack
0148 0864
014A 0644 DECT 4
014C 06A0 BL @>0848 Set stack pointer, write GROM address
014E 0848
0150 D09D MOVB *13,2 Fetch data
0152 0882 SRA 2,8 in R2 Lbyte
0154 05A4 INC @>8300(4) Increment GROM address on stack
0156 8300
0158 05C4 INCT 4 Old substack
015A 06A0 BL @>084C Write old GROM address
015C 084C
015E C106 MOV 6,4 Old R4
0160 1066 JMP >022E Execute

GPL CASE:
0162 0602 DEC 2
0164 1782 JNC >006A Smaller than 0, then go on (thus new address in GROM)
0166 D15D MOVB *13,5 Counting of „PC“
0168 1000 JMP >016A
016A D15D MOVB *13,5
016C 10FA JMP >0162 Loop

```

```

GPL PUSH:
016E B80E AB 14,@>8372 Increase data stack pointer(Attention system flag)
0170 8372
0172 D1A0 MOVB @>8372,6 Fetch pointer
0174 8372
0176 0986 SRL 6,8
0178 D9A0 MOVB @>83E5,@>8300(6) Push byte on data stack
017A 83E5
017C 8300
017E 0460 B @>0070 and go on
0180 0070

GPL DECT:
0182 09E0 SRL 0,14

GPL INCT:
0184 0600 DEC 0 -1

GPL INC und SUB:
0186 0500 NEG 0 Negate

GPL DEC und ADD:
0188 D145 MOVB 5,5
018A 134B JEQ >0222 Byte?, then execute
018C A080 A 0,2 Add word
018E 104C JMP >0228 Execute

GPL AND:
0190 0540 INV 0 Invert
0192 4080 SZC 0,2 Execute AND
0194 1049 JMP >0228 and execute

GPL OR:
0196 E080 SOC 0,2 Or
0198 1047 JMP >0228 Execute

GPL XOR:
019A 2880 XOR 0,2 Exclusive OR
019C 1045 JMP >0228 Execute

GPL ST:
019E C080 MOV 0,2 R0 in R2
01A0 1046 JMP >022E Execute

GPL EX:
01A2 C242 MOV 2,9 Save data S
01A4 C080 MOV 0,2 Data D in Data S
01A6 06A0 BL @>0232 D write new data
01A8 0232
01AA 06C4 SWPB 4 Exchange VDP flag
01AC C0C1 MOV 1,3 Exchange address
01AE 101B JMP >01E6 Restore old data and execute

GPL SRA:
01B0 0802 SRA 2,0 Shift to R0
01B2 103D JMP >022E Execute

GPL SLL:
01B4 0A02 SLA 2,0
01B6 103B JMP >022E

GPL SRL:
01B8 D145 MOVB 5,5 No word?
01BA 1601 JNE >01BE
01BC 7082 SB 2,2 Hbyte 0

```

```

01BE 0902 SRL 2,0
01C0 1036 JMP >022E Execute

GPL SRC:
01C2 D145 MOVB 5,5 No word?
01C4 1602 JNE >01CA
01C6 D0A0 MOVB @>83E5,2 Lbyte in Hbyte
01C8 83E5
01CA 0B02 SRC 2,0 Shift
01CC 1030 JMP >022E Execute

GPL MUL:
01CE C202 MOV 2,8 Save data
01D0 D145 MOVB 5,5 Word?
01D2 1601 JNE >01D6
01D4 7208 SB 8,8 Hbyte 0
01D6 3A00 MPY 0,8 Multiply
01D8 D145 MOVB 5,5 Word?
01DA 1602 JNE >01E0
01DC D809 MOVB 9,@>83F1 Hbyte R9 in Lbyte R8
01DE 83F1
01E0 C088 MOV 8,2 New data
01E2 06A0 BL @>0232 Write first data
01E4 0232
01E6 C089 MOV 9,2 Second data in R2
01E8 1022 JMP >022E Execute

GPL DIV:
01EA D805 MOVB 5,@>837C Save R5
01EC 837C
01EE C202 MOV 2,8
01F0 C080 MOV 0,2
01F2 C043 MOV 3,1 S address in 1
01F4 0581 INC 1 Address +1
01F6 D145 MOVB 5,5 Word?
01F8 1301 JEQ >01FC
01FA 0581 INC 1 Address +1
01FC D104 MOVB 4,4 VDP?
01FE 1303 JEQ >0206
0200 06A0 BL @>07FA Write address VDP
0202 07FA
0204 1002 JMP >020A
0206 06A0 BL @>07A8 Fetch data from CPU RAM
0208 07A8
020A C240 MOV 0,9
020C D145 MOVB 5,5 Word?
020E 1603 JNE >0216
0210 D260 MOVB @>83F1,9 Lbyte in Hbyte R9
0212 83F1
0214 0888 SRA 8,8
0216 3E02 DIV 2,8 Division
0218 19E3 JNO >01E0 No overflow, then write
021A F820 SOCB @>0013,@>837C Set overflow in condition bit
021C 0013
021E 837C
0220 10DF JMP >01E0 And write data

0222 B820 AB @>83E1,@>83E5 Lbyte R0 + Lbyte R2
0224 83E1
0226 83E5
0228 02CB STST 11 CPU status becomes
022A D80B MOVB 11,@>837C GPL status
022C 837C
022E 020B LI 11,>0070 Trick return GPL interpreter
0230 0070
0232 D104 MOVB 4,4 VPD address?

```



```

0234 130F JEQ >0254
0236 D7E0 MOVB @>83E7,*15 Write VDP address
0238 83E7
023A 0263 ORI 3,>4000 Write data
023C 4000
023E D7C3 MOVB 3,*15
0240 D145 MOVB 5,5 Word?
0242 1303 JEQ >024A
0244 DBC2 MOVB 2,@>FFFE(15) Data is in VDP RAM
0246 FFFE
0248 0583 INC 3
024A DBE0 MOVB @>83E5,@>FFFE(15)
024C 83E5
024E FFFE
0250 0583 INC 3
0252 045B B *11 To GPL interpreter

0254 D145 MOVB 5,5 Word?
0256 1301 JEQ >025A
0258 DCC2 MOVB 2,*3+ Write data
025A 06C2 SWPB 2
025C DCC2 MOVB 2,*3+
025E 0283 CI 3,>837E Was address screen buffer?
0260 837E
0262 16F7 JNE >0252 No, end
0264 C18B MOV 11,6 Save return
0266 06A0 BL @>0880 Screen address from YPT and XPT
0268 0880
026A DBC2 MOVB 2,@>FFFE(15) Byte on screen
026C FFFE
026E 0456 B *6 To GPL interpreter

```

GPL subinterpreter for >00 through >1F:

```

0270 0A39 SLA 9,3
0272 09A9 SRL 9,10 Table value from command
0274 C129 MOV @>0C3E(9),4 Fetch address from table
0276 0C3E
0278 0454 B *4 And execute

```

GPL RAND:

```

027A 0204 LI 4,>6FE5 Generate random number
027C 6FE5
027E 3920 MPY @>83C0,4
0280 83C0
0282 0225 AI 5,>7AB9
0284 7AB9
0286 C805 MOV 5,@>83C0 Load random number seed
0288 83C0
028A D19D MOVB *13,6 Fetch limit
028C 0986 SRL 6,8 in Lbyte
028E 0586 INC 6 +1
0290 04C4 CLR 4
0292 06C5 SWPB 5
0294 3D06 DIV 6,4
0296 D820 MOVB @>83EB,@>8378 Random number on source
0298 83EB
029A 8378
029C 1006 JMP >02AA To GPL interpreter

```

GPL BACK:

```

029E 0207 LI 7,>8700 Prepare R7 for writing in VDP register 07
02A0 8700
02A2 D81D MOVB *13,@>83EF Fetch colour
02A4 83EF
02A6 06A0 BL @>089A Load register 7
02A8 089A

```

```

02AA 0460 B    @>0070    To GPL interpreter
02AC 0070

```

GPL SCAN (Scan keyboard with return to GPL interpreter):

```

02AE 020B LI   11,>0070    Trick with return
02B0 0070

```

Keyboard scanning

```

02B2 C80B MOV  11,@>83D8    Save R11
02B4 83D8
02B6 06A0 BL   @>0864    Save GROM address to substack
02B8 0864
02BA 04CC CLR  12          CRU 0
02BC 1D15 SBO  >0015    Alpha lock
02BE D160 MOVB @>8374,5    Keyboard mode
02C0 8374
02C2 0985 SRL  5,8
02C4 C185 MOV  5,6
02C6 1312 JEQ  >02EC     0
02C8 0200 LI   0,>0FFF
02CA 0FFF
02CC 0606 DEC  6          1 R0=>0FFF
02CE 1312 JEQ  >02F4
02D0 0200 LI   0,>F0FF
02D2 F0FF
02D4 0606 DEC  6          2 R0=>F0FF
02D6 130E JEQ  >02F4
02D8 0606 DEC  6          3
02DA 8806 C    6,@>0072
02DC 0072
02DE 1B51 JH   >0382    >5?
02E0 D806 MOVB 6,@>8374    Keyboard mode 0
02E2 8374
02E4 06C6 SWPB 6
02E6 D806 MOVB 6,@>83C6    Keyboard debounce (3=0, 4=1, 5=2)
02E8 83C6
02EA 04C5 CLR  5
02EC 04C0 CLR  0
02EE 04C6 CLR  6
02F0 101E JMP  >032E    Mode 0,3,4,5
02F2 2925 DATA >2925
02F4 020C LI   12,>0024    Mode 1,2 scan Joystick
02F6 0024
02F8 30E5 LD CR @>0405(5),3 CRU 06 and 07
02FA 0405
02FC 020C LI   12,>0006
02FE 0006
0300 04C3 CLR  3
0302 0704 SETO 4
0304 3544 ST CR 4,5      Fetch CRU
0306 0994 SRL  4,9
0308 1803 JOC  >0310    No fire key?
030A D825 MOVB @>02F1(5),@>83E7    Lbyte R3
030C 02F1
030E 83E7
0310 0A14 SLA  4,1
0312 0224 AI   4,>16E0    GROM address Joystick table
0314 16E0
0316 DB44 MOVB 4,@>0402(13) Write address
0318 0402
031A DB60 MOVB @>83E9,@>0402(13)
031C 83E9
031E 0402
0320 1000 JMP  >0322
0322 D81D MOVB *13,@>8376    Fetch values Y
0324 8376

```

```

0326 D81D  MOVB *13,@>8377 and X
0328 8377
032A C0C3  MOV  3,3      Fire key ?
032C 163E  JNE  >03AA
032E 0201  LI   1,>0005   Scan keys
0330 0005
0332 04C2  CLR  2
0334 04C7  CLR  7
0336 020C  LI   12,>0024
0338 0024
033A 06C1  SWPB 1
033C 30C1  LD CR 1,3      Load CRU
033E 06C1  SWPB 1
0340 020C  LI   12,>0006
0342 0006
0344 0704  SETO 4
0346 3604  ST CR 4,8      Fetch CRU
0348 0544  INV  4          Invert bits
034A C041  MOV  1,1      0?
034C 1603  JNE  >0354
034E D1C4  MOVB 4,7
0350 0244  ANDI 4,>0F00
0352 0F00
0354 4100  SZC  0,4
0356 1311  JEQ  >037A     No key?
0358 C041  MOV  1,1      Last loop?
035A 1602  JNE  >0360
035C C145  MOV  5,5      Mode 1 or 2?
035E 160D  JNE  >037A
0360 C082  MOV  2,2      Already a key?
0362 160B  JNE  >037A
0364 0702  SETO 2
0366 C0C1  MOV  1,3
0368 0A33  SLA  3,3
036A 0603  DEC  3
036C 0583  INC  3
036E 0A14  SLA  4,1      Built key value in R3
0370 17FD  JNC  >036C
0372 C041  MOV  1,1
0374 1302  JEQ  >037A
0376 0201  LI   1,>0001   Shorten loop
0378 0001
037A 0601  DEC  1
037C 18DC  JOC  >0336     Smaller than 0?
037E C082  MOV  2,2      Key pressed?
0380 1614  JNE  >03AA
0382 04C6  CLR  6          Set pointer for no key
0384 D806  MOVB 6,@>83C7
0386 83C7
0388 0700  SETO 0
038A 9940  CB   0,@>83C8(5)
038C 83C8
038E 1302  JEQ  >0394
0390 06A0  BL   @>0498     Time delay
0392 0498
0394 D800  MOVB 0,@>83C8
0396 83C8
0398 D940  MOVB 0,@>83C8(5)
039A 83C8
039C C145  MOV  5,5      Mode 1 or 2?
039E 166C  JNE  >0478     End
03A0 D800  MOVB 0,@>83C9
03A2 83C9
03A4 D800  MOVB 0,@>83CA
03A6 83CA
03A8 1067  JMP  >0478     End

```

```

03AA 9960 CB    @>83E7,@>83C8(5)  Same key as last time?
03AC 83E7
03AE 83C8
03B0 131A JEQ   >03E6
03B2 0206 LI    6,>2000      Set GPL status
03B4 2000
03B6 06A0 BL    @>0498      Time delay
03B8 0498
03BA D820 MOVB  @>83E7,@>83C8  Set pointer (Lbyte R3)
03BC 83E7
03BE 83C8
03C0 D960 MOVB  @>83E7,@>83C8(5)
03C2 83E7
03C4 83C8
03C6 C145 MOV   5,5          Mode 1 or 2
03C8 160C JNE   >03E2
03CA C303 MOV   3,12
03CC 022C AI    12,>FFF8
03CE FFF8
03D0 1108 JLT   >03E2
03D2 0201 LI    1,>0002
03D4 0002
03D6 093C SRL   12,3
03D8 1801 JOC   >03DC
03DA 0601 DEC   1
03DC D860 MOVB  @>83E7,@>83C8(1)
03DE 83E7
03E0 83C8
03E2 D807 MOVB  7,@>83C7    Last loop scanning on >83C7
03E4 83C7
03E6 D1E0 MOVB  @>83C7,7
03E8 83C7
03EA 0201 LI    1,>17C0    GROM table mode 1 and 2
03EC 17C0
03EE C145 MOV   5,5
03F0 160E JNE   >040E
03F2 0201 LI    1,>1790    GROM table CNTRL
03F4 1790
03F6 0A27 SLA   7,2
03F8 180A JOC   >040E
03FA 0201 LI    1,>1760    GROM table FCTN
03FC 1760
03FE 09F7 SRL   7,15
0400 1806 JOC   >040E
0402 0201 LI    1,>1730    GROM table SHIFT
0404 1730
0406 0607 DEC   7
0408 1302 JEQ   >040E
040A 0201 LI    1,>1700    GROM address table small letters
040C 1700
040E A043 A     3,1          +Key value
0410 DB41 MOVB  1,@>0402(13) Write address
0412 0402
0414 DB60 MOVB  @>83E3,@>0402(13)
0416 83E3
0418 0402
041A 1000 JMP   >041C
041C D01D MOVB  *13,0        Values from GROM
041E C145 MOV   5,5          Mode 1 or 2?
0420 162B JNE   >0478
0422 D820 MOVB  @>83C6,@>83E7 Keyboard mode in R3 Lbyte
0424 83C6
0426 83E7
0428 06A0 BL    @>04A2      Compare
042A 04A2
042C 617A DATA >617A

```

```

042E 160A JNE >0444 Small letter?
0430 04CC CLR 12
0432 C0C3 MOV 3,3 Mode <4
0434 1304 JEQ >043E
0436 1E15 SBZ >0015 Activate ALPHA LOCK
0438 0BEC SRC 12,14
043A 1F07 TB >0007 Scan ALPHA LOCK
043C 1302 JEQ >0442
043E 7020 SB @>03B4,0 ->20 (Capital letter)
0440 03B4
0442 1D15 SBO >0015
0444 C0C3 MOV 3,3 Mode 4 or 5
0446 1607 JNE >0456
0448 06A0 BL @>04A2 Compare
044A 04A2
044C 101F DATA >101F
044E 1399 JEQ >0382 R0 between >10 and >1F
0450 9800 CB 0,@>0587 5F?
0452 0587
0454 1B96 JH >0382
0456 0603 DEC 3 Mode 5
0458 160F JNE >0478
045A 9800 CB 0,@>0025 0D (CR)?
045C 0025
045E 130C JEQ >0478
0460 9800 CB 0,@>02CA 0F?
0462 02CA
0464 1B03 JH >046C
0466 F020 SOCB @>0470,0 Set 1st bit
0468 0470
046A 1006 JMP >0478
046C 06A0 BL @>04A2 Compare
046E 04A2
0470 809F DATA >809F
0472 1602 JNE >0478 R0 smaller than >80 or bigger than >9F
0474 5020 SZCB @>0470,0 Reset 1st bit
0476 0470
0478 D800 MOVB 0,@>8375 ASCII value key on >8375
047A 8375
047C 06A0 BL @>0842 Restore GROM address
047E 0842
0480 D806 MOVB 6,@>837C Set GPL status
0482 837C
0484 1306 JEQ >0492
0486 D7E0 MOVB @>83D4,*15 Load VDP register 01
0488 83D4
048A 04E0 CLR @>83D6 Clear screen timeout counter
048C 83D6
048E D7E0 MOVB @>0B61,*15
0490 0B61
0492 C2E0 MOV @>83D8,11 Fetch return
0494 83D8
0496 045B B *11

Time delay
0498 020C LI 12,>04E2 Loop counter
049A 04E2
049C 060C DEC 12
049E 16FE JNE >049C
04A0 045B B *11

Compare R0 with status EQU if R0 is in area of Hbyte and Lbyte
of the data value
04A2 C33B MOV *11+,12
04A4 9300 CB 0,12
04A6 1A04 JL >04B0

```

```

04A8 9800 CB 0,>83F9
04AA 83F9
04AC 1B01 JH >04B0
04AE 9000 CB 0,0
04B0 045B B *11

```

CLEAR keyboard scanning(status EQU, if pressed):

```

04B2 020C LI 12,>0024 Load CRU keyboard select
04B4 0024
04B6 30E0 LDCR @>0012,3 >00
04B8 0012
04BA 0B7C SRC 12,7
04BC 020C LI 12,>0006
04BE 0006
04C0 360C STCR 12,8 Fetch CRU
04C2 2720 CZC @>0036,12 Right key?
04C4 0036
04C6 160A JNE >04DC
04C8 020C LI 12,>0024 CRU 2nd key
04CA 0024
04CC 30E0 LDCR @>0074,3 >03
04CE 0074
04D0 0B7C SRC 12,7
04D2 020C LI 12,>0006
04D4 0006
04D6 360C STCR 12,8 Fetch CRU
04D8 2720 CZC @>0036,12 Right key
04DA 0036
04DC 045B B *11 Back

```

GPL FMT:

```

04DE 04C9 CLR 9
04E0 04C3 CLR 3
04E2 06A0 BL @>0880 Write actual screen address from XPT and YPT
04E4 0880
04E6 D21D MOVB *13,8 Fetch 1st byte GROM
04E8 020C LI 12,>8373 Pointer substack
04EA 8373
04EC C148 MOV 8,5 Save R8 IN R5
04EE 0A38 SLA 8,3 Eliminate 3 bits
04F0 09B8 SRL 8,11 In Lbyte
04F2 0548 INV 8
04F4 09C5 SRL 5,12 1st Nybble
04F6 C165 MOV @>0CDC(5),5 Fetch routine address
04F8 0CDC
04FA 0202 LI 2,>050A Return address
04FC 050A
04FE 0704 SETO 4 Flag
0500 0455 B *5 Execute routine

0502 0A54 SLA 4,5 Prepare flag

0504 8CB2 C *2+,*2+ New return
0506 1001 JMP >050A

0508 0A54 SLA 4,5 Prepare flag

050A D19D MOVB *13,6 Fetch byte from GROM
050C A183 A 3,6 Add offset
050E DBC6 MOVB 6,@>FFFE(15) Write data
0510 FFFE
0512 61C4 S 4,7
0514 0287 CI 7,>0320
0516 0320
0518 1405 JHE >0524
051A 0287 CI 7,>0300 Address too big?

```

051C	0300			
051E	1A02	JL	>0524	
0520	0227	AI	7,>FD00	Screen address 0
0522	FD00			
0524	06A0	BL	@>05B8	Set YPT and XPT
0526	05B8			
0528	06A0	BL	@>0880	Write screen address
052A	0880			
052C	0588	INC	8	End format command
052E	13DB	JEQ	>04E6	
0530	0452	B	*2	and on
0532	0A58	SLA	8,5	Compute format
0534	61C8	S	8,7	
0536	0708	SETO	8	
0538	10F0	JMP	>051A	
053A	0589	INC	9	
053C	05DC	INCT	*12	Increase substack
053E	D19C	MOVB	*12,6	Fetch last value
0540	0986	SRL	6,8	
0542	D9A0	MOVB	@>83F1,@>8300(6)	R8 Lbyte on Stack
0544	83F1			
0546	8300			
0548	10CE	JMP	>04E6	Next format command
054A	065C	DECT	*12	Decrease substack
054C	0609	DEC	9	
054E	10CB	JMP	>04E6	Next format command
0550	C249	MOV	9,9	End ?
0552	1330	JEQ	>05B4	End with setting new XPT, YPT
0554	D11D	MOVB	*13,4	Next byte from GROM
0556	D19C	MOVB	*12,6	Fetch byte from stack
0558	D15D	MOVB	*13,5	Next byte from GROM
055A	0986	SRL	6,8	
055C	B98E	AB	14,@>8300(6)	Count on stack
055E	8300			
0560	13F4	JEQ	>054A	Go on
0562	DB44	MOVB	4,@>0402(13)	Write GROM address
0564	0402			
0566	DB45	MOVB	5,@>0402(13)	
0568	0402			
056A	10BD	JMP	>04E6	Next format
056C	0288	CI	8,>FFE4	End ? (>1B original)
056E	FFE4			
0570	13EF	JEQ	>0550	
0572	1511	JGT	>0596	
0574	C04D	MOV	13,1	
0576	0288	CI	8,>FFE2	
0578	FFE2			
057A	1309	JEQ	>058E	
057C	150A	JGT	>0592	
057E	06A0	BL	@>05B8	Set YPT and XPT
0580	05B8			
0582	0508	NEG	8	
0584	DA1D	MOVB	*13,@>835F(8)	
0586	835F			
0588	06A0	BL	@>0880	Set screen address
058A	0880			
058C	10AC	JMP	>04E6	Go on
058E	06A0	BL	@>0778	Fetch address
0590	0778			
0592	D0D1	MOVB	*1,3	

```

0594 10A8 JMP >04E6 Go on
0596 06A0 BL @>0778 Fetch address
0598 0778
059A 0202 LI 2,>059E Change return
059C 059E
059E D1B1 MOVB *1+,6
05A0 10B5 JMP >050C Write

```

GPL ALL:

```

05A2 D15D MOVB *13,5 Fetch ASCII
05A4 06A0 BL @>08A4 Write VDP address
05A6 08A4
05A8 0207 LI 7,>0300 Screen
05AA 0300
05AC DBC5 MOVB 5,@>FFFE(15)
05AE FFFE
05B0 0607 DEC 7
05B2 16FC JNE >05AC Loop
05B4 020B LI 11,>0070 GPL return address
05B6 0070
05B8 0A37 SLA 7,3
05BA D807 MOVB 7,@>837E New YPT (0)
05BC 837E
05BE 0A87 SLA 7,8
05C0 0937 SRL 7,3
05C2 D807 MOVB 7,@>837F New XPT (0)
05C4 837F
05C6 045B B *11 and on

```

GPL I/O:

```

05C8 C080 MOV 0,2 Prepare address
05CA C043 MOV 3,1
05CC A082 A 2,2
05CE C122 MOV @>0CEC(2),4 Fetch routine address
05D0 0CEC
05D2 04C9 CLR 9
05D4 0454 B *4 and execute

```

I/O Sound:

```

05D6 024E ANDI 14,>FFFE System flag
05D8 FFFE
05DA E380 SOC 0,14 Pointer GROM or VDP in system flag (00 or 01)
05DC C813 MOV *3,@>83CC Load pointer sound list
05DE 83CC
05E0 D80E MOVB 14,@>83CE Load sound byte
05E2 83CE
05E4 0460 B @>0070 To GPL interpreter
05E6 0070

```

I/O CRU Input:

```

05E8 0589 INC 9 Flag R9

```

I/O CRU Output

```

05EA C331 MOV *1+,12 CRU address
05EC A30C A 12,12 Complete
05EE 04C2 CLR 2
05F0 D0B1 MOVB *1+,2 Number bits in 2
05F2 0A42 SLA 2,4
05F4 E242 SOC 2,9
05F6 0B69 SRC 9,6
05F8 0269 ORI 9,>3012 Prepare command
05FA 3012
05FC D091 MOVB *1,2 Fetch pointer
05FE 06C2 SWPB 2
0600 0222 AI 2,>8300 Complete address
0602 8300

```



```

0604 0489 X 9 Execute
0606 10EE JMP >05E4 To GPL interpreter

GPL XML:
0608 D25D MOVB *13,9 Fetch data
060A C109 MOV 9,4 Prepare register for table access
060C 09C9 SRL 9,12 In R9 2nd table *2
060E 0A19 SLA 9,1
0610 0A44 SLA 4,4 In R4 1st table *2
0612 09B4 SRL 4,11
0614 A129 A @>0CFA(9),4 Fetch table address
0616 0CFA
0618 C114 MOV *4,4 Fetch address routine
061A 0694 BL *4 Execute
061C 10E3 JMP >05E4 To GPL interpreter

```

```

GPL MOVE:
061E D14E MOVB 14,5
0620 0999 SRL 9,9 Test on IMM value
0622 1804 JOC >062C Jump, if number immediate value
0624 06A0 BL @>077A Fetch number
0626 077A
0628 C200 MOV 0,8 In R8
062A 1004 JMP >0634
062C D21D MOVB *13,8 Fetch number
062E 0AF4 SLA 4,15 Time loss
0630 D81D MOVB *13,@>83F1 Complete number in R8
0632 83F1
0634 04C4 CLR 4
0636 0AC9 SLA 9,12
0638 06A0 BL @>0758 Determine destination
063A 0758
063C C081 MOV 1,2
063E B249 AB 9,9
0640 1702 JNC >0646
0642 0224 AI 4,>0003 Set VDP flag
0644 0003
0646 C1C4 MOV 4,7
0648 04C4 CLR 4
064A 06A0 BL @>0758 Determine source
064C 0758
064E A104 A 4,4 R4 *2
0650 C1A4 MOV @>0CCE(4),6 Source routine in R6
0652 0CCE
0654 A1C7 A 7,7
0656 C1E7 MOV @>0CD4(7),7 Destination routine in R7
0658 0CD4
065A 06A0 BL @>0864 Push GROM address on substack
065C 0864
065E 0456 B *6 Execute source

```

```

Source ROM or RAM:
0660 D2F1 MOVB *1+,11 Fetch
0662 0457 B *7 Execute destination

```

```

Source VDP RAM:
0664 D7E0 MOVB @>83E3,*15 Write address
0666 83E3
0668 D7C1 MOVB 1,*15
066A 0581 INC 1
066C D2EF MOVB @>FBFE(15),11 Fetch data
066E FBFE
0670 0457 B *7 Execute destination

```

```

Source GROM:
0672 DB41 MOVB 1,@>0402(13) Write GROM address

```

```

0674 0402
0676 DB60 MOVB @>83E3,@>0402(13)
0678 83E3
067A 0402
067C 0581 INC 1
067E D2DD MOVB *13,11 Fetch data
0680 0457 B *7 Execute destination

```

```

Destination RAM:
0682 DC8B MOVB 11,*2+ Write
0684 1022 JMP >06CA Go on

```

```

Destination GROM:
0686 DB42 MOVB 2,@>0402(13) Write GROM address
0688 0402
068A DB60 MOVB @>83E5,@>0402(13)
068C 83E5
068E 0402
0690 0582 INC 2 Next address
0692 DB4B MOVB 11,@>0400(13) Write into GRAM
0694 0400
0696 1019 JMP >06CA Go on

```

```

Destination VDP register:
0698 93A0 CB @>83E5,14 R2 Lbyte >01?
069A 83E5
069C 1607 JNE >06AC
069E 23A0 COC @>0012,14 Version?
06A0 0012
06A2 1602 JNE >06A8
06A4 026B ORI 11,>8000 Set 16k bit
06A6 8000
06A8 D80B MOVB 11,@>83D4 Register value 1
06AA 83D4
06AC D7CB MOVB 11,*15 Write
06AE 0262 ORI 2,>0080 VDP register
06B0 0080
06B2 D7E0 MOVB @>83E5,*15 Write from Lbyte R2
06B4 83E5
06B6 0582 INC 2 Next register
06B8 1008 JMP >06CA Go on

```

```

Destination VDP RAM:
06BA D7E0 MOVB @>83E5,*15 Write address VDP
06BC 83E5
06BE 0262 ORI 2,>4000 Writing
06C0 4000
06C2 D7C2 MOVB 2,*15
06C4 0582 INC 2 Next address
06C6 DBCB MOVB 11,@>FFFE(15) Write data
06C8 FFFE

```

```

06CA 0608 DEC 8 End ?
06CC 15C8 JGT >065E No, go on
06CE 0460 B @>083E Return GPL interpreter, set condition bit and
06D0 083E GROM address from substack

```

```

GPL COINC:
06D2 C200 MOV 0,8
06D4 C0C8 MOV 8,3
06D6 70C2 SB 2,3 Difference Y
06D8 06C8 SWPB 8
06DA 06C2 SWPB 2
06DC 7202 SB 2,8 Difference X
06DE D01D MOVB *13,0 Fetch mapping value
06E0 0980 SRL 0,8

```

```

06E2 D15D MOVB *13,5      Fetch coincidence table address
06E4 06C5 SWPB 5
06E6 D15D MOVB *13,5
06E8 06C5 SWPB 5
06EA 06A0 BL @>0864      Push GROM address on substack
06EC 0864
06EE DB45 MOVB 5,@>0402(13) Write table address GROM
06F0 0402
06F2 06C5 SWPB 5
06F4 DB45 MOVB 5,@>0402(13)
06F6 0402
06F8 06C5 SWPB 5
06FA D09D MOVB *13,2      Fetch data from table
06FC 1000 JMP >06FE
06FE D05D MOVB *13,1
0700 1000 JMP >0702
0702 D19D MOVB *13,6
0704 1000 JMP >0706
0706 D1DD MOVB *13,7
0708 C000 MOV 0,0          Mapping 0?
070A 1302 JEQ >0710
070C 0803 SRA 3,0         Seek coincidence
070E 0808 SRA 8,0
0710 B207 AB 7,8
0712 111E JLT >0750
0714 B0C6 AB 6,3
0716 111C JLT >0750
0718 9083 CB 3,2
071A 151A JGT >0750
071C 9048 CB 8,1
071E 1518 JGT >0750
0720 0981 SRL 1,8         Which table value is needed?
0722 0581 INC 1
0724 0983 SRL 3,8
0726 3843 MPY 3,1
0728 0988 SRL 8,8
072A A088 A 8,2
072C C002 MOV 2,0
072E 0242 ANDI 2,>FFF8
0730 FFF8
0732 6002 S 2,0
0734 0832 SRA 2,3
0736 A085 A 5,2
0738 8CB2 C *2+,*2+
073A DB42 MOVB 2,@>0402(13) Write GROM address of the right data
073C 0402
073E 0580 INC 0
0740 DB60 MOVB @>83E5,@>0402(13)
0742 83E5
0744 0402
0746 0202 LI 2,>2000
0748 2000
074A D0DD MOVB *13,3      Fetch value
074C 0A03 SLA 3,0         Coincidence?
074E 1801 JOC >0752
0750 04C2 CLR 2
0752 D802 MOVB 2,@>837C  Set GPL status
0754 837C
0756 10BB JMP >06CE      Return GPL interpreter with POP GROM address from
substack

0758 B249 AB 9,9          Check kind
075A 180F JOC >077A
075C D0DD MOVB *13,3      Fetch destination in R3
075E 0584 INC 4           Flag
0760 D81D MOVB *13,@>83E7

```

```

0762 83E7
0764 C30B MOV 11,12      Save return
0766 B249 AB 9,9
0768 1704 JNC >0772
076A D05D MOVB *13,1     Fetch byte from GROM
076C 06A0 BL @>077E
076E 077E
0770 A0C0 A 0,3
0772 C043 MOV 3,1
0774 0919 SRL 9,1
0776 045C B *12        Return

0778 04C5 CLR 5

```

GPL addressing modes(fetch address and data):

```

077A D05D MOVB *13,1     Fetch GPL byte
077C 111E JLT >07BA     Negative? Jump at format II through V
077E 0981 SRL 1,8
0780 0221 AI 1,>8300    Scratch pad address
0782 8300
0784 0281 CI 1,>837D    Character buffer?
0786 837D
0788 160F JNE >07A8     General
078A 04CA CLR 10
078C C18B MOV 11,6
078E 06A0 BL @>0884     Write address screen
0790 0884
0792 C2C6 MOV 6,11
0794 D02F MOVB @>FBFE(15),0 Fetch byte
0796 FBFE
0798 23A0 COC @>0072,14 Check multicolor
079A 0072
079C 1603 JNE >07A4
079E 1701 JNC >07A2
07A0 0A40 SLA 0,4       Prepare multicolor
07A2 0940 SRL 0,4
07A4 D800 MOVB 0,@>837D in character buffer
07A6 837D
07A8 D011 MOVB *1,0      Fetch data from CPU
07AA D145 MOVB 5,5       Word?
07AC 1602 JNE >07B2
07AE 0880 SRA 0,8
07B0 045B B *11        Return

```

```

07B2 D821 MOVB @>0001(1),@>83E1 Fetch 2nd byte
07B4 0001
07B6 83E1
07B8 045B B *11

```

GPL addressing modes II through V:

```

07BA D81D MOVB *13,@>83E3
07BC 83E3
07BE C281 MOV 1,10
07C0 0241 ANDI 1,>0FFF
07C2 0FFF
07C4 0281 CI 1,>0F00    Extended range?
07C6 0F00
07C8 1103 JLT >07D0     No, jump
07CA 0A81 SLA 1,8
07CC D81D MOVB *13,@>83E3 R1 address extended range
07CE 83E3
07D0 0A2A SLA 10,2      Test 2nd bit(Mode II and IV)
07D2 1708 JNC >07E4     No, jump
07D4 D19D MOVB *13,6     Fetch index
07D6 0986 SRL 6,8
07D8 D026 MOVB @>8300(6),0 Data in R0

```

```

07DA 8300
07DC D826 MOVB @>8301(6),@>83E1
07DE 8301
07E0 83E1
07E2 A040 A 0,1 R1 = Indicated address
07E4 0A1A SLA 10,1 Test VDP RAM
07E6 1715 JNC >0812 No, jump
07E8 05C4 INCT 4 Flag VDP
07EA 0A1A SLA 10,1 Test indirect
07EC 1706 JNC >07FA No, jump
07EE D021 MOVB @>8300(1),0 Fetch value
07F0 8300
07F2 D821 MOVB @>8301(1),@>83E1
07F4 8301
07F6 83E1
07F8 C040 MOV 0,1 Value in R1
07FA D7E0 MOVB @>83E3,*15 Write address VDP
07FC 83E3
07FE D7C1 MOVB 1,*15
0800 0A80 SLA 0,8
0802 D02F MOVB @>FBFE(15),0 Data in R0
0804 FBFE
0806 D145 MOVB 5,5 Word?
0808 13D2 JEQ >07AE
080A D82F MOVB @>FBFE(15),@>83E1 2nd byte in R0
080C FBFE
080E 83E1
0810 045B B *11 Return

0812 0A1A SLA 10,1 Test indirect
0814 17B5 JNC >0780 Jump if direct
0816 0281 CI 1,>007C GPL statusbyte?
0818 007C
081A 1605 JNE >0826
081C D060 MOVB @>8372,1 Fetch data stack pointer
081E 8372
0820 780E SB 14,@>8372 Decrease
0822 8372
0824 10AC JMP >077E Go on
0826 D061 MOVB @>8300(1),1 Fetch from CPU RAM
0828 8300
082A 10A9 JMP >077E Go on

GPL RTGR:
082C 06A0 BL @>0842 POP GROM address from substack
082E 0842
0830 C364 MOV @>8300(4),13 Data substack new GRMRD
0832 8300
0834 DB44 MOVB 4,@>0400(13) Write stack value in new GROM
0836 0400

GPL RTN:
0838 5820 SZCB @>011B,@>837C Reset condition bit
083A 011B
083C 837C

GPL RTNC:
083E 020B LI 11,>0070 Return GPL interpreter
0840 0070

POP GROM address from substack and write address:
0842 D120 MOVB @>8373,4 GROM address from subroutine stack
0844 8373
0846 0984 SRL 4,8
0848 0660 DECT @>8373 Decrease stack pointer
084A 8373

```

```

084C DB64 MOVB @>8300(4),@>0402(13) Write GROM address
084E 8300
0850 0402
0852 DB64 MOVB @>8301(4),@>0402(13)
0854 8301
0856 0402
0858 045B B *11 Return
085A D19D MOVB *13,6 Reading from GROM
085C 020B LI 11,>0060 Trick return with writing GROM address from R6
085E 0060
0860 D81D MOVB *13,@>83ED R6 complete
0862 83ED

```

```

PUSH actual GROM address on subroutine stack
0864 05E0 INCT @>8373 Increase stack pointer
0866 8373
0868 D120 MOVB @>8373,4
086A 8373
086C 0984 SRL 4,8
086E D92D MOVB @>0002(13),@>8300(4) Address GROM on stack
0870 0002
0872 8300
0874 D92D MOVB @>0002(13),@>8301(4)
0876 0002
0878 8301
087A 0624 DEC @>8300(4) Correct address
087C 8300
087E 045B B *11 Return

```

```

Screen address from YPT and XPT for writing:
0880 020A LI 10,>4000 Write
0882 4000

```

```

Screen address from YPT and XPT for reading:
0884 D1E0 MOVB @>837F,7 Screen line
0886 837F
0888 23A0 COC @>0072,14 Check multicolor
088A 0072
088C 130E JEQ >08AA
088E 0A37 SLA 7,3
0890 0987 SRL 7,8
0892 D1E0 MOVB @>837E,7 Fetch row
0894 837E
0896 0937 SRL 7,3
0898 A1CA A 10,7
089A D7E0 MOVB @>83EF,*15 Write address VDP
089C 83EF
089E D7C7 MOVB 7,*15
08A0 61CA S 10,7
08A2 045B B *11 Return

```

```

Write on screen at 0
08A4 0207 LI 7,>4000
08A6 4000
08A8 10F8 JMP >089A

```

```

Prepare address for multicolor mode:
08AA D020 MOVB @>837E,0
08AC 837E
08AE C200 MOV 0,8 Prepare address:
08B0 0A58 SLA 8,5
08B2 09D8 SRL 8,13
08B4 09B0 SRL 0,11
08B6 0A80 SLA 0,8
08B8 A008 A 8,0
08BA C207 MOV 7,8

```

```

08BC 0247  ANDI 7,>3E00
08BE 3E00
08C0 0967  SRL 7,6
08C2 A1C0  A 0,7
08C4 0227  AI 7,>0800
08C6 0800
08C8 D7E0  MOVB @>83EF,*15 Write address
08CA 83EF
08CC 0A88  SLA 8,8
08CE D7C7  MOVB 7,*15
08D0 028B  CI 11,>026A Back if not >026A (DIV) return address
08D2 026A
08D4 16E6  JNE >08A2
08D6 D02F  MOVB @>FBFE(15),0 Read Data from VDP in R0
08D8 FBFE
08DA D220  MOVB @>837D,8 Character buffer in R8
08DC 837D
08DE 0248  ANDI 8,>0F00
08E0 0F00
08E2 1804  JOC >08EC
08E4 0240  ANDI 0,>0F00
08E6 0F00
08E8 0A48  SLA 8,4
08EA 1002  JMP >08F0
08EC 0240  ANDI 0,>F000
08EE F000
08F0 0267  ORI 7,>4000 Writing address
08F2 4000
08F4 06A0  BL @>089A Write address
08F6 089A
08F8 A008  A 8,0
08FA DBC0  MOVB 0,@>FFFE(15) Write data
08FC FFFE
08FE 0456  B *6 Return

```

Interrupt routine

```

0900 0300  LIMI >0000 Disable interrupt
0902 0000
0904 02E0  LWPI >83E0 Load GPLWS!
0906 83E0
0908 04CC  CLR 12 Clear CRU
090A 23A0  COC @>0032,14 Cassette interrupt?
090C 0032
090E 1602  JNE >0914 No, jump
0910 0460  B @>1404
0912 1404

0914 1F02  TB >0002
0916 1619  JNE >094A Jump, if VDP interrupt
0918 020C  LI 12,>0F00 Clear CRU
091A 0F00
091C 1D01  SBO >0001
091E 1E00  SBZ >0000
0920 022C  AI 12,>0100
0922 0100
0924 028C  CI 12,>2000
0926 2000
0928 130E  JEQ >0946 End CRU
092A 1D00  SBO >0000
092C 9820  CB @>4000,@>000D ROM exists
092E 4000
0930 000D
0932 16F5  JNE >091E No, next
0934 C0A0  MOV @>400C,2 Intlnk?
0936 400C
0938 13F2  JEQ >091E No, next ROM

```

```

093A C002 MOV 2,0
093C C0A2 MOV @>0002(2),2 Fetch INT address
093E 0002
0940 0692 BL *2 And execute
0942 C090 MOV *0,2 Next Int routine
0944 10F9 JMP >0938
0946 0460 B @>0AB8 End interrupt from CRU
0948 0AB8

094A 1D02 SBO >0002 Clear VDP interrupt
094C D060 MOVB @>83C2,1 Fetch interrupt flag byte
094E 83C2
0950 0A11 SLA 1,1 No interrupt permitted
0952 1702 JNC >0958
0954 0460 B @>0A84 Then jump
0956 0A84

0958 0A11 SLA 1,1
095A 1846 JOC >09E8 No sprite move permitted, then jump
095C D320 MOVB @>837A,12 Number sprites
095E 837A
0960 1343 JEQ >09E8 No sprite end
0962 098C SRL 12,8
0964 0202 LI 2,>8800 VDP RD
0966 8800
0968 0203 LI 3,>8C00 VDP WD
096A 8C00
096C 0208 LI 8,>0780 Sprite motion table
096E 0780
0970 D7E0 MOVB @>83F1,*15 Write address motion table
0972 83F1
0974 D7C8 MOVB 8,*15
0976 04C4 CLR 4
0978 D112 MOVB *2,4 Datas Y velocity
097A 04C6 CLR 6
097C D192 MOVB *2,6 Datas X velocity
097E 0844 SRA 4,4
0980 D152 MOVB *2,5 Auxiliary datas
0982 0845 SRA 5,4
0984 A144 A 4,5
0986 D1D2 MOVB *2,7
0988 0846 SRA 6,4
098A 0847 SRA 7,4
098C A1C6 A 6,7
098E 0228 AI 8,>FB80 Address sprite descriptor table
0990 FB80
0992 D7E0 MOVB @>83F1,*15 Write address
0994 83F1
0996 D7C8 MOVB 8,*15
0998 04C4 CLR 4
099A D112 MOVB *2,4 Fetch position
099C A105 A 5,4
099E 0284 CI 4,>C0FF
09A0 C0FF
09A2 1209 JLE >09B6
09A4 0284 CI 4,>E000 Compute new position
09A6 E000
09A8 1B06 JH >09B6
09AA C145 MOV 5,5
09AC 1502 JGT >09B2
09AE 0224 AI 4,>C000
09B0 C000
09B2 0224 AI 4,>2000
09B4 2000
09B6 04C6 CLR 6
09B8 D192 MOVB *2,6

```


09BA	A187	A	7,6	
09BC	0268	ORI	8,>4000	VDP address for writing
09BE	4000			
09C0	D7E0	MOVB	@>83F1,*15	
09C2	83F1			
09C4	D7C8	MOVB	8,*15	
09C6	D4C4	MOVB	4,*3	Write positions
09C8	0228	AI	8,>0482	
09CA	0482			
09CC	D4C6	MOVB	6,*3	
09CE	06C5	SWPB	5	
09D0	D7E0	MOVB	@>83F1,*15	Write address motion table
09D2	83F1			
09D4	D7C8	MOVB	8,*15	
09D6	0945	SRL	5,4	
09D8	D4C5	MOVB	5,*3	Write auxiliary values
09DA	06C7	SWPB	7	
09DC	0947	SRL	7,4	
09DE	D4C7	MOVB	7,*3	
09E0	0228	AI	8,>C002	New address motion table
09E2	C002			
09E4	060C	DEC	12	Last sprite?
09E6	15C4	JGT	>0970	No, once again
09E8	0A11	SLA	1,1	
09EA	183D	JOC	>0A66	No sound process jump
09EC	D0A0	MOVB	@>83CE,2	Number of sound byte
09EE	83CE			
09F0	133A	JEQ	>0A66	None, then end
09F2	780E	SB	14,@>83CE	-1
09F4	83CE			
09F6	1637	JNE	>0A66	Not 0, then end
09F8	C0E0	MOV	@>83CC,3	Pointer sound list
09FA	83CC			
09FC	C14E	MOV	14,5	
09FE	0915	SRL	5,1	GROM or VDP?
0A00	180A	JOC	>0A16	1=VDP, then jump
0A02	06A0	BL	@>0864	Push GROM address on substack
0A04	0864			
0A06	0205	LI	5,>0402	
0A08	0402			
0A0A	A14D	A	13,5	GROM write address
0A0C	D543	MOVB	3,*5	Write GROM address
0A0E	D560	MOVB	@>83E7,*5	
0A10	83E7			
0A12	C18D	MOV	13,6	Read address
0A14	1007	JMP	>0A24	
0A16	0205	LI	5,>8C02	VDPWA
0A18	8C02			
0A1A	D560	MOVB	@>83E7,*5	Write VDP address
0A1C	83E7			
0A1E	D543	MOVB	3,*5	
0A20	0206	LI	6,>8800	VDPRD
0A22	8800			
0A24	D216	MOVB	*6,8	Fetch byte
0A26	130F	JEQ	>0A46	0?
0A28	9220	CB	@>0A9C,8	
0A2A	0A9C			
0A2C	130A	JEQ	>0A42	>FF? Yes,switch to another(well possible)!
0A2E	0988	SRL	8,8	Number
0A30	A0C8	A	8,3	To address
0A32	D816	MOVB	*6,@>8400	Load sound process
0A34	8400			
0A36	0608	DEC	8	How many bytes?
0A38	16FC	JNE	>0A32	Next byte
0A3A	05C3	INCT	3	
0A3C	D096	MOVB	*6,2	Fetch duration

```

0A3E 1309 JEQ >0A52
0A40 1009 JMP >0A54 Go on
0A42 2BA0 XOR @>0378,14 Change system flags
0A44 0378
0A46 D0D6 MOVB *6,3 Fetch new address
0A48 0202 LI 2,>0100 Sound byte >01
0A4A 0100
0A4C D816 MOVB *6,@>83E7 Complete address
0A4E 83E7
0A50 1001 JMP >0A54 Once again
0A52 7082 SB 2,2
0A54 C803 MOV 3,@>83CC New pointer sound list
0A56 83CC
0A58 D802 MOVB 2,@>83CE Sound byte
0A5A 83CE
0A5C 0285 CI 5,>8C02 From VDP?
0A5E 8C02
0A60 1302 JEQ >0A66
0A62 06A0 BL @>0842 POP GROM address from substack
0A64 0842
0A66 0A11 SLA 1,1
0A68 180D JOC >0A84 No QUIT key, then jump
0A6A 020C LI 12,>0024 Load CRU
0A6C 0024
0A6E 30E0 LDCR @>0012,3
0A70 0012
0A72 0B7C SRC 12,7
0A74 020C LI 12,>0006
0A76 0006
0A78 3605 STCR 5,8 Fetch CRU
0A7A 2560 CZC @>004C,5 QUIT key?
0A7C 004C
0A7E 1602 JNE >0A84
0A80 0420 BLWP @>0000 Software reset
0A82 0000
0A84 D82F MOVB @>FC00(15),@>837B VDP status in copy RAM
0A86 FC00
0A88 837B
0A8A 02E0 LWPI >83C0 INTWS
0A8C 83C0
0A8E 05CB INCT 11 Screen timeout counter
0A90 160B JNE >0AA8 Not 0
Interrupt level 2:
0A92 D30A MOVB 10,12 VDP register 1
0A94 098C SRL 12,8
0A96 026C ORI 12,>8160 Basis value
0A98 8160
0A9A 024C ANDI 12,>FFBF Turn off screen
0A9C FFBF
0A9E D820 MOVB @>83D9,@>8C02 Load VDP register
0AA0 83D9
0AA2 8C02
0AA4 D80C MOVB 12,@>8C02
0AA6 8C02
0AA8 02E0 LWPI >83E0 GPLWS
0AAA 83E0
0AAC B80E AB 14,@>8379 VDP interrupt timer (system flags!)
0AAE 8379
0AB0 C320 MOV @>83C4,12 User defined interrupt
0AB2 83C4
0AB4 1301 JEQ >0AB8 None, then jump
0AB6 069C BL *12 Otherwise execute
0AB8 04C8 CLR 8 Clear GROM search pointer
0ABA 02E0 LWPI >83C0 INTWS
0ABC 83C0
0ABE 0380 RTWP And end interrupt

```

XML 19 GROM DSRLNK, similar to assembler, >836D data, but correct pointers are missing >8356 (left pointing DSR name), return to GPL occurs.

```
0AC0 04C1 CLR 1
0AC2 C320 MOV @>83D0,12 Fetch GROM search routine (CRU!)
0AC4 83D0
0AC6 1618 JNE >0AF8
0AC8 020C LI 12,>0F00 Scan CRU
0ACA 0F00
0ACC C30C MOV 12,12
0ACE 1301 JEQ >0AD2
0AD0 1E00 SBZ >0000
0AD2 022C AI 12,>0100
0AD4 0100
0AD6 04E0 CLR @>83D0
0AD8 83D0
0ADA 028C CI 12,>2000
0ADC 2000
0ADE 1320 JEQ >0B20
0AE0 C80C MOV 12,@>83D0
0AE2 83D0
0AE4 1D00 SBO >0000
0AE6 0202 LI 2,>4000 Does ROM exist?
0AE8 4000
0AEA 9812 CB *2,@>000D >AA?
0AEC 000D
0AEE 16EE JNE >0ACC
0AF0 B820 AB @>836D,@>83E5 Add data R2 Lbyte
0AF2 836D
0AF4 83E5
0AF6 1003 JMP >0AFE
0AF8 C0A0 MOV @>83D2,2 Fetch ROM search pointer
0AFA 83D2
0AFC 1D00 SBO >0000
0AFE C092 MOV *2,2 Check routine existing
0B00 13E5 JEQ >0ACC
0B02 C802 MOV 2,@>83D2 ROM pointer next routine
0B04 83D2
0B06 05C2 INCT 2
0B08 C272 MOV *2+,9
0B0A 06A0 BL @>0BE8 Check name
0B0C 0BE8
0B0E 10F4 JMP >0AF8 Not the right
0B10 0581 INC 1
0B12 0699 BL *9 Execute routine
0B14 10F1 JMP >0AF8
0B16 1E00 SBZ >0000 Turn off DSR ROM
0B18 1001 JMP >0B1C
0B1A 04D8 CLR *8
0B1C 06A0 BL @>0842 POP GROM address from substack! corresponds RTN
0B1E 0842
0B20 0460 B @>006A Return GPL status reset
0B22 006A
```

XML 1A GSRLNK (Search DSR in GROM):

```
0B24 0207 LI 7,>83D2 ROM search pointer
0B26 83D2
0B28 0208 LI 8,>83D0 GROM search pointer
0B2A 83D0
0B2C 06A0 BL @>0864 Push GROM address on substack
0B2E 0864
0B30 C057 MOV *7,1
0B32 C098 MOV *8,2
0B34 1604 JNE >0B3E GROM search pointer <>0, then execution
0B36 0202 LI 2,>9800 GROM read data
0B38 9800
```

```

0B3A 0201 LI 1,>E000 Highest GROM
0B3C E000
0B3E 2460 CZC @>0128,1 Beginning
0B40 0128
0B42 160E JNE >0B60
0B44 C602 MOV 2,*8 GROM search pointer
0B46 D881 MOVB 1,@>0402(2) Write GROM address
0B48 0402
0B4A D8A0 MOVB @>83E3,@>0402(2)
0B4C 83E3
0B4E 0402
0B50 B820 AB @>836D,@>83E3 Data + R1 LB
0B52 836D
0B54 83E3
0B56 D801 MOVB 1,@>83CB Save R1 Hbyte
0B58 83CB
0B5A 9812 CB *2,@>000D GROM header?
0B5C 000D
0B5E 1632 JNE >0BC4
0B60 D881 MOVB 1,@>0402(2) Write GROM LINK address
0B62 0402
0B64 D8A0 MOVB @>83E3,@>0402(2)
0B66 83E3
0B68 0402
0B6A 0A4A SLA 10,4 Time loss
0B6C D0D2 MOVB *2,3 Fetch LINK table address
0B6E 1000 JMP >0B70
0B70 D812 MOVB *2,@>83E7
0B72 83E7
0B74 C5C3 MOV 3,*7 R3 on ROM search pointer (next LINK address)
0B76 1326 JEQ >0BC4 0?
0B78 05C3 INCT 3
0B7A D883 MOVB 3,@>0402(2) Write start address
0B7C 0402
0B7E D8A0 MOVB @>83E7,@>0402(2)
0B80 83E7
0B82 0402
0B84 1000 JMP >0B86
0B86 D252 MOVB *2,9 Start address in R9
0B88 0A4A SLA 10,4
0B8A D812 MOVB *2,@>83F3
0B8C 83F3
0B8E 06A0 BL @>0BE8 Check name
0B90 0BE8
0B92 10CE JMP >0B30
0B94 B820 AB @>0030,@>8372 Data stack pointer +2
0B96 0030
0B98 8372
0B9A B80E AB 14,@>836C Count
0B9C 836C
0B9E D120 MOVB @>8372,4 Fetch data stack pointer
0BA0 8372
0BA2 0984 SRL 4,8
0BA4 0643 DECT 3
0BA6 9820 CB @>836D,@>0C04 Does program LINK?
0BA8 836D
0BAA 0C04
0BAC 1601 JNE >0BB0
0BAE C243 MOV 3,9 Addresses of the programs on stack
0BB0 D909 MOVB 9,@>8300(4)
0BB2 8300
0BB4 D920 MOVB @>83F3,@>8301(4)
0BB6 83F3
0BB8 8301
0BBA C342 MOV 2,13 Set GROM pointer
0BBC 06A0 BL @>0842 POP GROM address from substack. Corresbonds RTN

```

```

0BBE 0842
0BC0 0460 B @>00CE Set GPL interpreter condition bit
0BC2 00CE

0BC4 04C1 CLR 1
0BC6 D060 MOVB @>83CB,1 Fetch GROM number
0BC8 83CB
0BCA 0221 AI 1,>E000 ->10
0BCC E000
0BCE C5C1 MOV 1,*7 New GROM on GROM search pointer
0BD0 0281 CI 1,>E000 End ?
0BD2 E000
0BD4 16B4 JNE >0B3E
0BD6 8CB2 C *2+,*2+ R2 +4 Oh !, Works for differentiated GRMRD of
0BD8 C602 MOV 2,*8 >04 each difference! But not supported by
0BDA 0282 CI 2,>9840 16times console.
0BDC 9840
0BDE 139D JEQ >0B1A End
0BE0 D160 MOVB @>8355,5 Length 0?
0BE2 8355
0BE4 16AA JNE >0B3A No
0BE6 109A JMP >0B1C Go on

```

```

Check name ( name on FAC, Length on >8355)
0BE8 D160 MOVB @>8355,5 Length 0?
0BEA 8355
0BEC 130D JEQ >0C08
0BEE 9485 CB 5,*2 Length right ?
0BF0 160C JNE >0C0A
0BF2 0985 SRL 5,8
0BF4 0206 LI 6,>834A FAC
0BF6 834A
0BF8 0282 CI 2,>9800 In GROM?
0BFA 9800
0BFC 1401 JHE >0C00
0BFE 0582 INC 2 No, R2+1
0C00 94B6 CB *6+,*2 Compare
0C02 1603 JNE >0C0A Don't fit, end
0C04 0605 DEC 5 Length complete ?
0C06 16F8 JNE >0BF8 No, go on
0C08 05CB INCT 11 Yes, right name, return +2
0C0A 045B B *11

```

```

GPL extension for future (>14->1E,>98->9F,>EE->EF,>FC->FF):
0C0C 06A0 BL @>0C28 Turn on CRU
0C0E 0C28
0C10 0460 B @>4020 Jump to entry address
0C12 4020

```

```

GPL extension for future (>1F):
0C14 06A0 BL @>0C28 Turn on CRU
0C16 0C28
0C18 0460 B @>401C Execute
0C1A 401C

```

```

Not used up to now in operating system:
0C1C 02E0 LWPI >2800
0C1E 2800
0C20 06A0 BL @>0C28 Turn on CRU
0C22 0C28
0C24 0460 B @>4028 Execute
0C26 4028

0C28 020C LI 12,>1B00 Load CRU
0C2A 1B00
0C2C 1D00 SBO >0000 Turn on

```

0C2E 045B B *11 Return

0C30 0000

0C32 0000

0C34 0000

GPL jump table 1st byte HNybble

0C36 0270 Various (>00->1F)

0C38 061E MOVE (>20->3F)

0C3A 011A BR (>40->5F)

0C3C 010E BS (>60->7F)

GPL jump table Code >00->1F

0C3E 0838 RTN (>00)

0C40 083E RTNC (>01)

0C42 027A RAND (>02)

0C44 02AE SCAN (>03)

0C46 029E BACK (>04)

0C48 0104 B (>05)

0C4A 085A CALL (>06)

0C4C 05A2 ALL (>07)

0C4E 04DE FMT (>08)

0C50 00F4 H (>09)

0C52 00F4 GT (>0A)

0C54 0024 EXIT (>0B)

0C56 00F4 CARRY (>0C)

0C58 00F4 OVF (>0D)

0C5A 18C8 PARSE (>0E)

0C5C 0608 XML (>0F)

0C5E 1920 CONT (>10)

0C60 1968 EXEC (>11)

0C62 19F0 RTNB (>12)

0C64 082C RTGR (>13)

0C66 0C0C For extension

0C68 0C0C "

0C6A 0C0C "

0C6C 0C0C "

0C6E 0C0C "

0C70 0C0C "

0C72 0C0C "

0C74 0C0C "

0C76 0C0C "

0C78 0C0C "

0C7A 0C0C "

0C7C 0C14 " (>1F)

GPL jump table Code >80->9F

0C7E 0136 ABS (>80)

0C80 013A NEG (>82)

0C82 0140 INV (>84)

0C84 013E CLR (>86)

0C86 0144 FETCH (>88)

0C88 0162 CASE (>8A)

0C8A 016E PUSH (>8C)

0C8C 00EA CZ (>8E)

0C8E 0186 INC (>90)

0C90 0188 DEC (>92)

0C92 0184 INCT (>94)

0C94 0182 DECT (>96)

0C96 0C0C For extension

0C98 0C0C "

0C9A 0C0C "

0C9C 0C0C "

GPL jump table Code >A0->FF

0C9E 0188 ADD (>A0)

0CA0	0186	SUB (>A4)
0CA2	01CE	MUL (>A8)
0CA4	01EA	DIV (>AC)
0CA6	0190	AND (>B0)
0CA8	0196	OR (>B4)
0CAA	019A	XOR (>B8)
0CAC	019E	ST (>BC)
0CAE	01A2	EX (>C0)
0CB0	00D6	CH (>C4)
0CB2	00DA	CHE (>C8)
0CB4	00DE	CGT (>CC)
0CB6	00CC	CGE (>D0)
0CB8	00EC	CEQ (>D4)
0CBA	00E2	CLOG (>D8)
0CBC	01B0	SRA (>DC)
0CBE	01B4	SLL (>E0)
0CC0	01B8	SRL (>E4)
0CC2	01C2	SRC (>E8)
0CC4	06D2	COINC (>ED, Incompletely decoded)
0CC6	0C0C	For extension (>F0)
0CC8	05C8	I/O (>F6, Incompletely decoded!)
0CCA	004E	SWGR (>F8)
0CCC	0C0C	For extension (>FC)

Jump table for addresses at MOVE:

0CCE	0660	Source in ROM or RAM
0CD0	0672	Source in GROM or GRAM
0CD2	0664	Source in VDP RAM
0CD4	0682	Destination in ROM or RAM
0CD6	0686	Destination in GROM
0CD8	06BA	Destination in VDP RAM
0CDA	0698	Destination is VDP register

FMT format jump table

0CDC	050A	0,1 Horizontal string projection
0CDE	0508	2,3 Vertical string projection
0CE0	0504	4,5 Repeat horizontal character
0CE2	0502	6,7 Repeat vertical character
0CE4	0534	8,9 Relative fixed row
0CE6	0532	A,B Relative fixed column
0CE8	053A	C,D Loop values
0CEA	056C	E,F Fixed position row and column, screen offset

I/O jump table

0CEC	05D6	Sound Grom	00
0CEE	05D6	Sound VDP	01
0CF0	05E8	CRU Input	02
0CF2	05EA	CRU Output	03
0CF4	1346	Cassette Write	04
0CF6	142E	Cassette Read	05
0CF8	1426	Cassette verify	06

XMLLNK table 1st Nybble

0CFA	0D1A	Floating point routines (>0X)	
0CFC	12A0	„XTAB“ (>1X)	
0CFE	2000	Low memory expansion (>2X)	
0D00	3FC0	Basic enhancement (>3X)	
0D02	3FE0	Basic enhancement (>4X)	
0D04	4010	Probably for GPL extension (>5X)	Also usable in DSR
0D06	4030	Probably for GPL extension (>6X)	Also usable in DSR
0D08	6010	ROM modul (>7X)	
0D0A	6030	ROM modul (>8X)	
0D0C	7000	ROM modul (>9X)	
0D0E	8000	Future expansion (>AX)	
0D10	A000	(>BX)	
0D12	B000	(>CX)	

```

0D14 C000 (>DX)
0D16 D000 (>EX)
0D18 8300 Scratch PAD RAM (>FX)

```

FLTAB (XMLLNK 2nd Nybble >0X)

```

0D1A 0000
0D1C 0F54 Rounding of floating point numbers 9 bytes long (>01)
0D1E 0FB2 Rounding of floating point numbers length @>8354 (>02)
0D20 0FA4 Status EQU if FAC (word) =0 (>03)
0D22 0FC2 Overflow (>8376 Byte negative toward 0 positive toward
infinite(>04)
0D24 0FCC Set overflow number (>8375 negative or positive (>05)
0D26 0D80 FADD (>06)
0D28 0D7C FSUB (>07)
0D2A 0E88 FMUL (>08)
0D2C 0FF4 FDIV (>09)
0D2E 0D3A FCOMP (>0A)
0D30 0D84 SADD (>0B)
0D32 0D74 SSUB (>0C)
0D34 0E8C SMULT (>0D)
0D36 0FF8 SDIV (>0E)
0D38 0D46 SCOMP (>0F)

```

FCOMP (XML >0A):

```

0D3A C28B MOV 11,10
0D3C 0203 LI 3,>0FAA End load
0D3E 0FAA
0D40 1007 JMP >0D50 Execute

```

Set SCOMP with direct return without GPL status:

```

0D42 C0CB MOV 11,3
0D44 1003 JMP >0D4C

```

SCOMP (XML >0F):

```

0D46 0203 LI 3,>0FAA
0D48 0FAA
0D4A C28B MOV 11,10
0D4C 06A0 BL @>1FA8 Fetch number from stack of VDP
0D4E 1FA8
0D50 0207 LI 7,>835C ARG
0D52 835C
0D54 0205 LI 5,>834A FAC
0D56 834A
0D58 8D57 C *7,*5+ Compare 1st word
0D5A 160B JNE >0D72
0D5C C1B7 MOV *7+,6
0D5E 1309 JEQ >0D72 0?
0D60 1503 JGT >0D68 >0?
0D62 C185 MOV 5,6 Exchange if smaller 0
0D64 C147 MOV 7,5 (Invert logic)
0D66 C1C6 MOV 6,7
0D68 8D77 C *7+,*5+ 2nd word
0D6A 1603 JNE >0D72
0D6C 8D77 C *7+,*5+ 3rd word
0D6E 1601 JNE >0D72
0D70 8557 C *7,*5 4th word
0D72 0453 B *3 Return or set GPL status byte

```

SSUB (XML >0C):

```

0D74 C28B MOV 11,10
0D76 06A0 BL @>1FA8 Pop number from VDP stack
0D78 1FA8
0D7A C2CA MOV 10,11

```

FSUB (XML (>07):

```

0D7C 0520 NEG @>834A Make subtraction from addition

```



```

0D7E 834A

FADD (XML >06):
0D80 C28B MOV 11,10
0D82 1003 JMP >0D8A

SADD (XML >0B):
0D84 C28B MOV 11,10
0D86 06A0 BL @>1FA8 Pop number from VDP stack
0D88 1FA8
0D8A C1E0 MOV @>835C,7 ARG in R7
0D8C 835C
0D8E 130A JEQ >0DA4 0?
0D90 C220 MOV @>834A,8 FAC in R8
0D92 834A
0D94 1609 JNE >0DA8 <>0?
0D96 0201 LI 1,>FFF8
0D98 FFF8
0D9A C861 MOV @>8364(1),@>8352(1) ARG in FAC
0D9C 8364
0D9E 8352
0DA0 05C1 INCT 1 All 8 bytes?
0DA2 11FB JLT >0D9A
0DA4 0460 B @>0FA6 Set status (EQU if 0)
0DA6 0FA6

0DA8 29C8 XOR 8,7 Sign
0DAA 0760 ABS @>834A Positive
0DAC 834A
0DAE 0760 ABS @>835C Positive
0DB0 835C
0DB2 0203 LI 3,>FFF8 8 bytes
0DB4 FFF8
0DB6 88E3 C @>8352(3),@>8364(3) Compare
0DB8 8352
0DBA 8364
0DBC 150E JGT >0DDA All o.k.
0DBE 1103 JLT >0DC6 Smaller, exchange
0DC0 05C3 INCT 3
0DC2 16F9 JNE >0DB6 Number end?
0DC4 100A JMP >0DDA
0DC6 C023 MOV @>8364(3),0 Bigger number in FAC
0DC8 8364
0DCA C8E3 MOV @>8352(3),@>8364(3)
0DCC 8352
0DCE 8364
0DD0 C8C0 MOV 0,@>8352(3)
0DD2 8352
0DD4 05C3 INCT 3
0DD6 16F7 JNE >0DC6 Number end?
0DD8 2A07 XOR 7,8 Sign in R8
0DDA 04C5 CLR 5
0DDC 04E0 CLR @>8352 Clear
0DDE 8352
0DE0 04E0 CLR @>8364 The same
0DE2 8364
0DE4 D808 MOVB 8,@>8375 Save sign (in ASCII key)
0DE6 8375
0DE8 04C6 CLR 6
0DEA D820 MOVB @>834A,@>83ED Exponent in R6 Lbyte
0DEC 834A
0DEE 83ED
0DF0 C806 MOV 6,@>8376 Save R6
0DF2 8376
0DF4 D805 MOVB 5,@>834A
0DF6 834A

```

0DF8	7820	SB	@>835C,@>83ED	Difference exponent
0DFA	835C			
0DFC	83ED			
0DFE	0286	CI	6,>0007	Digit number
0E00	0007			
0E02	1540	JGT	>0E84	Bigger, then end
0E04	C006	MOV	6,0	
0E06	0208	LI	8,>0100	
0E08	0100			
0E0A	0209	LI	9,>6400	100 decimal
0E0C	6400			
0E0E	0205	LI	5,>8353	FAC +9
0E10	8353			
0E12	0206	LI	6,>8365	ARG +9
0E14	8365			
0E16	6180	S	0,6	Digit difference
0E18	C100	MOV	0,4	
0E1A	0224	AI	4,>FFF7	Difference loop counter
0E1C	FFF7			
0E1E	C047	MOV	7,1	Negative?
0E20	1120	JLT	>0E62	
0E22	B556	AB	*6,*5	Add
0E24	9255	CB	*5,9	Overflow?
0E26	1A03	JL	>0E2E	
0E28	7549	SB	9,*5	-100
0E2A	B948	AB	8,@>FFFF(5)	+1 on digit higher
0E2C	FFFF			
0E2E	0605	DEC	5	
0E30	0606	DEC	6	
0E32	0584	INC	4	Loop till end
0E34	11F6	JLT	>0E22	
0E36	1002	JMP	>0E3C	
0E38	0605	DEC	5	
0E3A	B548	AB	8,*5	
0E3C	7549	SB	9,*5	Overflow 1st digit
0E3E	15FC	JGT	>0E38	
0E40	13FB	JEQ	>0E38	
0E42	B549	AB	9,*5	Repair old value
0E44	D060	MOVB	@>834A,1	
0E46	834A			
0E48	130B	JEQ	>0E60	
0E4A	05A0	INC	@>8376	Increase exponent
0E4C	8376			
0E4E	0201	LI	1,>8352	All 1 byte up
0E50	8352			
0E52	0202	LI	2,>0009	
0E54	0009			
0E56	D851	MOVB	*1,@>0001(1)	
0E58	0001			
0E5A	0601	DEC	1	
0E5C	0602	DEC	2	
0E5E	16FB	JNE	>0E56	Loop 8 bytes
0E60	107A	JMP	>0F56	
0E62	7556	SB	*6,*5	Minus
0E64	1504	JGT	>0E6E	Overflow?
0E66	1303	JEQ	>0E6E	
0E68	B549	AB	9,*5	Execute overflow
0E6A	7948	SB	8,@>FFFF(5)	
0E6C	FFFF			
0E6E	0605	DEC	5	
0E70	0606	DEC	6	
0E72	0584	INC	4	
0E74	11F6	JLT	>0E62	Loop
0E76	1003	JMP	>0E7E	
0E78	B549	AB	9,*5	
0E7A	0605	DEC	5	

```

0E7C 7548 SB 8,*5
0E7E D115 MOVB *5,4 Check overflow last byte
0E80 11FB JLT >0E78
0E82 104C JMP >0F1C Check subtraction <>0
0E84 0460 B @>0F86 Set exponent and end
0E86 0F86

FMUL (XML >08):
0E88 C28B MOV 11,10
0E8A 1003 JMP >0E92

SMUL (XML >0D):
0E8C C28B MOV 11,10
0E8E 06A0 BL @>1FA8 Fetch ARG from VDP stack
0E90 1FA8
0E92 0203 LI 3,>834A FAC
0E94 834A
0E96 0205 LI 5,>835C ARG
0E98 835C
0E9A C213 MOV *3,8 FAC 0?
0E9C 1346 JEQ >0F2A Set 0
0E9E 2A15 XOR *5,8 Sign in R8
0EA0 0755 ABS *5 ARG to small
0EA2 1343 JEQ >0F2A Set 0
0EA4 0753 ABS *3
0EA6 04C9 CLR 9
0EA8 D253 MOVB *3,9
0EAA B255 AB *5,9 New exponent
0EAC 06C9 SWPB 9
0EAE 0229 AI 9,>FFC1 Correction
0EB0 FFC1
0EB2 C809 MOV 9,@>8376 Save
0EB4 8376
0EB6 D808 MOVB 8,@>8375 Save sign
0EB8 8375
0EBA 0205 LI 5,>8352 Clear >8352->835A
0EBC 8352
0EBE 04F5 CLR *5+
0EC0 0285 CI 5,>835A
0EC2 835A
0EC4 16FC JNE >0EBE
0EC6 0205 LI 5,>8352
0EC8 8352
0ECA 0605 DEC 5 Fetch FAC
0ECC D015 MOVB *5,0 0?
0ECE 13FD JEQ >0ECA Go on
0ED0 0207 LI 7,>0008 Fetch ARG
0ED2 0008
0ED4 0607 DEC 7
0ED6 D027 MOVB @>835C(7),0 0?
0ED8 835C
0EDA 13FC JEQ >0ED4 Go on
0EDC 04C0 CLR 0
0EDE 3880 MPY 0,2 Trick clear R2 and R3
0EE0 C185 MOV 5,6 R5 actual value FAC absolute
0EE2 0208 LI 8,>83E1 LByte R0
0EE4 83E1
0EE6 0209 LI 9,>0064 Decimal 100
0EE8 0064
0EEA C107 MOV 7,4 R7 actual value ARG relative
0EEC A187 A 7,6
0EEE D815 MOVB *5,@>83E7 Lbyte R3
0EF0 83E7
0EF2 D543 MOVB 3,*5 Toward 0
0EF4 D624 MOVB @>835C(4),*8 Lbyte R0
0EF6 835C

```

```

0EF8 3803 MPY 3,0      Multiplying of values
0EFA D816 MOVB *6,@>83E5 Lbyte R2
0EFC 83E5
0EFE A042 A 2,1      Add digit
0F00 3C09 DIV 9,0     Divided by 100
0F02 D5A0 MOVB @>83E3,*6 Write Lbyte R1 to digit
0F04 83E3
0F06 0606 DEC 6      Integer of division to new digit
0F08 B598 AB *8,*6
0F0A 0604 DEC 4      Loop for FAC
0F0C 15F3 JGT >0EF4
0F0E 0606 DEC 6
0F10 0605 DEC 5
0F12 0285 CI 5,>834A Loop for ARG
0F14 834A
0F16 15E9 JGT >0EEA
0F18 04E0 CLR @>8354 No error
0F1A 8354
0F1C 0201 LI 1,>FFF7
0F1E FFF7
0F20 D0A1 MOVB @>8354(1),2 Result <>0
0F22 8354
0F24 1607 JNE >0F34
0F26 0581 INC 1
0F28 11FB JLT >0F20 Loop
0F2A 04E0 CLR @>834A Clear
0F2C 834A
0F2E 04E0 CLR @>834C
0F30 834C
0F32 1039 JMP >0FA6
0F34 C001 MOV 1,0     First digit
0F36 0220 AI 0,>0009 ARG+1
0F38 0009
0F3A 130D JEQ >0F56 Yes, end with rounding
0F3C 6800 S 0,@>8376 Subtract exponent
0F3E 8376
0F40 0202 LI 2,>834B Shift to FAC
0F42 834B
0F44 DCA1 MOVB @>8354(1),*2+
0F46 8354
0F48 0581 INC 1
0F4A 11FC JLT >0F44 Loop
0F4C DC81 MOVB 1,*2+ Additional bytes
0F4E 0600 DEC 0
0F50 15FD JGT >0F4C
0F52 1001 JMP >0F56 End with rounding

```

XML >01 Rounding of floating point numbers

```

0F54 C28B MOV 11,10
0F56 0200 LI 0,>3200 Decimal 50
0F58 3200
0F5A 8020 C @>8352,0 Compare
0F5C 8352
0F5E 1113 JLT >0F86 Smaller, end
0F60 0201 LI 1,>0007
0F62 0007
0F64 0202 LI 2,>0100
0F66 0100
0F68 0200 LI 0,>6400 Decimal 100
0F6A 6400
0F6C B842 AB 2,@>834A(1) +1
0F6E 834A
0F70 9021 CB @>834A(1),0
0F72 834A
0F74 1A08 JL >0F86 Smaller 100, then end
0F76 7840 SB 0,@>834A(1) Minus 100

```

```

0F78 834A
0F7A 0601 DEC 1
0F7C 15F7 JGT >0F6C Next digit
0F7E 05A0 INC @>8376 Increase exponent
0F80 8376
0F82 D802 MOVB 2,@>834B 1 on first digit
0F84 834B
0F86 C0E0 MOV @>8376,3 Fetch exponent
0F88 8376
0F8A 0283 CI 3,>0080 To big?
0F8C 0080
0F8E 141A JHE >0FC4 Overflow
0F90 D820 MOVB @>83E7,@>834A Set exponent
0F92 83E7
0F94 834A
0F96 D0A0 MOVB @>8375,2
0F98 8375
0F9A 0542 INV 2
0F9C 1102 JLT >0FA2 Negative? No, end
0F9E 0520 NEG @>834A Negate number
0FA0 834A
0FA2 1001 JMP >0FA6

```

XML >03 CPU status becomes GPL status in depending of FAC(word)

```

0FA4 C28B MOV 11,10
0FA6 C060 MOV @>834A,1 Fetch FAC
0FA8 834A

```

Store status

```

0FAA 02C2 STST 2
0FAC D802 MOVB 2,@>837C CPU status becomes GPL status
0FAE 837C
0FB0 045A B *10 Return

```

XML >02 Rounding with digit number in >8354

```

0FB2 C28B MOV 11,10
0FB4 D060 MOVB @>8354,1 Digit number in R1
0FB6 8354
0FB8 0981 SRL 1,8 Lbyte
0FBA 10D4 JMP >0F64 Execute

```

0FBC 0209 LI 9,>0200 Overflow +- Infinite with error code 02

```

0FBE 0200
0FC0 1008 JMP >0FD2 Execution

```

XML >04 Overflow

```

0FC2 C28B MOV 11,10
0FC4 D0A0 MOVB @>8376,2 Fetch sign
0FC6 8376
0FC8 11B0 JLT >0F2A Execute toward 0
0FCA 1001 JMP >0FCE Toward infinite

```

XML >05 Set overflow on FAC

```

0FCC C28B MOV 11,10
0FCE 0209 LI 9,>0100 Error code 01
0FD0 0100
0FD2 0200 LI 0,>809D
0FD4 809D
0FD6 D0A0 MOVB @>8375,2 Fetch sign
0FD8 8375
0FDA 1101 JLT >0FDE Positive?
0FDC 0500 NEG 0
0FDE 0202 LI 2,>834A FAC
0FE0 834A
0FE2 CC80 MOV 0,*2+ Load exponent and 1 digit
0FE4 0200 LI 0,>6363 Decimal 99

```

```

0FE6 6363
0FE8 CC80 MOV 0,*2+ Write digits
0FEA CC80 MOV 0,*2+
0FEC C480 MOV 0,*2
0FEE D809 MOVB 9,@>8354 Error code on >8354
0FF0 8354
0FF2 10D9 JMP >0FA6 End set GPL-Status

FDIV (XML >09):
0FF4 C28B MOV 11,10
0FF6 1003 JMP >0FFE

SDIV (XML >0E):
0FF8 C28B MOV 11,10
0FFA 06A0 BL @>1FA8 Fetch number from VDP stack
0FFC 1FA8
0FFE 0203 LI 3,>834A FAC
1000 834A
1002 C213 MOV *3,8
1004 0200 LI 0,>835C ARG
1006 835C
1008 2A10 XOR *0,8
100A D808 MOVB 8,@>8375 Save sign of division
100C 8375
100E 0753 ABS *3 Check on >0000 FAC
1010 13D5 JEQ >0FBC Error with code >02
1012 0750 ABS *0 Check on >0000 ARG
1014 138A JEQ >0F2A End with 0000
1016 D250 MOVB *0,9 Exponent
1018 7253 SB *3,9 New exponent
101A 0889 SRA 9,8
101C 0229 AI 9,>0040 Correction
101E 0040
1020 C809 MOV 9,@>8376 Save on >8376
1022 8376
1024 0204 LI 4,>0004 Save FAC on >8354+
1026 0004
1028 0205 LI 5,>8364 Clear at >8364 through >836B
102A 8364
102C C8F3 MOV *3+,@>0008(3) Execute
102E 0008
1030 04F5 CLR *5+
1032 0604 DEC 4
1034 15FB JGT >102C Loop
1036 D804 MOVB 4,@>835C
1038 835C
103A 0205 LI 5,>83E1 R0 Lbyte
103C 83E1
103E 0206 LI 6,>83E3 R1 Lbyte
1040 83E3
1042 0207 LI 7,>0064
1044 0064
1046 04C2 CLR 2
1048 D820 MOVB @>8355,@>83E5
104A 8355
104C 83E5
104E 0282 CI 2,>0031 Decimal 49
1050 0031
1052 151E JGT >1090
1054 0582 INC 2
1056 04C3 CLR 3
1058 C107 MOV 7,4
105A 3CC2 DIV 2,3 Divided by 100
105C 0209 LI 9,>835C
105E 835C
1060 0204 LI 4,>0008

```

```

1062 0008
1064 0604 DEC 4
1066 0609 DEC 9
1068 D019 MOVB *9,0
106A 13FC JEQ >1064 0?
106C 04C0 CLR 0
106E C080 MOV 0,2
1070 D559 MOVB *9,*5
1072 3803 MPY 3,0
1074 A042 A 2,1
1076 3C07 DIV 7,0
1078 D656 MOVB *6,*9
107A 0609 DEC 9
107C 0604 DEC 4
107E 15F7 JGT >106E
1080 0289 CI 9,>8354
1082 8354
1084 1603 JNE >108C
1086 0209 LI 9,>8364
1088 8364
108A 10EA JMP >1060
108C D815 MOVB *5,@>835C
108E 835C
1090 0206 LI 6,>0008
1092 0008
1094 0606 DEC 6 R6=last digit
1096 D026 MOVB @>8354(6),0
1098 8354
109A 13FC JEQ >1094
109C 04C7 CLR 7
109E D820 MOVB @>8355,@>83EF 1st digit R7 Lbyte
10A0 8355
10A2 83EF
10A4 C207 MOV 7,8
10A6 3A20 MPY @>1044,8 *100
10A8 1044
10AA D820 MOVB @>8356,@>83F1 2nd digit R8 Lbyte
10AC 8356
10AE 83F1
10B0 A248 A 8,9
10B2 0205 LI 5,>FFF7 Loop counter
10B4 FFF7
10B6 020B LI 11,>835C
10B8 835C
10BA 04C2 CLR 2
10BC D81B MOVB *11,@>83E5
10BE 83E5
10C0 38A0 MPY @>1044,2 *100
10C2 1044
10C4 04C0 CLR 0
10C6 D82B MOVB @>0001(11),@>83E1
10C8 0001
10CA 83E1
10CC A0C0 A 0,3
10CE 3C87 DIV 7,2 Divide
10D0 38E0 MPY @>1044,3 Remainder *100
10D2 1044
10D4 D82B MOVB @>0002(11),@>83E1
10D6 0002
10D8 83E1
10DA A100 A 0,4 +Remainder
10DC C002 MOV 2,0
10DE 3808 MPY 8,0
10E0 8802 C 2,@>1044
10E2 1044
10E4 1302 JEQ >10EA Prepare over 100

```

```

10E6 6044 S 4,1
10E8 1003 JMP >10F0
10EA 6044 S 4,1
10EC 0602 DEC 2
10EE 6049 S 9,1
10F0 15FD JGT >10EC
10F2 C082 MOV 2,2
10F4 1329 JEQ >1148
10F6 04C3 CLR 3
10F8 C106 MOV 6,4
10FA A2C6 A 6,11 Next section
10FC C0C0 MOV 0,3
10FE D824 MOVB @>8354(4),@>83E1
1100 8354
1102 83E1
1104 3802 MPY 2,0
1106 A043 A 3,1
1108 3C20 DIV @>1044,0
110A 1044
110C 76E0 SB @>83E3,*11
110E 83E3
1110 1504 JGT >111A
1112 1303 JEQ >111A
1114 B6E0 AB @>1045,*11
1116 1045
1118 0580 INC 0
111A 060B DEC 11
111C 0604 DEC 4
111E 15EE JGT >10FC
1120 76E0 SB @>83E1,*11
1122 83E1
1124 1511 JGT >1148
1126 1310 JEQ >1148
1128 0602 DEC 2
112A C106 MOV 6,4
112C A2C6 A 6,11
112E B6E4 AB @>8354(4),*11 Add ARG
1130 8354
1132 981B CB *11,@>1045 More than 100?
1134 1045
1136 1A05 JL >1142 O.k.
1138 76E0 SB @>1045,*11 Minus 100
113A 1045
113C BAE0 AB @>0E59,@>FFFF(11) +1 one digit higher
113E 0E59
1140 FFFF
1142 060B DEC 11
1144 0604 DEC 4
1146 15F3 JGT >112E
1148 D960 MOVB @>83E5,@>8354(5)
114A 83E5
114C 8354
114E 058B INC 11
1150 0585 INC 5
1152 11B3 JLT >10BA
1154 0460 B @>0F18 End with rounding and shifting in FAC
1156 0F18

1158 3203 DATA >3203

115A 04C4 CLR 4 Convert ASCII in integer
115C 04C0 CLR 0
115E C24B MOV 11,9
1160 1008 JMP >1172
1162 3920 MPY @>117A,4 *10

```



```

1164 117A
1166 C104 MOV 4,4 Overflow?
1168 160D JNE >1184
116A 0580 INC 0
116C A148 A 8,5
116E C105 MOV 5,4
1170 1109 JLT >1184
1172 0693 BL *3 Read character
1174 0228 AI 8,>FFD0 ASCII correction
1176 FFD0
1178 0288 CI 8,>000A Smaller 10?
117A 000A
117C 1AF2 JL >1162 No , go on
117E C000 MOV 0,0
1180 1306 JEQ >118E Overflow
1182 0459 B *9

1184 0209 LI 9,>1190 Trick return is changed
1186 1190
1188 10F4 JMP >1172 Stop in spite

118A 0460 B @>0F2A Set FAC 0 and return
118C 0F2A

118E 045A B *10 Return

1190 0606 DEC 6 New end address
1192 C806 MOV 6,@>8356 on >8356
1194 8356
1196 808C C 12,2 No string for changing
1198 13F8 JEQ >118A Error toward 0
119A C801 MOV 1,@>8376 Set +-
119C 8376
119E 0460 B @>0FC4 Overflow toward +- infinite and end
11A0 0FC4

```

XML >11 (CSN mit Flag auf >8389 0=VDP, <>0=Grom):

```

11A2 D0E0 MOV @>8389,3 Check flag
11A4 8389
11A6 1303 JEQ >11AE
11A8 0203 LI 3,>1FDA From GROM
11AA 1FDA
11AC 1002 JMP >11B2

```

CSN (XML >10):

```

11AE 0203 LI 3,>1FC8 Fetch from VDP
11B0 1FC8
11B2 C28B MOV 11,10
11B4 C1A0 MOV @>8356,6 Fetch address
11B6 8356
11B8 0693 BL *3 Read 1st byte
11BA 04C7 CLR 7
11BC C086 MOV 6,2
11BE 0288 CI 8,>002B ASCII +
11C0 002B
11C2 1304 JEQ >11CC
11C4 0288 CI 8,>002D ASCII -
11C6 002D
11C8 1603 JNE >11D0 No sign
11CA 0707 SETO 7 Flag for minus
11CC 0582 INC 2 Fix length
11CE 0693 BL *3 Next sign
11D0 0288 CI 8,>0030 0?
11D2 0030
11D4 13FC JEQ >11CE Then next sign
11D6 D807 MOV 7,@>8375 Save sign

```

11D8	8375			
11DA	C306	MOV	6,12	Address start string in R12
11DC	060C	DEC	12	Right address
11DE	0707	SETO	7	
11E0	1002	JMP	>11E6	
11E2	0587	INC	7	
11E4	0693	BL	*3	Fetch character
11E6	0288	CI	8,>0030	Compare, if character 0 through 9
11E8	0030			
11EA	1A03	JL	>11F2	
11EC	0288	CI	8,>0039	
11EE	0039			
11F0	12F8	JLE	>11E2	If yes, next character
11F2	0288	CI	8,>002E	Point?
11F4	002E			
11F6	1614	JNE	>1220	
11F8	0582	INC	2	Compute digit left of the point
11FA	C1C7	MOV	7,7	
11FC	1102	JLT	>1202	
11FE	1007	JMP	>120E	
1200	0607	DEC	7	
1202	0693	BL	*3	Next character
1204	0288	CI	8,>0030	0?
1206	0030			
1208	13FB	JEQ	>1200	Go on
120A	0606	DEC	6	
120C	C306	MOV	6,12	
120E	0693	BL	*3	Fetch character
1210	0288	CI	8,>0030	0?
1212	0030			
1214	1A03	JL	>121C	
1216	0288	CI	8,>0039	
1218	0039			
121A	12F9	JLE	>120E	
121C	8086	C	6,2	
121E	13B5	JEQ	>118A	Set 0 and error
1220	C086	MOV	6,2	End of number
1222	04C4	CLR	4	
1224	0602	DEC	2	Correction
1226	04C1	CLR	1	
1228	0288	CI	8,>0045	E?
122A	0045			
122C	160F	JNE	>124C	
122E	0693	BL	*3	Sign exponent
1230	0288	CI	8,>002B	+
1232	002B			
1234	1306	JEQ	>1242	
1236	0288	CI	8,>002D	-
1238	002D			
123A	1602	JNE	>1240	
123C	0601	DEC	1	
123E	1001	JMP	>1242	
1240	0606	DEC	6	
1242	06A0	BL	@>115A	Fetch exponent in integer
1244	115A			
1246	D041	MOVB	1,1	Negative number
1248	1301	JEQ	>124C	
124A	0504	NEG	4	
124C	0606	DEC	6	
124E	C806	MOV	6,@>8356	End address
1250	8356			
1252	808C	C	12,2	
1254	139A	JEQ	>118A	Set error
1256	0224	AI	4,>0080	Correct exponent
1258	0080			
125A	04C1	CLR	1	

```

125C A107 A 7,4
125E C1C4 MOV 4,7
1260 0814 SRA 4,1 Exponent :2, Basis 100!!
1262 C804 MOV 4,@>8376
1264 8376
1266 0B17 SRC 7,1
1268 0205 LI 5,>0008 8 digits of the number
126A 0008
126C 0200 LI 0,>834B Begin number
126E 834B
1270 C18C MOV 12,6 Address, begin of the number in R6
1272 8086 C 6,2
1274 130F JEQ >1294
1276 0693 BL *3
1278 0288 CI 8,>002E Point?
127A 002E
127C 13FA JEQ >1272 Next character
127E 0228 AI 8,>FFD0 ASCII correction
1280 FFD0
1282 0547 INV 7 First, second digit
1284 1105 JLT >1290
1286 3A20 MPY @>117A,8 *10
1288 117A
128A D060 MOVB @>83F3,1 R9 Lbyte in R1
128C 83F3
128E 10F1 JMP >1272 Next character
1290 B060 AB @>83F1,1 R8 Lbyte add to R1
1292 83F1
1294 DC01 MOVB 1,*0+ R1 on FAC
1296 04C1 CLR 1
1298 0605 DEC 5 All digits
129A 16EB JNE >1272
129C 0460 B @>0F56 Rounding and end
129E 0F56

```

XTAB Table XMLLNK 2nd Nybble (attention limit >1B)

```

12A0 11AE CSN (>10)
12A2 11A2 CSN with flag on >8389 byte (0=VDP Ram, 0<>Grom) (>11)
12A4 12B8 CFI (>12)
12A6 1648 Name from VDP OR GROM (to 00) then search in variable list(>13)
12A8 164E Build stack entry from variable list (>834A Pointer to entry)
12AA 1642 Assign value to a variable(stack entry) (>15)
12AC 15D6 Search var name(Name on FAC, >8359 Length, GPL return ) (>16)
12AE 163C VPUSHG (>17)
12B0 1F2E VPOP (>18)
12B2 0AC0 GPL-DSRLNK (>19) Name on FAC, >8359 Length, GPL return
12B4 0B24 GSRLNK (1A) GPL return
12B6 1868 Read byte from >8342, flag >8389 (0=VDP,1=GROM), address >832C (>1B)
)

```

CFI (XML >12)

```

12B8 C120 MOV @>834A,4
12BA 834A
12BC 1342 JEQ >1342 0, End
12BE 04C0 CLR 0
12C0 0202 LI 2,>834B
12C2 834B
12C4 04C3 CLR 3
12C6 0760 ABS @>834A
12C8 834A
12CA 04C5 CLR 5
12CC D160 MOVB @>834A,5 Exponent in R5
12CE 834A
12D0 0285 CI 5,>3F00 Too small
12D2 3F00
12D4 1134 JLT >133E Set 0 end
12D6 1318 JEQ >1308

```

```

12D8 0285 CI 5,>4100 100
12DA 4100
12DC 1112 JLT >1302
12DE 1308 JEQ >12F0
12E0 0285 CI 5,>4200 10000
12E2 4200
12E4 1B25 JH >1330 Overflow, error
12E6 D832 MOVB *2+,@>83E1 in Lbyte R0
12E8 83E1
12EA 3820 MPY @>1320,0 *100
12EC 1320
12EE C001 MOV 1,0
12F0 D832 MOVB *2+,@>83E7 Lbyte R3
12F2 83E7
12F4 A003 A 3,0 +
12F6 3820 MPY @>1320,0 *100
12F8 1320
12FA C000 MOV 0,0 Overflow?
12FC 1619 JNE >1330
12FE C001 MOV 1,0
1300 1117 JLT >1330
1302 D832 MOVB *2+,@>83E7
1304 83E7
1306 A003 A 3,0
1308 9832 CB *2+,@>1158
130A 1158
130C 110B JLT >1324
130E 1509 JGT >1322 Round up
1310 C104 MOV 4,4 Negative?
1312 1507 JGT >1322
1314 D0F2 MOVB *2+,3 Next byte
1316 1605 JNE >1322
1318 0282 CI 2,>8352 End ?
131A 8352
131C 1AFB JL >1314 Loop
131E 1002 JMP >1324 Go on
1320 0064 DATA 100
1322 0580 INC 0 Round up
1324 0280 CI 0,>8000
1326 8000
1328 1A07 JL >1338
132A 1B02 JH >1330 Overflow
132C C104 MOV 4,4
132E 1106 JLT >133C
1330 D820 MOVB @>1159,@>8354 Set error
1332 1159
1334 8354
1336 045B B *11 Return

1338 0544 INV 4 Flag negative
133A 1101 JLT >133E
133C 0500 NEG 0 Negate
133E C800 MOV 0,@>834A Integer on FAC
1340 834A
1342 045B B *11 End

1344 0010 DATA

Cassette write (GPL I/O):
1346 04C0 CLR 0
1348 0202 LI 2,>0300
134A 0300
134C 0208 LI 8,>1E19
134E 1E19
1350 0203 LI 3,>0023
1352 0023

```

```

1354 06A0 BL @>13BA Set CRU and pointer
1356 13BA
1358 0200 LI 0,>13E2 Print routine
135A 13E2
135C 0300 LIMI >0001 Enable interrupt
135E 0001
1360 04C4 CLR 4 Print signal constant >300 * >00
1362 0690 BL *0
1364 0602 DEC 2
1366 16FC JNE >1360 Loop
1368 0704 SETO 4
136A 0690 BL *0 Print signal >FF
136C C105 MOV 5,4
136E 06C4 SWPB 4
1370 0690 BL *0 Length of the transfer 1st byte
1372 C105 MOV 5,4
1374 06C4 SWPB 4
1376 0690 BL *0 Print 2nd byte
1378 04C9 CLR 9
137A 0202 LI 2,>0008
137C 0008
137E 04C4 CLR 4
1380 0690 BL *0 Print 8 times >00
1382 0602 DEC 2
1384 16FC JNE >137E
1386 0704 SETO 4
1388 0690 BL *0 Print 1 time >FF
138A D7E0 MOVB @>83F5,*15 Write buffer address to VDP
138C 83F5
138E 0202 LI 2,>0040 Number of data blocks
1390 0040
1392 D7CA MOVB 10,*15
1394 04C7 CLR 7
1396 04C4 CLR 4
1398 D12F MOVB @>FBFE(15),4 Fetch byte from VDP
139A FBFE
139C A1C4 A 4,7 Build check sum
139E 0690 BL *0 Print byte
13A0 0602 DEC 2 64 bytes?
13A2 16F9 JNE >1396 No, next byte
13A4 C107 MOV 7,4 Transfer check sum
13A6 0690 BL *0
13A8 0549 INV 9 Loop flag
13AA 16E7 JNE >137A The whole twice
13AC 022A AI 10,>0040 Increase buffer address
13AE 0040
13B0 0605 DEC 5 All data blocks?
13B2 16E3 JNE >137A No, go on
13B4 10FF JMP >13B4 Wait for interrupt
13B6 0460 B @>155E CRU reset and end
13B8 155E

```

Pointer for cassette transfer

```

13BA C171 MOV *1+,5 Fetch number of bytes
13BC 0225 AI 5,>003F Integer >40
13BE 003F
13C0 0965 SRL 5,6
13C2 E011 SOC *1,0 Address data buffer
13C4 C280 MOV 0,10 in R10
13C6 D7E0 MOVB @>83E1,*15 Write VDP address
13C8 83E1
13CA 04C1 CLR 1
13CC 04CC CLR 12
13CE D7C0 MOVB 0,*15
13D0 E3A0 SOC @>0032,14 >0020 Set interrupt flag
13D2 0032

```

```

13D4 1E02 SBZ >0002      Interrupt enable
13D6 1E0C SBZ >000C
13D8 33C3 LDCR 3,15      Load CRU
13DA 1E00 SBZ >0000
13DC 1E01 SBZ >0001
13DE 1D03 SBO >0003
13E0 045B B *11         Return

```

Output of a byte to cassette recorder:

```

13E2 0206 LI 6,>0008     Loop counter 8 bits
13E4 0008
13E6 0544 INV 4          Invert byte
13E8 10FF JMP >13E8      Wait for interrupt
13EA 0488 X 8           SBZ >0019 Mag tape out
13EC 2A20 XOR @>135C,8  Command to SB0
13EE 135C
13F0 10FF JMP >13F0      Wait for interrupt
13F2 C104 MOV 4,4         Set bit
13F4 1103 JLT >13FC      Yes, then jump to next byte
13F6 0488 X 8           SBO >0019 Mag tape out
13F8 2A20 XOR @>135C,8  Command to SBZ
13FA 135C
13FC 0A14 SLA 4,1        Next bit
13FE 0606 DEC 6          All 8 bits?
1400 16F3 JNE >13E8      No, go on
1402 045B B *11         Return

```

Interrupt cassette:

```

1404 1E00 SBZ >0000      Control 9901 set
1406 1D03 SBO >0003      Timer interrupt reset
1408 C041 MOV 1,1         R1 Negative?
140A 1107 JLT >141A
140C 02E0 LWPI >83C0     INTWS
140E 83C0
1410 881E C *14,@>13F0 Compare *R14 with>10FF (JMP -2)
1412 13F0
1414 1602 JNE >141A
1416 05CE INCT 14        R14+2 Trick, jump from infinite loop
1418 0380 RTWP           End
141A 02E0 LWPI >83C0
141C 83C0
141E C3A0 MOV @>83EC,14 R6 GPLWS becomes new R14
1420 83EC
1422 10FA JMP >1418

1424 2100 DATA >2100

```

Cassette verify (GPL I/O):

```

1426 E3A0 SOC @>1344,14 Set flag verify
1428 1344
142A 04C0 CLR 0          Read VDP
142C 1004 JMP >1436

```

Cassette read (GPL I/O):

```

142E 43A0 SZC @>1344,14 Flag read
1430 1344
1432 0200 LI 0,>4000     Write VDP
1434 4000
1436 0203 LI 3,>002B     CRU
1438 002B
143A 06A0 BL @>13BA      Fetch pointer, write VDP address
143C 13BA
143E C1CA MOV 10,7       Data buffer in R7
1440 04C0 CLR 0
1442 D820 MOV @>1443,@>837C Set condition bit GPL status
1444 1443

```

1446	837C			
1448	0208	LI	8,>7530	Waiting loop reception
144A	7530			
144C	0300	LIMI	>0001	Enable interrupt
144E	0001			
1450	0206	LI	6,>1458	Over interrupt new PC
1452	1458			
1454	0203	LI	3,>002B	CRU
1456	002B			
1458	0241	ANDI	1,>00FF	
145A	00FF			
145C	0608	DEC	8	Count down loop
145E	137F	JEQ	>155E	End with error
1460	0202	LI	2,>0030	48
1462	0030			
1464	C000	MOV	0,0	
1466	1601	JNE	>146A	
1468	A082	A	2,2	R2 double
146A	06A0	BL	@>1572	
146C	1572			
146E	1001	JMP	>1472	Receive character, go on
1470	10F3	JMP	>1458	Receive no character, once again
1472	0602	DEC	2	At least R2 character
1474	16FA	JNE	>146A	No, next character
1476	0209	LI	9,>7FFF	
1478	7FFF			
147A	0208	LI	8,>0008	
147C	0008			
147E	33C9	LDCR	9,15	Set CRU
1480	1E00	SBZ	>0000	
1482	1D03	SBO	>0003	
1484	06A0	BL	@>15BA	Receive character
1486	15BA			
1488	1001	JMP	>148C	Receive change (bit)
148A	10FC	JMP	>1484	No change
148C	0608	DEC	8	8 Bits?
148E	16FA	JNE	>1484	Once more
1490	1D00	SBO	>0000	
1492	37C3	STCR	3,15	Fetch CRU
1494	6243	S	3,9	Change
1496	C0C9	MOV	9,3	
1498	0A29	SLA	9,2	
149A	A0C9	A	9,3	
149C	0963	SRL	3,6	
149E	0263	ORI	3,>0001	Set last bit
14A0	0001			
14A2	020A	LI	10,>14B0	Trick return
14A4	14B0			
14A6	0283	CI	3,>001F	
14A8	001F			
14AA	11D4	JLT	>1454	
14AC	0460	B	@>1580	Read bit
14AE	1580			
14B0	06A0	BL	@>1572	Receive 1 bit
14B2	1572			
14B4	10FD	JMP	>14B0	No change
14B6	0202	LI	2,>0007	Go on 7 bits
14B8	0007			
14BA	06A0	BL	@>1572	
14BC	1572			
14BE	10CC	JMP	>1458	Once more
14C0	0602	DEC	2	All 7?
14C2	16FB	JNE	>14BA	No, go on
14C4	0206	LI	6,>14F8	Trick return
14C6	14F8			
14C8	C000	MOV	0,0	Data block

```

14CA 1631 JNE >152E
14CC D820 MOVB @>1424,@>837C Set error
14CE 1424
14D0 837C
14D2 C007 MOV 7,0 Prepare address
14D4 04C7 CLR 7
14D6 06A0 BL @>15A0 Receive number of data blocks
14D8 15A0
14DA 8105 C 5,4 Enough storage
14DC 1A40 JL >155E End with error
14DE C144 MOV 4,5 New number data blocks
14E0 0585 INC 5
14E2 0507 NEG 7
14E4 06A0 BL @>15A0 Fetch 2nd time
14E6 15A0
14E8 163A JNE >155E
14EA 101D JMP >1526 Go on with 1st data block
14EC 0247 ANDI 7,>00FF Clear 1st byte
14EE 00FF
14F0 0507 NEG 7 Negate
14F2 06A0 BL @>15A0 Fetch check sum
14F4 15A0
14F6 1307 JEQ >1506 O.k. (Addition must result in 0,if data o.k)
14F8 C145 MOV 5,5 Already the 2nd time
14FA 1131 JLT >155E End with error
14FC D7E0 MOVB @>83E1,*15 Write VDP address
14FE 83E1
1500 0505 NEG 5 Flag R5
1502 D7C0 MOVB 0,*15
1504 10A1 JMP >1448 Once more from beginning
1506 C145 MOV 5,5 1 time
1508 1108 JLT >151A No, jump
150A 0202 LI 2,>0049 Receive 49 character
150C 0049
150E 0206 LI 6,>1516 New PC over interrupt
1510 1516
1512 06A0 BL @>15A0 Fetch byte
1514 15A0
1516 0602 DEC 2 All ?
1518 16FA JNE >150E
151A 0220 AI 0,>0040 New VDP address
151C 0040
151E D7E0 MOVB @>83E1,*15 Write address
1520 83E1
1522 0745 ABS 5
1524 D7C0 MOVB 0,*15
1526 04C7 CLR 7
1528 0605 DEC 5 All data blocks
152A 168E JNE >1448
152C 1015 JMP >1558 End
152E 0202 LI 2,>0040 >40 Character
1530 0040
1532 04C7 CLR 7
1534 06A0 BL @>15A0 Receive
1536 15A0
1538 06C4 SWPB 4
153A 23A0 COC @>1344,14 >0010
153C 1344
153E 1607 JNE >154E
1540 712F SB @>FBFE(15),4 Verify
1542 FBFE
1544 1306 JEQ >1552 O.k. Jump
1546 0285 CI 5,>0001
1548 0001
154A 1303 JEQ >1552 End of data blocks ?
154C 10D5 JMP >14F8

```



```

154E DBC4 MOVB 4,@>FFFE(15) Byte in VDP
1550 FFFE
1552 0602 DEC 2 End of data blocks ?
1554 16EF JNE >1534
1556 10CA JMP >14EC
1558 D820 MOVB @>1438,@>837C Clear condition bit GPL status
155A 1438
155C 837C
155E 43A0 SZC @>1344,14 Clear interrupt flags
1560 1344
1562 43A0 SZC @>0032,14
1564 0032
1566 1E03 SBZ >0003 CRU reset
1568 1D0C SBO >000C
156A 1D01 SBO >0001
156C 1D02 SBO >0002
156E 0460 B @>0070 To GPL interpreter
1570 0070

```

```

1572 C28B MOV 11,10
1574 10FF JMP >1574 Wait for interrupt
1576 06A0 BL @>15BA
1578 15BA
157A 05CA INCT 10 No change from R1
157C 0261 ORI 1,>FF00
157E FF00
1580 2460 CZC @>145A,1 R1 >FF00
1582 145A
1584 1303 JEQ >158C
1586 1F1B TB >001B Mag tape in, NEQ if R1=>FF
1588 1603 JNE >1590
158A 10FD JMP >1586
158C 1F1B TB >001B Mag tape in, EQU if R1=>00
158E 16FE JNE >158C
1590 33C3 LD CR 3,15 Load new CRU
1592 1E00 SBZ >0000
1594 1D03 SBO >0003
1596 0241 ANDI 1,>00FF R1 >00XX
1598 00FF
159A 2860 XOR @>145A,1 XOR with>00FF
159C 145A
159E 045A B *10 Return

```

Receive byte in R4 and built check sum in R7

```

15A0 0208 LI 8,>0008 8 Bits
15A2 0008
15A4 04C4 CLR 4
15A6 C24B MOV 11,9
15A8 0A14 SLA 4,1
15AA 06A0 BL @>1572 Fetch 1 bit
15AC 1572
15AE 1001 JMP >15B2 See TB entry
15B0 0584 INC 4 Set bit
15B2 0608 DEC 8
15B4 16F9 JNE >15A8 All 8 bits?
15B6 A1C4 A 4,7 Built check sum
15B8 0459 B *9 Return

15BA 1F1B TB >001B Mag tape in if EQU R1=00, then R1=FF u. B*11
15BC 1306 JEQ >15CA if EQU R1=FF, then R1=FF u. B*11+2
15BE 2460 CZC @>145A,1 >FF if NEQ R1=00, then R1=00 u. B*11+2
15C0 145A if NEQ R1=FF, then R1=00 u. B*11
15C2 1306 JEQ >15D0
15C4 2860 XOR @>145A,1
15C6 145A
15C8 045B B *11

```

```

15CA 2460 CZC @>145A,1
15CC 145A
15CE 13FA JEQ >15C4
15D0 05CB INCT 11
15D2 045B B *11

15D4 6500 DATA >6500

XML >16 (Search variable name), leads back to GPL
15D6 06A0 BL @>15E0 Search name
15D8 15E0
15DA 006A DATA >006A Return reset condition bit
15DC 0460 B @>00CE Return set condition bit
15DE 00CE

15E0 C120 MOV @>833E,4 Pointer fetch var list
15E2 833E
15E4 1312 JEQ >160A No list, end reset condition bit
15E6 D0E0 MOVB @>8359,3 Fetch length byte
15E8 8359
15EA 04C7 CLR 7
15EC 0584 INC 4
15EE D7E0 MOVB @>83E9,*15 Write VDP address
15F0 83E9
15F2 1000 JMP >15F4
15F4 D7C4 MOVB 4,*15
15F6 020A LI 10,>8800 VDP read data
15F8 8800
15FA 90DA CB *10,3 Compare length of variable
15FC 1308 JEQ >160E Right, check name
15FE D19A MOVB *10,6 Address next variable
1600 1000 JMP >1602
1602 D81A MOVB *10,@>83ED
1604 83ED
1606 C106 MOV 6,4 New address in R4
1608 16F1 JNE >15EC Go on
160A C2DB MOV *11,11 Fetch return
160C 045B B *11 Return

160E D19A MOVB *10,6 Address next variable
1610 1000 JMP >1612
1612 D81A MOVB *10,@>83ED
1614 83ED
1616 1000 JMP >1618
1618 D15A MOVB *10,5 Address name of variable
161A D803 MOVB 3,@>83EF Length byte in R7 Lbyte
161C 83EF
161E D09A MOVB *10,2
1620 D7C2 MOVB 2,*15 Write address VDP
1622 1000 JMP >1624
1624 D7C5 MOVB 5,*15
1626 0202 LI 2,>834A FAC
1628 834A
162A 9C9A CB *10,*2+ Compare name
162C 16EC JNE >1606 Next variable
162E 0607 DEC 7
1630 15FC JGT >162A Until length end
1632 0604 DEC 4
1634 C804 MOV 4,@>834A Address on FAC shows to value of variables
1636 834A
1638 046B B @>0002(11) Return +2
163A 0002

VPUSHG (XML >17)
163C 0206 LI 6,>1EAA Routine address

```

```

163E 1EAA
1640 1008 JMP >1652

```

XML >15 Coordinate VDP stack entry to variable

```

1642 0206 LI 6,>1788 Routine address
1644 1788
1646 1005 JMP >1652

```

XML >13 Look on FAC and in symbole list for variable name from GROM or from VDP

```

1648 0206 LI 6,>176A Routine address
164A 176A
164C 1002 JMP >1652

```

XML >14 Value stack entry for variable on FAC

```

164E 0206 LI 6,>1670 Routine address
1650 1670
1652 C1CB MOV 11,7 Save return
1654 06A0 BL @>0864 Push GROM address on substack
1656 0864
1658 06A0 BL @>1E7A R9 Address substack, R8=>8342, actual Basic byte
165A 1E7A
165C 05C9 INCT 9 ROM return address on substack
165E C647 MOV 7,*9
1660 0696 BL *6 Execute routine
1662 C1D9 MOV *9,7 Return address from substack
1664 0649 DECT 9
1666 06A0 BL @>1E8C Set substack pointer and >8342
1668 1E8C
166A 06A0 BL @>0842 POP GROM address from substack
166C 0842
166E 0457 B *7 Return

```

XML >14 Main part:

```

1670 05C9 INCT 9 Return on substack
1672 C64B MOV 11,*9
1674 C820 MOV @>834A,@>834E FAC on FAC+4
1676 834A
1678 834E
167A A820 A @>1816,@>834E Add FAC+4 >0006
167C 1816
167E 834E
1680 06A0 BL @>19F6 Fetch one byte from VDP address in R1
1682 19F6
1684 1108 JLT >1696 Byte in R1 negative = string
1686 04E0 CLR @>834C No jump
1688 834C
168A 0288 CI 8,>B700 Basic token (
168C B700
168E 131E JEQ >16CC Jump
1690 C2D9 MOV *9,11 Fetch return from stack
1692 0649 DECT 9
1694 045B B *11 Return

1696 0288 CI 8,>B700 Basic token (
1698 B700
169A 1318 JEQ >16CC Jump
169C C820 MOV @>15D4,@>834C >6500 on >834C (String tag)
169E 15D4
16A0 834C
16A2 C0E0 MOV @>834E,3
16A4 834E
16A6 C803 MOV 3,@>834A Address +6
16A8 834A
16AA 06A0 BL @>1880 Byte in R1 from VDP
16AC 1880

```

```

16AE D820 MOV @>8800,@>83E3 Word pointer value variable?
16B0 8800
16B2 83E3
16B4 C801 MOV 1,@>834E Address string
16B6 834E
16B8 C0C1 MOV 1,3 Space string
16BA 1304 JEQ >16C4 Jump
16BC 0603 DEC 3 Address length
16BE 06A0 BL @>1880 Fetch byte
16C0 1880
16C2 0981 SRL 1,8
16C4 C801 MOV 1,@>8350 Length on >8350
16C6 8350
16C8 10E3 JMP >1690 End
16CA 0007 DATA >0007
Data field:
16CC 0A51 SLA 1,5 1st 5 bits way, number of dimensions
16CE 09D1 SRL 1,13
16D0 C801 MOV 1,@>834C On >834C (Number dimensions)
16D2 834C
16D4 04C2 CLR 2 Length
16D6 C802 MOV 2,@>8350
16D8 8350
16DA 06A0 BL @>1F7E Fetch 1 byte from VDP or GROM
16DC 1F7E
16DE 06A0 BL @>1E9C VPUSHG with PARSE Basic interpreter until )
16E0 1E9C
16E2 B700 DATA >B700 Token )
16E4 9820 CB @>834C,@>15D4 >65
16E6 834C
16E8 15D4
16EA 1425 JHE >1736
16EC 04E0 CLR @>8354 For possible error
16EE 8354
16F0 06A0 BL @>12B8 CFI
16F2 12B8
16F4 D120 MOV @>8354,4 Error?
16F6 8354
16F8 1631 JNE >175C End with setting error
16FA C160 MOV @>834A,5 Pointer for variable table
16FC 834A
16FE 06A0 BL @>1F2E VPOP
1700 1F2E
1702 06A0 BL @>187C Fetch byte from VDP
1704 187C
1706 834E DATA >834E Pointer to value
1708 D820 MOV @>8800,@>83E3 Word in R1
170A 8800
170C 83E3
170E 8045 C 5,1 Right value
1710 1B25 JH >175C
1712 D120 MOV @>8343,4 Option base value
1714 8343
1716 1303 JEQ >171E
1718 0605 DEC 5
171A 1120 JLT >175C
171C 1001 JMP >1720
171E 0581 INC 1
1720 3860 MPY @>8350,1 Right address to pointer on value
1722 8350
1724 A085 A 5,2 + Begin
1726 05E0 INCT @>834E
1728 834E
172A 0620 DEC @>834C All dimensions?
172C 834C
172E 1305 JEQ >173A

```

```

1730 0288 CI 8,>B300 Token ,?
1732 B300
1734 13D0 JEQ >16D6
1736 0460 B @>1A2C End with error
1738 1A2C

173A 0288 CI 8,>B600 Token )
173C B600
173E 16FB JNE >1736
1740 06A0 BL @>1F7E VPOP
1742 1F7E
1744 06A0 BL @>187C Fetch byte from VDP
1746 187C
1748 834A DATA >834A
174A 1104 JLT >1754 String
174C 0A32 SLA 2,3 8 (for numeric data field)
174E A802 A 2,@>834E Correct address value
1750 834E
1752 109E JMP >1690 End
1754 0A12 SLA 2,1 2
1756 A802 A 2,@>834E Correct address
1758 834E
175A 10A0 JMP >169C End with setting string tag on >834C
175C 0200 LI 0,>0503 Error
175E 0503
1760 0460 B @>1A30 End with error
1762 1A30

1764 0200 LI 0,>0603 Error
1766 0603
1768 10FB JMP >1760

```

XML >13 Main part:

```

176A 04E0 CLR @>8359 Length pointer 0
176C 8359
176E 0202 LI 2,>834A FAC
1770 834A
1772 C04B MOV 11,1 Save return
1774 DC88 MOVB 8,*2+ R8 on FAC (at >8342)
1776 05A0 INC @>8359 Increase length byte
1778 8359
177A 06A0 BL @>1F7E Fetch byte from VDP or GROM
177C 1F7E
177E 15FA JGT >1774 Until negative
1780 06A0 BL @>15E0 Search name in variable list
1782 15E0
1784 1736 DATA >1736 Data (Return not found)
1786 0451 B *1 O.k. found and return

```

XML >15 Main part:

```

1788 C28B MOV 11,10
178A 06A0 BL @>1FA8 Fetch 8 bytes from stack on ARG
178C 1FA8
178E 06A0 BL @>187C Read 1 byte from VDP in R1
1790 187C
1792 835C DATA >835C Data VDP addresses pointer
1794 9820 CB @>835E,@>15D4 String tag?
1796 835E
1798 15D4
179A 1307 JEQ >17AA O.k. go on
179C 9820 CB @>834C,@>15D4 String tag?
179E 834C
17A0 15D4
17A2 14E0 JHE >1764 Error
17A4 0202 LI 2,>0008
17A6 0008

```

```

17A8 1051 JMP >184C
17AA 9820 CB @>834C,@>15D4 String tag?
17AC 834C
17AE 15D4
17B0 16D9 JNE >1764 Error
17B2 C060 MOV @>8360,1 Address?
17B4 8360
17B6 130E JEQ >17D4 Set 0
17B8 06A0 BL @>187C 1 Byte in R1
17BA 187C
17BC 835C DATA >83C5 Pointer address VDP
17BE D820 MOVB @>8800,@>83E3 Fetch 2nd byte
17C0 8800
17C2 83E3
17C4 C801 MOV 1,@>8360 Pointer to string
17C6 8360
17C8 8801 C 1,@>834E
17CA 834E
17CC 1317 JEQ >17FC String identical, end
17CE 04C6 CLR 6
17D0 06A0 BL @>18AA Write R6 in VDP (string offset -3)
17D2 18AA
17D4 C120 MOV @>8350,4 Length string
17D6 8350
17D8 130C JEQ >17F2 0 then end
17DA C0E0 MOV @>834A,3
17DC 834A
17DE 0283 CI 3,>001C String expression?
17E0 001C
17E2 160D JNE >17FE
17E4 C120 MOV @>834E,4 Pointer to string
17E6 834E
17E8 C1A0 MOV @>835C,6 Pointer variable list
17EA 835C
17EC C044 MOV 4,1
17EE 06A0 BL @>18AA Write pointer to variable list
17F0 18AA
17F2 C060 MOV @>835C,1 Pointer variable list
17F4 835C
17F6 C184 MOV 4,6
17F8 06A0 BL @>18AE Write R6 in VDP without offset
17FA 18AE
17FC 045A B *10 End

17FE C820 MOV @>8350,@>830C String length
1800 8350
1802 830C
1804 C820 MOV @>16CA,@>8322 Error code 7
1806 16CA
1808 8322
180A 06A0 BL @>1EAA VPUSHG
180C 1EAA
180E C80A MOV 10,@>834A
1810 834A
1812 06A0 BL @>1A4A Fetch space for string (over GROM)
1814 1A4A
1816 0006 DATA >0006
1818 C2A0 MOV @>834A,10
181A 834A
181C 06A0 BL @>1F2E VPOP
181E 1F2E
1820 C0E0 MOV @>834E,3 String address
1822 834E
1824 C160 MOV @>831C,5 Pointer to string space
1826 831C
1828 C105 MOV 5,4

```

```

182A C0A0 MOV @>8350,2 String length
182C 8350
182E 0265 ORI 5,>4000
1830 4000
1832 06A0 BL @>1880 1 Byte from VDP without data pointer
1834 1880
1836 D7E0 MOVB @>83EB,*15 Write VDP address
1838 83EB
183A 1000 JMP >183C
183C D7C5 MOVB 5,*15
183E 0585 INC 5
1840 D801 MOVB 1,@>8C00 Write R1 in VDP
1842 8C00
1844 0583 INC 3 New address
1846 0602 DEC 2 Length-1
1848 15F4 JGT >1832 Go on
184A 10CE JMP >17E8 End

184C C160 MOV @>8360,5 Value pointer
184E 8360
1850 D7E0 MOVB @>83EB,*15 Write VDP address
1852 83EB
1854 0265 ORI 5,>4000 For writing
1856 4000
1858 D7C5 MOVB 5,*15
185A 0204 LI 4,>834A FAC
185C 834A
185E D834 MOVB *4+,@>8C00 From FAC in VDP
1860 8C00
1862 0602 DEC 2
1864 15FC JGT >185E Until end
1866 045A B *10 Return

XML >1B Read 1 byte from VDP or GROM
1868 C30B MOV 11,12
186A 06A0 BL @>0864 Push GROM address on substack
186C 0864
186E 06A0 BL @>1F7E Fetch byte
1870 1F7E
1872 D808 MOVB 8,@>8342 Read byte at >8242 (actual Basic byte)
1874 8342
1876 06A0 BL @>0842 POP GROM address from substack
1878 0842
187A 045C B *12 Return

Read 1 byte from VDP, Entry over data address pointer
187C C0FB MOV *11+,3 Fetch data
187E C0D3 MOV *3,3 Fetch address
1880 D7E0 MOVB @>83E7,*15 Write address
1882 83E7
1884 1000 JMP >1886
1886 D7C3 MOVB 3,*15
1888 1000 JMP >188A
188A D060 MOVB @>8800,1 Read byte
188C 8800
188E 045B B *11

Fetch 8 bytes from VDP over stack entry FAC
1890 D7E0 MOVB @>834F,*15 Write address
1892 834F
1894 0202 LI 2,>0008 Loop counter
1896 0008
1898 D7E0 MOVB @>834E,*15
189A 834E
189C 0203 LI 3,>834A FAC
189E 834A

```

```

18A0 DCE0 MOVB @>8800,*3+  Fetch byte
18A2 8800
18A4 0602 DEC 2
18A6 15FC JGT >18A0      Loop end?
18A8 045B B *11      Return

Write R6 in VDP (R1=Address+3), used for variable table and string pointer
18AA 0221 AI 1,>FFFD R1-3
18AC FFFD
18AE D7E0 MOVB @>83E3,*15 Write address
18B0 83E3
18B2 0261 ORI 1,>4000 For writing
18B4 4000
18B6 D7C1 MOVB 1,*15
18B8 1000 JMP >18BA
18BA D806 MOVB 6,@>8C00 Write 1st byte
18BC 8C00
18BE D820 MOVB @>83ED,@>8C00 2nd byte
18C0 83ED
18C2 8C00
18C4 045B B *11      Return

18C6 19CA DATA >19CA

GPL PARSE
18C8 06A0 BL @>1E7A      Pointer substack in R9, >8342 in R8
18CA 1E7A
18CC D2ED MOVB @>0002(13),11 Actual GROM address in R11
18CE 0002
18D0 D82D MOVB @>0002(13),@>83F7
18D2 0002
18D4 83F7
18D6 022B AI 11,>7FFF Trick R11 1st bit is set=GROM
18D8 7FFF
18DA 05C9 INCT 9
18DC 0289 CI 9,>83BA Substack full
18DE 83BA
18E0 1B1D JH >191C
18E2 C64B MOV 11,*9 R11 on substack
18E4 D1C8 MOVB 8,7
18E6 1102 JLT >18EC Basic token?
18E8 0460 B @>1B94 Fetch variable name and go on
18EA 1B94

18EC 06A0 BL @>1F7E Fetch byte from VDP or GROM
18EE 1F7E
18F0 0977 SRL 7,7 Old Basic byte
18F2 0227 AI 7,>FE92 ->B7*2
18F4 FE92
18F6 0287 CI 7,>004C Bigger >26 (>DD)
18F8 004C
18FA 1B28 JH >194C Error
18FC C1E7 MOV @>1CE2(7),7 Fetch jump
18FE 1CE2
1900 152B JGT >1958 Positive, then jump
1902 0247 ANDI 7,>7FFF Correct GROM address
1904 7FFF
1906 A1E0 A @>8328,7 Add NUD table address (>4E84 at master console)
1908 8328
190A 06A0 BL @>1E8C Set substack pointer and actual Basic token
190C 1E8C
190E DB47 MOVB 7,@>0402(13) Write GROM address
1910 0402
1912 DB60 MOVB @>83EF,@>0402(13)
1914 83EF
1916 0402

```



```

1918 0460 B    @>006A    GPL interpreter with reset condition bit
191A 006A

191C 0460 B    @>1F22    End with error
191E 1F22

GPL CONT
1920 06A0 BL    @>1E7A    Substack pointer in R9, Basic byte in R8
1922 1E7A
1924 C199 MOV   *9,6    Fetch address from substack
1926 1508 JGT   >1938    Not negative, then jump
1928 0246 ANDI  6,>7FFF    GROM address clear 1st bit
192A 7FFF
192C DB46 MOVB  6,@>0402(13) and write
192E 0402
1930 DB60 MOVB  @>83ED,@>0402(13)
1932 83ED
1934 0402
1936 C18D MOV   13,6
1938 9216 CB    *6,8    Compare byte in GROM with old byte
193A 1455 JHE   >19E6
193C 0288 CI    8,>B800    &?
193E B800
1940 130C JEQ   >195A
1942 0978 SRL   8,7
1944 0228 AI    8,>FE84    ->BE*2
1946 FE84
1948 0288 CI    8,>0010    Basic byte >BE->C5
194A 0010
194C 1B6F JH    >1A2C    Jump
194E C1E8 MOV   @>1D2E(8),7 Fetch routine
1950 1D2E
1952 04C8 CLR   8
1954 06A0 BL    @>1F7E    Set substack pointer and actual Basic byte
1956 1F7E
1958 0457 B    *7

195A 0200 LI    0,>0008    GPL routine &
195C 0008
195E 1068 JMP   >1A30    Execute

1960 0649 DECT  9    New substack pointer
1962 0227 AI    7,>8001    Address +1
1964 8001
1966 10D1 JMP   >190A    Write GROM address and to GPL interpreter

GPL EXEC
1968 06A0 BL    @>1E7A    Substack pointer in R9, Basic byte in R8
196A 1E7A
196C 04E0 CLR   @>8322    Error code
196E 8322
1970 C020 MOV   @>8344,0    Basic flag in R0
1972 8344
1974 1317 JEQ   >19A4    Nothing,then jump
1976 0300 LIMI  >0003    Enable interrupt
1978 0003
197A 0300 LIMI  >0000    Disable
197C 0000
197E 04E0 CLR   @>83D6    Clear screen time out counter
1980 83D6
1982 06A0 BL    @>0020    Clear key scan
1984 0020
1986 134F JEQ   >1A26    End
1988 D020 MOVB  @>8388,0    Error flag bit
198A 8388
198C 0A30 SLA   0,3    Check 2nd bit

```

```

198E 1160 JLT >1A50
1990 C820 MOV @>832E,@>832C Line pointer on text in linee pointer
1992 832E
1994 832C
1996 06A0 BL @>1F7E Fetch 1 byte from VDP or GROM
1998 1F7E
199A 1142 JLT >1A20
199C D808 MOV 8,@>832C Fetch line address
199E 832C
19A0 D81A MOV *10,@>832D
19A2 832D
19A4 05C9 INCT 9 Increase stack pointer
19A6 C660 MOV @>18C6,*9 Value>19CA on substack
19A8 18C6
19AA 06A0 BL @>1F7E Fetch 1st byte from line
19AC 1F7E
19AE 1102 JLT >19B4 <0?
19B0 0460 B @>1BEA
19B2 1BEA

19B4 C1C8 MOV 8,7 Basic token in R7
19B6 06A0 BL @>1FA0 Next byte of the line
19B8 1FA0
19BA 0977 SRL 7,7 1st token in R7 Lbyte *2
19BC 0227 AI 7,>FEBC
19BE FEBC
19C0 1535 JGT >1A2C Bigger >A2 ?
19C2 C1E7 MOV @>1CE0(7),7
19C4 1CE0
19C6 119D JLT >1902 Smaller 0, then GROM address
19C8 0457 B *7 Execute routine

19CA 0065 DATA >0065 Dec 101

19CC C020 MOV @>8344,0 Basic run flag and Basic flag byte
19CE 8344
19D0 1331 JEQ >1A34 No, execute direct
19D2 6820 S @>1D4E,@>832E Next line (-4)
19D4 1D4E
19D6 832E
19D8 8820 C @>832E,@>8330 Smaller then lowest end of line list?
19DA 832E
19DC 8330
19DE 14CB JHE >1976 Execute in the limit
19E0 1029 JMP >1A34 End program

19E2 D208 MOV 8,8
19E4 1623 JNE >1A2C
19E6 C1D9 MOV *9,7 Data from substack
19E8 11BB JLT >1960 Negative,then jump
19EA 0649 DECT 9 Decrease stack pointer
19EC 0467 B @>0002(7) and execute routine (value+2)
19EE 0002

GPL RTNB:
19F0 06A0 BL @>1E7A Substack pointer in R9, actual Basic byte in R8
19F2 1E7A
19F4 10F8 JMP >19E6 Go on

19F6 C08B MOV 11,2 Save return
19F8 06A0 BL @>187C 1 Byte from VDP in R1
19FA 187C
19FC 834A DATA >834A Pointer to address
19FE C101 MOV 1,4
1A00 0A21 SLA 1,2 Is 2nd bit set?
1A02 1814 JOC >1A2C Yes, error

```

1A04	C044	MOV	4,1	Again original byte
1A06	0452	B	*2	Return
1A08	0000			
1A0A	0000			
1A0C	0000			
1A0E	0000			
1A10	0000			
1A12	0000			
1A14	0000			
1A16	0000			
1A18	0000			
1A1A	0000			
1A1C	0000			
1A1E	0000			
1A20	D020	MOVB	@>8389,0	Fetch GROM flag
1A22	8389			
1A24	16BB	JNE	>199C	Jump, if in GROM
1A26	0200	LI	0,>0001	GPL routine break
1A28	0001			
1A2A	1002	JMP	>1A30	
1A2C	0200	LI	0,>0003	GPL error routine, error code 0
1A2E	0003			
1A30	C800	MOV	0,@>8322	Set GPL routine on vector
1A32	8322			
1A34	C1E0	MOV	@>8326,7	Fetch return address Basic
1A36	8326			
1A38	0460	B	@>190A	Set stack pointer, write Basic return address
1A3A	190A			Return GPL interpreter
1A3C	0649	DECT	9	Decrease substack pointer
1A3E	10FA	JMP	>1A34	Return Basic
1A40	C820	MOV	@>1D4E,@>8322	GPL routine memory full?
1A42	1D4E			
1A44	8322			
1A46	020B	LI	11,>1922	Trick return >1922
1A48	1922			
1A4A	05C9	INCT	9	Increase stack pointer
1A4C	C64B	MOV	11,*9	Return address on substack
1A4E	10F2	JMP	>1A34	Goto Basic
1A50	C820	MOV	@>0072,@>8322	GPL routine TRACE
1A52	0072			
1A54	8322			
1A56	020B	LI	11,>198E	Return for substack, go on in Basic interpreter
1A58	198E			
1A5A	10F7	JMP	>1A4A	Go on
1A5C	C820	MOV	@>832C,@>8356	Text pointer
1A5E	832C			
1A60	8356			
1A62	06C8	SWPB	8	
1A64	A808	A	8,@>832C	Text pointer+R8
1A66	832C			
1A68	04E0	CLR	@>8354	
1A6A	8354			
1A6C	06A0	BL	@>1E90	Set stack pointer
1A6E	1E90			
1A70	06A0	BL	@>11A2	CSN
1A72	11A2			
1A74	06A0	BL	@>1E7A	Fetch substack and Basic byte in R8
1A76	1E7A			
1A78	8820	C	@>8356,@>832C	Text pointer in Basic line
1A7A	8356			

1A7C	832C			
1A7E	16D6	JNE	>1A2C	
1A80	06A0	BL	@>1F7E	Fetch byte
1A82	1F7E			
1A84	D020	MOVB	@>8354,0	Test error
1A86	8354			
1A88	16DB	JNE	>1A40	
1A8A	0460	B	@>1924	CONT
1A8C	1924			
1A8E	04C3	CLR	3	
1A90	1017	JMP	>1AC0	
1A92	06A0	BL	@>18DA	PARSE with different return
1A94	18DA			
1A96	B366	DATA	>B366	
1A98	06A0	BL	@>1E70	String tag?
1A9A	1E70			
1A9C	04E0	CLR	@>8354	
1A9E	8354			
1AA0	06A0	BL	@>12B8	CFI
1AA2	12B8			
1AA4	D020	MOVB	@>8354,0	Error?
1AA6	8354			
1AA8	1603	JNE	>1AB0	
1AAA	C0E0	MOV	@>834A,3	
1AAC	834A			
1AAE	1503	JGT	>1AB6	
1AB0	0200	LI	0,>0203	Error code
1AB2	0203			
1AB4	10BD	JMP	>1A30	
1AB6	0288	CI	8,>8500	Token GO
1AB8	8500			
1ABA	1608	JNE	>1ACC	
1ABC	06A0	BL	@>1F7E	
1ABE	1F7E			
1AC0	0288	CI	8,>B100	Token ELSE
1AC2	B100			
1AC4	1353	JEQ	>1B6C	
1AC6	0288	CI	8,>A100	Token SUB
1AC8	A100			
1ACA	1005	JMP	>1AD6	
1ACC	0288	CI	8,>8600	Token GOTO
1ACE	8600			
1AD0	134D	JEQ	>1B6C	
1AD2	0288	CI	8,>8700	Token GOSUB
1AD4	8700			
1AD6	16AA	JNE	>1A2C	
1AD8	06A0	BL	@>1F7E	Fetch byte
1ADA	1F7E			
1ADC	1002	JMP	>1AE2	Execute
1ADE	10A6	JMP	>1A2C	Error
1AE0	04C3	CLR	3	
1AE2	C820	MOV	@>832E,@>834A	Pointer to actual line
1AE4	832E			
1AE6	834A			
1AE8	D820	MOVB	@>1A97,@>834C	>66
1AEA	1A97			
1AEC	834C			
1AEE	C803	MOV	3,@>8350	Save R3
1AF0	8350			
1AF2	06A0	BL	@>1EAA	VPUSHG
1AF4	1EAA			
1AF6	C0E0	MOV	@>8350,3	Old R3
1AF8	8350			

```

1AFA 1001 JMP >1AFE
1AFC 04C3 CLR 3
1AFE 0288 CI 8,>C900 Token line number?
1B00 C900
1B02 16ED JNE >1ADE Error
1B04 06A0 BL @>1F7E Fetch byte
1B06 1F7E
1B08 D008 MOVB 8,0 Save R8
1B0A 06A0 BL @>1FA0 2nd byte and INC text pointer
1B0C 1FA0
1B0E 0603 DEC 3
1B10 1528 JGT >1B62 R3 >0
1B12 C060 MOV @>8330,1 Start line table
1B14 8330
1B16 D0A0 MOVB @>8389,2 GROM flag
1B18 8389
1B1A 1307 JEQ >1B2A
1B1C DB41 MOVB 1,@>0402(13) Write GROM address
1B1E 0402
1B20 C08D MOV 13,2
1B22 DB60 MOVB @>83E3,@>0402(13)
1B24 83E3
1B26 0402
1B28 1005 JMP >1B34
1B2A D7E0 MOVB @>83E3,*15 Write VDP address
1B2C 83E3
1B2E 0202 LI 2,>8800
1B30 8800
1B32 D7C1 MOVB 1,*15
1B34 8801 C 1,@>8332 End line list
1B36 8332
1B38 140B JHE >1B50
1B3A 9012 CB *2,0 Searched line ?
1B3C 1607 JNE >1B4C
1B3E 9212 CB *2,8
1B40 130A JEQ >1B56
1B42 D0D2 MOVB *2,3 Jump over line address
1B44 0221 AI 1,>0004
1B46 0004
1B48 D0D2 MOVB *2,3
1B4A 10F4 JMP >1B34 And go on
1B4C D0D2 MOVB *2,3 Jump over one bit
1B4E 10F9 JMP >1B42
1B50 0200 LI 0,>0303 Error
1B52 0303
1B54 10AF JMP >1AB4
1B56 05C1 INCT 1 Found line
1B58 C801 MOV 1,@>832E New actual line pointer
1B5A 832E
1B5C 0649 DECT 9 Decrease stack
1B5E 0460 B @>1976 EXEC
1B60 1976

1B62 06A0 BL @>1F7E Fetch one byte
1B64 1F7E
1B66 0288 CI 8,>B300 Token ,
1B68 B300
1B6A 16A2 JNE >1AB0 Error
1B6C 06A0 BL @>1F7E Next byte
1B6E 1F7E
1B70 10C6 JMP >1AFE Search

1B72 10B5 JMP >1ADE Error

1B74 06A0 BL @>1F2E VPOP
1B76 1F2E

```

```

1B78 9820 CB    @>1A97,@>834C  >66?  GOSUB stack entry?
1B7A 1A97
1B7C 834C
1B7E 16FA JNE   >1B74      Go on
1B80 D208 MOVB  8,8
1B82 16F7 JNE   >1B72      Error
1B84 C820 MOV   @>834A,@>832E  New actual line address
1B86 834A
1B88 832E
1B8A 0460 B     @>19E6      Address of substack and execute routine
1B8C 19E6

1B8E 0200 LI    0,>0006   GPL routine DEF
1B90 0006
1B92 1090 JMP   >1AB4      Error

1B94 06A0 BL    @>176A      XML >13 fetch name
1B96 176A
1B98 06A0 BL    @>187C      One byte from VDP in R1
1B9A 187C
1B9C 834A DATA >834A
1B9E 0A11 SLA   1,1      User defined function?
1BA0 11F6 JLT   >1B8E      Yes, execute GPL routine
1BA2 06A0 BL    @>1670      XML >14
1BA4 1670
1BA6 9820 CB    @>834C,@>19CB  String tag?
1BA8 834C
1BAA 19CB
1BAC 1302 JEQ   >1BB2
1BAE 06A0 BL    @>1890      8 Bytes from VDP over stack entry
1BB0 1890
1BB2 0460 B     @>1924      CONT
1BB4 1924

1BB6 06A0 BL    @>18DA      PARSE without GROM address
1BB8 18DA
1BBA B367 DATA >B367
1BBC 06A0 BL    @>1E70      String tag?
1BBE 1E70
1BC0 04C3 CLR   3
1BC2 0288 CI    8,>B000   Token THEN
1BC4 B000
1BC6 16D5 JNE   >1B72      Error
1BC8 0520 NEG   @>834A
1BCA 834A
1BCC 16CF JNE   >1B6C
1BCE 06A0 BL    @>1F7E      Fetch byte
1BD0 1F7E
1BD2 0288 CI    8,>C900   Token line number
1BD4 C900
1BD6 16CD JNE   >1B72      Error
1BD8 05E0 INCT @>832C      Text pointer
1BDA 832C
1BDC 06A0 BL    @>1F7E      Fetch byte
1BDE 1F7E
1BE0 0288 CI    8,>8100   Token ELSE
1BE2 8100
1BE4 13C3 JEQ   >1B6C      Go on
1BE6 0460 B     @>19E2      Address from substack and execute
1BE8 19E2

1BEA 06A0 BL    @>176A      XML >13 Fetch name
1BEC 176A
1BEE 06A0 BL    @>1670      XML >14 Built stack entry on FAC
1BF0 1670
1BF2 0288 CI    8,>BE00   Token =

```

```

1BF4 BE00
1BF6 16BD JNE >1B72 Error
1BF8 06A0 BL @>1F7E Fetch byte
1BFA 1F7E
1BFC 06A0 BL @>1E9C VPUSHG with changed return
1BFE 1E9C
1C00 8D30 DATA >8D30
1C02 06A0 BL @>1788 XML >15 Coordinate value
1C04 1788
1C06 0460 B @>1924 CONT
1C08 1924

1C0A 0000
1C0C 0000
1C0E 0000
1C10 0000
1C12 0000

1C14 06A0 BL @>176A XML >13 Fetch name and search
1C16 176A
1C18 C120 MOV @>834A,4
1C1A 834A
1C1C 06A0 BL @>1F2E VPOP
1C1E 1F2E
1C20 9820 CB @>834C,@>1BBB >67? For next entry on stack?
1C22 834C
1C24 1BBB
1C26 1302 JEQ >1C2C
1C28 0460 B @>1F78 Error
1C2A 1F78

1C2C 8804 C 4,@>834A
1C2E 834A
1C30 1304 JEQ >1C3A
1C32 6820 S @>1C68,@>836E >0010
1C34 1C68
1C36 836E
1C38 10F1 JMP >1C1C
1C3A 06A0 BL @>1890 8 Bytes from VDP over stack entry
1C3C 1890
1C3E 06A0 BL @>1E8C Set stack pointer and Basic byte
1C40 1E8C
1C42 06A0 BL @>0D84 SADD
1C44 0D84
1C46 06A0 BL @>1E7A Fetch stack pointer and Basic byte
1C48 1E7A
1C4A A820 A @>1C68,@>836E >0010
1C4C 1C68
1C4E 836E
1C50 06A0 BL @>1788 XML >15 Assign value
1C52 1788
1C54 6820 S @>1EAC,@>836E >0008
1C56 1EAC
1C58 836E
1C5A 06A0 BL @>0D42 SCOMP
1C5C 0D42
1C5E 02C4 STST 4
1C60 130A JEQ >1C76
1C62 C0E0 MOV @>836E,3
1C64 836E
1C66 0223 AI 3,>0010
1C68 0010
1C6A 06A0 BL @>1880 Read byte
1C6C 1880
1C6E D041 MOVB 1,1
1C70 1112 JLT >1C96

```

```

1C72 0A14 SLA 4,1
1C74 150E JGT >1C92
1C76 A820 A @>1F0E,@>836E >0018
1C78 1F0E
1C7A 836E
1C7C C0E0 MOV @>836E,3
1C7E 836E
1C80 0223 AI 3,>0006
1C82 0006
1C84 06A0 BL @>1880 Read byte
1C86 1880
1C88 D820 MOVB @>8800,@>832F Another byte in Lbyte actual line pointer
1C8A 8800
1C8C 832F
1C8E D801 MOVB 1,@>832E Complete line pointer
1C90 832E
1C92 0460 B @>1924 CONT
1C94 1924

1C96 0A14 SLA 4,1 Negative?
1C98 15EE JGT >1C76 Go on
1C9A 10FB JMP >1C92 Stop and end

```

Jump table for EXEC

```

1C9C 1A2C
1C9E 1A2C ELSE
1CA0 1A2C : :
1CA2 1A2C !
1CA4 1BB6 IF
1CA6 1A8E GO
1CA8 1AFC GOTO
1CAA 1AE0 GOSUB
1CAC 1B74 RETURN
1CAE 19E6 DEF
1CB0 19E6 DIM
1CB2 1A3C END
1CB4 8000 FOR
1CB6 1BEA LET
1CB8 8002 BREAK
1CBA 8004 UNBREAK
1CBC 8006 TRACE
1CBE 8008 UNTRACE
1CC0 8016 INPUT
1CC2 19E6 DATA
1CC4 8012 RESTORE
1CC6 8014 RANDOMIZE
1CC8 1C14 NEXT
1CCA 800A READ
1CCC 1A3C STOP
1CCE 803E DELETE
1CD0 19E6 REM
1CD2 1A92 ON
1CD4 800C PRINT
1CD6 800E CALL
1CD8 19E6 OPTION
1CDA 8018 OPEN
1CDC 801A CLOSE
1CDE 1A2C SUB
1CE0 803C DISPLAY

```

Jump table for PARSE

```

1CE2 801C (
1CE4 1A2C &
1CE6 1A2C
1CE8 1A2C OR
1CEA 1A2C AND

```


1CEC	1A2C	XOR
1CEE	1A2C	NOT
1CF0	1A2C	=
1CF2	1A2C	<
1CF4	1A2C	>
1CF6	801E	+
1CF8	8020	-
1CFA	1A2C	*
1CFC	1A2C	/
1CFE	1A2C	^
1D00	1A2C	
1D02	8010	STRING IN „“
1D04	1A5C	STRING
1D06	1A2C	Line number
1D08	804A	EOF
1D0A	8022	ABS
1D0C	8024	ATN
1D0E	8026	COS
1D10	8028	EXP
1D12	802A	INT
1D14	802C	LOG
1D16	802E	SGN
1D18	8030	SIN
1D1A	8032	SQR
1D1C	8034	TAN
1D1E	8036	LEN
1D20	8038	CHR\$
1D22	803A	RND
1D24	8040	SEG\$
1D26	8046	POS
1D28	8044	VAL
1D2A	8042	STR\$
1D2C	8048	ASC

Jump table	CONT	
1D2E	1D5C	=
1D30	1D3E	<
1D32	1D4C	>
1D34	1DEC	+
1D36	1E18	-
1D38	1E24	*
1D3A	1E30	/
1D3C	1E3C	^

1D3E	0202	LI	2,>0002	Flag in R2	0:=
1D40	0002				1:<>
1D42	0288	CI	8,>C000		2:<
1D44	C000				3:<=
1D46	1604	JNE	>1D50		4:>
1D48	0642	DECT	2		5:>=
1D4A	1005	JMP	>1D56		
1D4C	0202	LI	2,>0004		
1D4E	0004				
1D50	0288	CI	8,>BE00		
1D52	BE00				
1D54	1605	JNE	>1D60		
1D56	06A0	BL	@>1F7E		
1D58	1F7E				
1D5A	1001	JMP	>1D5E		
1D5C	0702	SETO	2		
1D5E	0582	INC	2		
1D60	05C9	INCT	9		
1D62	C642	MOV	2,*9	Flag on substack	
1D64	06A0	BL	@>1E9C	Return on substack and VPUSHG	
1D66	1E9C				

```

1D68 C000 DATA >C000      Token >
1D6A C119 MOV *9,4        Fetch flag
1D6C 0649 DECT 9
1D6E D324 MOVB @>1DA8(4),12  Fetch jump
1D70 1DA8
1D72 088C SRA 12,8
1D74 06A0 BL @>1E4A      FAC and stack same type?
1D76 1E4A
1D78 131A JEQ >1DAE      String jump
1D7A 06A0 BL @>0D42      SCOMP
1D7C 0D42
1D7E 046C B @>1D82(12) Jump
1D80 1D82
Compare result in FAC:
1D82 1504 JGT >1D8C      >=
1D84 1303 JEQ >1D8C      =
1D86 04C4 CLR 4
1D88 1003 JMP >1D90
1D8A 13FD JEQ >1D86      <>
1D8C 0204 LI 4,>BFFF
1D8E BFFF
1D90 0203 LI 3,>834A
1D92 834A
1D94 CCC4 MOV 4,*3+
1D96 04F3 CLR *3+
1D98 04F3 CLR *3+
1D9A 04F3 CLR *3+
1D9C 1039 JMP >1E10      End
1D9E 13F6 JEQ >1D8C      <=
1DA0 11F5 JLT >1D8C      <
1DA2 10F1 JMP >1D86
1DA4 15F3 JGT >1D8C      >
1DA6 10EF JMP >1D86
Jump
1DA8 0208 DATA >0208
1DAA 1E1C DATA >1E1C
1DAC 2200 DATA >2200
Compare string:
1DAE C2A0 MOV @>834E,10 Address first string
1DB0 834E
1DB2 D1E0 MOVB @>8351,7 Length
1DB4 8351
1DB6 06A0 BL @>1F2E      VPOP
1DB8 1F2E
1DBA C120 MOV @>834E,4 Address second string
1DBC 834E
1DBE D1A0 MOVB @>8351,6 Length
1DC0 8351
1DC2 D146 MOVB 6,5
1DC4 91C6 CB 6,7
1DC6 1101 JLT >1DCA
1DC8 D147 MOVB 7,5
1DCA 0985 SRL 5,8
1DCC 130D JEQ >1DE8
1DCE C0CA MOV 10,3
1DD0 058A INC 10
1DD2 06A0 BL @>1880      Read byte from VDP
1DD4 1880
1DD6 D001 MOVB 1,0
1DD8 C0C4 MOV 4,3
1DDA 0584 INC 4
1DDC 06A0 BL @>1880      Read byte from VDP
1DDE 1880
1DE0 9001 CB 1,0
1DE2 16CD JNE >1D7E      Not equal, end
1DE4 0605 DEC 5

```

```

1DE6 15F3 JGT >1DCE
1DE8 91C6 CB 6,7
1DEA 10C9 JMP >1D7E End
1DEC 06A0 BL @>1E9C VPUSHG
1DEE 1E9C
1DF0 C200 DATA >C200 Token -
1DF2 0202 LI 2,>0D84 SADD
1DF4 0D84
1DF6 04E0 CLR @>8354
1DF8 8354
1DFA 06A0 BL @>1E4A FAC and stack same type?
1DFC 1E4A
1DFE 1336 JEQ >1E6C String jump
1E00 06A0 BL @>1E8C Set stack pointer
1E02 1E8C
1E04 0692 BL *2 Execute addition
1E06 06A0 BL @>1E7A
1E08 1E7A
1E0A D0A0 MOVB @>8354,2
1E0C 8354
1E0E 1602 JNE >1E14 Error
1E10 0460 B @>1924 CONT
1E12 1924

1E14 0460 B @>1A40 Error and back to Basic
1E16 1A40

1E18 06A0 BL @>1E9C Vpushg
1E1A 1E9C
1E1C C200 DATA >C200 Token -
1E1E 0202 LI 2,>0D74 SSUB
1E20 0D74
1E22 10E9 JMP >1DF6
1E24 06A0 BL @>1E9C Vpushg
1E26 1E9C
1E28 C400 DATA >C400 Token /
1E2A 0202 LI 2,>0E8C SMUL
1E2C 0E8C
1E2E 10E3 JMP >1DF6
1E30 06A0 BL @>1E9C
1E32 1E9C
1E34 C400 DATA >C400 Token /
1E36 0202 LI 2,>0FF8 SDIV
1E38 0FF8
1E3A 10DD JMP >1DF6
1E3C 06A0 BL @>1E9C
1E3E 1E9C
1E40 C500 DATA >C500 Token ^
1E42 0200 LI 0,>0005 GPL routine involution
1E44 0005
1E46 0460 B @>1A30
1E48 1A30

```

FAC and stack entry same type?

```

1E4A C1A0 MOV @>836E,6 Fetch VDP address from FAC stack entry
1E4C 836E
1E4E 05C6 INCT 6 +2
1E50 D7E0 MOVB @>83ED,*15 Write address
1E52 83ED
1E54 1000 JMP >1E56
1E56 D7C6 MOVB 6,*15
1E58 1000 JMP >1E5A
1E5A 9820 CB @>8800,@>19CB >65 String tag?
1E5C 8800
1E5E 19CB
1E60 1A07 JL >1E70 < Jump

```

```

1E62 1B04 JH >1E6C > Error
1E64 9820 CB @>834C,@>19CB String tag?
1E66 834C
1E68 19CB
1E6A 1306 JEQ >1E78 Return
1E6C 0460 B @>1764 Set error
1E6E 1764

1E70 9820 CB @>834C,@>19CB String tag?
1E72 834C
1E74 19CB
1E76 14FA JHE >1E6C Error
1E78 045B B *11 Return

Substack pointer in R9 and actual Basic byte in R8
1E7A 04C8 CLR 8
1E7C D220 MOVB @>8342,8 Fetch Basic byte
1E7E 8342
1E80 D260 MOVB @>8373,9 Fetch substack pointer
1E82 8373
1E84 0989 SRL 9,8
1E86 0229 AI 9,>8300 Complete address substack
1E88 8300
1E8A 045B B *11 Return

Set substack pointer and Basic byte:
1E8C D808 MOVB 8,@>8342 Set Basic byte
1E8E 8342
1E90 0229 AI 9,>7D00
1E92 7D00
1E94 D820 MOVB @>83F3,@>8373 Set substack pointer
1E96 83F3
1E98 8373
1E9A 045B B *11 Return

VPUSHG with return on substack and back over GPL CONT
For DATA comparison
1E9C 05C9 INCT 9 Substack to big?
1E9E 0289 CI 9,>83BA
1EA0 83BA
1EA2 1B3F JH >1F22 Error
1EA4 C64B MOV 11,*9 Return on substack
1EA6 020B LI 11,>18E4 new return address
1EA8 18E4

VPUSHG (XML >17):
1EAA 0200 LI 0,>0008
1EAC 0008
1EAE A800 A 0,@>836E Add 8 to value stack pointer
1EB0 836E
1EB2 C060 MOV @>836E,1 Fetch pointer in R1
1EB4 836E
1EB6 D7E0 MOVB @>83E3,*15 Write address
1EB8 83E3
1EBA 0261 ORI 1,>4000 For writing
1EBC 4000
1EBE D7C1 MOVB 1,*15
1EC0 0201 LI 1,>834A Write data
1EC2 834A
1EC4 D831 MOVB *1+,@>8C00
1EC6 8C00
1EC8 0600 DEC 0 8 Bytes
1ECA 15FC JGT >1EC4
1ECC C00B MOV 11,0 Save return
1ECE 9820 CB @>834C,@>19CB String tag?
1ED0 834C

```

```

1ED2 19CB
1ED4 160E JNE >1EF2
1ED6 C1A0 MOV @>836E,6 Value stack pointer in R6
1ED8 836E
1EDA 0226 AI 6,>0004 +4
1EDC 0004
1EDE C060 MOV @>834A,1
1EE0 834A
1EE2 0281 CI 1,>001C String expression?
1EE4 001C
1EE6 1605 JNE >1EF2
1EE8 C060 MOV @>834E,1 Pointer to string
1EEA 834E
1EEC 1302 JEQ >1EF2
1EEE 06A0 BL @>18AA Mark string
1EF0 18AA
1EF2 C060 MOV @>836E,1 Value stack pointer
1EF4 836E
1EF6 0221 AI 1,>0010 +16
1EF8 0010
1EFA 8801 C 1,@>831A Pointer end free RAM for strings
1EFC 831A
1EFE 123B JLE >1F76 O.k. End
1F00 05C9 INCT 9
1F02 C640 MOV 0,*9 Return to substack
1F04 D820 MOVB @>18F0,@>8323 GPL routine garbage collection
1F06 18F0
1F08 8323
1F0A 06A0 BL @>1A4A
1F0C 1A4A
1F0E 0018 DATA >0018
1F10 C019 MOV *9,0 Fetch return from substack
1F12 0649 DECT 9
1F14 C060 MOV @>836E,1 Fetch value stack pointer
1F16 836E
1F18 0221 AI 1,>0010 Stack +16
1F1A 0010
1F1C 8801 C 1,@>831A Smaller end free RAM
1F1E 831A
1F20 122A JLE >1F76 O.k.
1F22 0200 LI 0,>0103 Error code
1F24 0103
1F26 06A0 BL @>1E7A Set substack and Basic byte
1F28 1E7A
1F2A 0460 B @>1A30 Error over GPL
1F2C 1A30

VPOP (XML >18)
1F2E 0202 LI 2,>834A FAC
1F30 834A
1F32 C060 MOV @>836E,1 Fetch pointer value stack
1F34 836E
1F36 8801 C 1,@>8324 Basis value stack?
1F38 8324
1F3A 121E JLE >1F78 Error
1F3C D7E0 MOVB @>83E3,*15 Write VDP address
1F3E 83E3
1F40 0200 LI 0,>0008 8 bytes
1F42 0008
1F44 D7C1 MOVB 1,*15
1F46 6800 S 0,@>836E Increase value stack
1F48 836E
1F4A DCA0 MOVB @>8800,*2+ Fetch stack entry
1F4C 8800
1F4E 0600 DEC 0 8 bytes
1F50 15FC JGT >1F4A

```

```

1F52 C00B MOV 11,0 Save return
1F54 9820 CB @>834C,@>19CB String tag?
1F56 834C
1F58 19CB
1F5A 160D JNE >1F76 No, end
1F5C 04C6 CLR 6
1F5E C0E0 MOV @>834A,3
1F60 834A
1F62 0283 CI 3,>001C String expression?
1F64 001C
1F66 13C0 JEQ >1EE8 Clear
1F68 06A0 BL @>1880 Correct string address from variable list
1F6A 1880
1F6C D820 MOVB @>8800,@>83E3 2nd byte in R1 Lbyte
1F6E 8800
1F70 83E3
1F72 C801 MOV 1,@>834E on >834E
1F74 834E
1F76 0450 B *0 Return

1F78 0200 LI 0,>0403 Error code
1F7A 0403
1F7C 10D4 JMP >1F26 Print error

```

Fetch 1 byte from VDP or GROM in R8

```

1F7E D220 MOVB @>8389,8 Flag GROM
1F80 8389
1F82 1607 JNE >1F92 Yes, jump
1F84 D7E0 MOVB @>832D,*15 Write VDP address
1F86 832D
1F88 020A LI 10,>8800 VDP read data
1F8A 8800
1F8C D7E0 MOVB @>832C,*15
1F8E 832C
1F90 1007 JMP >1FA0
1F92 DB60 MOVB @>832C,@>0402(13) Write GROM address
1F94 832C
1F96 0402
1F98 DB60 MOVB @>832D,@>0402(13)
1F9A 832D
1F9C 0402
1F9E C28D MOV 13,10 Grom read data
1FA0 05A0 INC @>832C New text pointer
1FA2 832C
1FA4 D21A MOVB *10,8 Read byte
1FA6 045B B *11 Return

```

Fetch 8 bytes from VDP stack(Address STACK=>836E)

```

1FA8 0205 LI 5,>FFF8 8 Bytes
1FAA FFF8
1FAC 0206 LI 6,>835C Address ARG
1FAE 835C
1FB0 D7E0 MOVB @>836F,*15 Write address
1FB2 836F
1FB4 0207 LI 7,>8800 VDPRD
1FB6 8800
1FB8 D7E0 MOVB @>836E,*15
1FBA 836E
1FBC A805 A 5,@>836E STACK -8!
1FBE 836E
1FC0 DD97 MOVB *7,*6+ Fetch bytes
1FC2 0585 INC 5
1FC4 16FD JNE >1FC0
1FC6 045B B *11 Return

```

Read 1 byte from VDP in R8 (Address R6)

```

1FC8 D7E0 MOVB @>83ED,*15 Write VDP address
1FCA 83ED
1FCC 1000 JMP >1FCE
1FCE D7C6 MOVB 6,*15 Complete
1FD0 0586 INC 6
1FD2 D220 MOVB @>8800,8 Read byte
1FD4 8800
1FD6 0988 SRL 8,8 In Lbyte R8
1FD8 045B B *11 Return

Read 1 byte from GROM in R8 (Address in R6)
1FDA DB46 MOVB 6,@>0402(13) Write GROM address
1FDC 0402
1FDE DB60 MOVB @>83ED,@>0402(13)
1FE0 83ED
1FE2 0402
1FE4 0586 INC 6
1FE6 D21D MOVB *13,8 Read byte
1FE8 10F6 JMP >1FD6 In Lbyte R8 and return

1FEA 0000
1FEC 0000
1FEE 0000
1FF0 0000
1FF2 0000
1FF4 0000
1FF6 0000
1FF8 0000
1FFA 0000

1FFC 2A61 DATA >2A61 Check sum
1FFE A38A DATA >A38A

```

The Graphics Programming Language

GPL is a processor related language for the TI99/4A. Many of the GPL commands are almost identical with Assembler commands of the TMS 9900, the processor for the TI99/4A when using an additional interpreter. The programs of this language are located in so called GROM'S. These are elements in Memory Map Technique which can be read over certain CPU addresses.

GPL uses essentially the area of the CPU-RAM's >8372 through >83FF. The work space for the GPL interpreter is located at 8370. The pointer for the GPL data stack is located at >8370 and the pointer subroutine stack is located at 8372. The complete address for the stack consists of the pointer plus >8300. Usually the ROM area >8380 through >83BF is used for the stacks.

The GPL-Statusbyte is located at >837C. It is analogous to the statusbyte of the TMS 9900 processor. Following are the definitions of the bits;

Bit 0 : (MSB) High-Bit

Bit 1 : Greater-Bit

Bit 2 : Equal-Bit, called Condition-Bit in GPL

Bit 3 : Carry-Bit

Bit 4 : Overflow-Bit

The bits number 5 through 7 (LSB) are not used.

GPL offers a special access to the screen. A buffer for the screen is located at >837D. By writing on the buffer or reading from it, the screen is automatically accessed. The pointers for the row and column are located at >837F and >837E.

The GROM listings have been worked out with the GPL disassembler. The disassembler chose the source operand and the destination operand in reverse order from what has been used in the following explanations to the GPL commands. Please pay attention when reading the GROM listing.

THE GPL COMMANDS

Op-Code: >00 Description: RTN

Format type:3

Description: Takes the highest value of the substacks and sets the program counter (new GROM address). Reset condition-bit into status byte.

GPL-Statusbyte: Condition-bit reset

Op-Code: >01 Description: RTNC

Format type:3

Description: Same as RTN but the condition-bit is not influenced.

GPL-Statusbyte: Not influenced

Op-Code: >02 Description: RAND IMM

Format type:2 Result: Random number at >8378

Description: Creates a random number at >8378. The IMM shows the maximum. The minimum is always 0.

GPL-Statusbyte: Not influenced

Op-Code: >03 Description: SCAN

Format type:3 Result: Key value at >8375, Joystick
 values at >8376,>8377.

Description: Scans the keyboard (Modus at >8374) and sets the corresponding values at >8375 (Keys) and >8376/>8377 (Joystick).

GPL-Statusbyte: Condition-bit is set by pushing a new key.

Op-Code: >04 Description: BACK IMM

Format type:2 Result: VDP-Register 7 = IMM

Description: Sets the background color of the screen on the value of IMM.

GPL-Statusbyte: Not influenced

Op-Code: >05 Description: B IMM (2 Bytes)

Format type:2

Description: Jump to the absolute address of the IMM-value. Program counter takes the value of IMM.

GPL-Statusbyte: Condition-Bit reset

Op-Code: >06 Description: CALL IMM (2 Bytes)

Format type:2

Description: Jump to a subroutine. The program counter takes the value of IMM, the old value of the program counter is stored on the substack.

GPL-Statusbyte: Condition-Bit reset

Op-Code: >07 Description: ALL IMM

Format type:2

Description: Screen is filled with the IMM value.

GPL-Statusbyte: Not changed

Op-Code: >08 Description: FMT several operands

Description: Special output command for the screen. The FMT Interpreter is independent of the GPL Interpreter. (See ROM-Listing >04DE through >05A1)

GPL-Statusbyte: Not influenced

Op-Code: >09 Description: H

Format type:3 Result:Condition-Bit = H-Bit

Description: Checks the High-Bit in GPL-Statusbyte and sets the Condition-Bit accordingly.

GPL-Statusbyte: Condition-Bit is set on value of High-Bit.

Op-Code: >0A Description: GT

Format type:3 Result:Condition-Bit = GT-Bit

Description: Checks the Greater-Bit in the GPL-Status Byte and sets the Condition-Bit accordingly.

GPL-Statusbyte: Condition-Bit is set on value of Greater-Bit.

Op-Code: >0B Description: EXIT

Format type:3

Description: Software reset, returns to Master Titel Screen or Power-up routine.

GPL-Statusbyte: Not influenced

Op-Code: >0C Description: CARRY

Format type:3 Result:Condition-Bit = Carry-Bit

Description: Tests the Carry-Bit in the GPL-Statusbyte and sets the Condition-Bit accordingly.

GPL-Statusbyte: Condition-Bit is set on value of Carry-Bit.

Op-Code: >0D Description: OVF

Format-type:3 Result:Condition-Bit = OVF-Bit

Description: Tests the Overflow-Bit in the GPL-Statusbyte and sets the Condition-Bit accordingly.

GPL-Statusbyte: Condition-Bit is set on value of Overflow-Bit.

Op-Code: >0E Description: PARSE IMM

Format type:2

Description: Extension in Basic-Interpreter until a character appears in Basic which is smaller than the IMM value or decimal point. Is mostly used in GPL subprograms for Basic to store contents of variable on FAC.

Op-Code: >0F Description: XML IMM

Format type:2

Description: Execution of an assembler routine according to the table values of XMR. (See explanation to ROM)

GPL-Statusbyte: Depends on assembler routine

Op-Code: >10 Description: CONT

Format type:3

Description: Leads back to the Basic-Interpreter, is used at the end of the Basic routines (not the subprogram) which are located in GROM.

GPL-Statusbyte: Not influenced

Op-Code: >11 Description: EXEC

Format type:3

Description: Starts with execution of a Basic program.

GPL-Statusbyte: Not influenced

Op-Code: >12 Description: RTNB

Format type:3

Description: Leads back to Basic interpreter, return-address on substack.

GPL-Statusbyte: Not influenced

Op-Code: >14->1F Description: XGPL
>98->9F
>F0->F4
>FC->FF

Description: For GPL etensions. Contains in the interpreter up to now, is starting of a DSR on CRU address >1B00 and then the jump B >4020 or b >401C (Code 1F).

GPL-Statusbyte:

Op-Code: >20->3F Description: MOVE S1, from S2 to D

Format type:5

Description: Moves certain numbers of bytes (S1) from address S2 to destination address (D). The VDP register as well as GROM address can also be used as destination address.

GPL-Statusbyte: Not influenced

Op-Code: >40->5F Description: BR IMM

Format type:4

Description: Jumps to a certain address (only possible within a GROM) when the condition bit has not been set.

GPL-Statusbyte: Condition bit reset

Op-Code: >60->7F Description: BS IMM

Format type:4

Description: Jumps to a certain address (only possible in a GROM) when condition bit is set.

GPL-Statusbyte: Condition-Bit reset

Op-Code: >80->81 Description: ABS D
DABS D

Format type:5 Result: D = ABS(D)

Description: Replaces D by the absolut value of D

GPL-Statusbyte: Not influenced

Op-Code: >82->83 Description: NEG D
DNEG D

Format type:5 Result: D = -D

Description: Replaces destination by the two's compliment of destination.

GPL-Statusbyte: Not influenced

Op-Code: >84->85 Description: INV D
DINV D

Format type:5 Result: D = \overline{D}

Description: Inverts each bit in the destination.

GPL-Statusbyte: Not influenced

Op-Code: >86->87 Description: CLR D
DCLR D

Format type:5 Result: D = 0

Description: Sets destination on 0

GPL-Statusbyte: Not influenced

Op-Code: >88-(>89) Description: FETCH D
incompletely decoded

Format type:5

Description: Fetches a byte on which the return address on the subroutine-stack shows. Puts this byte in the destination and increases the return address on the substack by 1.

GPL-Statusbyte: Not influenced

Op-Code: >8A->8B Description: CASE D
DCASE D

Format type:5

Description: Adds twice the value of D to the Program Counter
(GROM address)

GPL-Statusbyte: Condition-Bit reset

Op-Code: >8C-(>8D) Description: PUSH D
incompletely decoded

Format type:5

Description: Puts bits from D on the GPL data stack and increases
data stack pointer one point. A pop can be realized with ST
*>837C,D. (specialty in the interpreter)

GPL-Statusbyte: Not influenced

Op-Code: >8E->8F Description: CZ D
DCZ D

Format type:5 Result: Condition-bit = 1 if D = 0

Description: Compairs D with 0 and sets the condition bit, if D
equals 0.

GPL-Statusbyte: Condition-Bit set if S = 0

Op-Code: >90->91 Description: INC D
DINC D

Format type:5 Result: D = D + 1

Description: Increases D by one

GPL-Statusbyte: GPU-Statusbyte equals GPL-Statusbyte

Op-Code: >92->93 Description: DEC D
DDEC D

Format type:5 Result: $D = D - 1$

Description: One point is subtraction from D

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >94->95 Description: INCT D
DINCT D

Format type:5 Result: $D = D + 2$

Description: D increased by 2

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >96->97 Description: DECT D
DDECT D

Format type:5 Result: $D = D - 2$

Description: D decreased by 2

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >A0->A3 Description: ADD S,D
DADD S,D

Format type:1 Result: $D = S + D$

Description: Adds source to destination and stores the result in the destination.

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >A4->A7 Description: SUB S,D
DSUB S,D

Format type:1 Result: $D = D - S$

Description: Subtracts source from destination and stores the result in the destination.

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >A8->AB Description: MUL S,D
DMUL S,S

Format type:1 Result: $D(D,D+1) = S * D$
or at Word: $D(D,D+2) = S * D$

Description: Source and destination are multiplied by each other. During a byte operation, the result is stored in the word of the destination; during a word operation, the result is stored in 2 words of the destination.

GPL-Statusbyte: Not influenced

Op-Code: >AC->AF Description: DIV S,D
DDIV S,D

Format type:1 Result: $D=D(D,D+1)/S$; D+1=remainder
or at word always +2

Description: Replaces the destination by the quotient of the destination by the source and the destination +1 (at word destination +2) by the remains of the destination divided by source.

GPL-Statusbyte: Not influenced

Op-Code: >B0->B3 Description: AND S,D
DAND S,D

Format type:1 Result: $D = S \text{ AND } D$

Description: Executes a AND operation by bits and stores the result in the destination.

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >B4->B7 Description: OR S,D
DOR S,D

Format type:1 Result: D = S OR D

Description: Executes an OR operation by bits and stores the result in the destination.

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >B8->BB Destination: XOR S,D
DXOR S,D

Format type:1 Result: D = S EXOR D

Description: Executes an exclusive OR operation by bits and stores the result in the destination.

GPL-Statusbyte: CPU-Statusbyte equals GPL-Statusbyte

Op-Code: >BC->BF Description: ST S,D
DST S,D

Format type:1 Result: D = S

Description: Replaces the destination by the source operand.

GPL-Statusbyte: Not influenced

Op-Code: >C0->C3 Description: EX S,D

Format type:1 Result: D = S, S = D

Description: Exchanges source and destination.

GPL-Statusbyte: Not influenced

Op-Code: >C4->C7 Description: CH S,D
DCH S,D

Format type:1

Description: Compares source and destination and sets the condition-Bit when the destination is logically greater than the source.

GPL-Statusbyte: Condition-Bit set according to comparison.

Op-Code: >C8->CB Description: CHE S,D
DCHE S,D

Format type:1

Description: Compares source and destination and sets the Condition-Bit when the destination is logically greater than or equal to the source.

GPL-Statusbyte: Condition-Bit set according to comparison.

Op-Code: >CC->CF Destination: CGT S,D
DCGT S,D

Format type:1

Description: Compares source and destination and sets the Condition-Bit if the destination is arithmetically greater than the source.

GPL-Statusbyte: Condition-Bit set according to comparison.

Op-Code: >D0->D3 Destination: CGE S,D
DCGE S,D

Format type:1

Description: Compares source and destination and sets the Condition-Bit if the destination is greater than or equal to the source.

GPL-Statusbyte: Condition-Bit set according to comparison.

Op-Code: >D4->D7 Description: CEQ S,D
DCEQ S,D

Format type:1

Description: Compares source and destination and sets the Condition-Bit if the destination and the source are equal.

GPL-Statusbyte: Condition-Bit set if source and destination are equal.

Op-Code: >D8->DB Description: CLOG S,D
DCLOG S,D

Format type:1

Description: Executes an AND operation by bits between destination and source and sets the Condition-Bit if the result is 0.

GPL-Statusbyte: Condition-Bit set if S AND D = 0

Op-Code: >DC->DF Description: SRA S,D
DSRA S,D

Format type:1

Description: The contents of the destination is moved to the right according to the number of bits of the source. The empty bit digits are filled with the MSB of the destination.

GPL-Statusbyte: Not influenced

Op-Code: >E0->E3 Description: SLL S,D
DSL L S,D

Format type:1

Description: The contents of the destination is moved to the left according to the number of bits of the source.

GPL-Statusbyte: Not influenced

Op-Code: >E4->E7 Description: SRL S,D
DSRL S,D

Format type:1

Description: The contents of the destination is moved to the left according to the number of bits of the source.

GPL-Statusbyte: Not influenced

Op-Code: >E8->EB Description: SRC S,D
DSRC S,D

Format type:1

Description: The contents of the destination will be cyclically moved to the right according to the number of bits in the source.

GPL-Statusbyte: Not influenced

Op-Code: >ED Description: COINC S,D
incompletely decoded from >EC to >EF

Format type:1

Description: Condition-Bit is set if points of 2 objects on the screen overlap. COINC requires special tables in GROM.

GPL-Statusbyte: Condition-Bit set in case of overlapping

Op-Code: >F4->F7 Description: I/O S,D

Format type:1

Description: I/O is a special command. The destination is the address of a list whose format depends on the output or input function. The source choses the function. Today the following values are permitted:

- 0 = Sound in GROM
- 1 = Sound in VDP
- 2 = CRU input
- 3 = CRU output
- 4 = Write cassette
- 5 = Read cassette
- 6 = Verify cassette

Op-Code: >13 Description: RTGR

Format type:3

Description: Takes the old GRMRD and the old program counter from the substack and resets GROM.

GPL-Statusbyte: Condition-Bit reset

Op-Code: >F8->FB Description: SWGR S,D

Format type:1

Description: Switches the GROM-read address (CPU). The source is the new GRMRD and destination is the program counter (GROM). Old PC and GRMRD are put on substack.

GPL-Statusbyte: Condition-Bit reset

THE GPL COMMAND FORMAT

Format type 1:

	MSB	0	1	2	3	4	5	6	7	LSB
	-	-	-	-	-	-	-	-	-	-
1 Byte	1	X	X	X	X	X	U	W		

Destination operand

Source operand

Format type 2:

	MSB	0	1	2	3	4	5	6	7	LSB
	-	-	-	-	-	-	-	-	-	-
1 Byte	0	0	0	X	X	X	X	X		

Immediate operand (byte or word).

Format type 3:

	MSB	0	1	2	3	4	5	6	7	LSB
		-	-	-	-	-	-	-	-	
1 Byte		0	0	0	X	X	X	X	X	

Format type 4:

	MSB	0	1	2	3	4	5	6	7	LSB
		-	-	-	-	-	-	-	-	
1 Byte		0	1	X	Address					

Address (1 byte)

Format type 5:

	MSB	0	1	2	3	4	5	6	7	LSB
		-	-	-	-	-	-	-	-	
1 Byte		1	0	0	X	X	X	X	W	

Destination operand

Format type 6:

	MSB	0	1	2	3	4	5	6	7	LSB
		-	-	-	-	-	-	-	-	
1 Byte		0	0	1	G	R	V	C	N	

Number

Destination operand

source operand

Explanation of indizes:

Format type 1 and 5:

W --> 0=Byte operation
1=Word operation (2 bytes)

U --> 0=No direct operand
1=Direct operand (IMM)

Format type 6:

G --> 0=Destination operand is GROM(only possible with GRAMS)
1=Destination operand is no GROM

R --> 0=Destination operand is not a VDP register
1=Destination operand is VDP register

V --> 0=Source operand is not the VDP RAM or CPU RAM
1=Source operand is the VDP RAM or CPU RAM

C --> 0=Source is not GROM addressed over CPU RAM
1=Source is GROM indicated or addressed over
CPU RAM

N --> 0=Number is not direct operand
1=Number is direct operand

The destination operands and source operands have five different forms:

	MSB	0	1	2	3	4	5	6	7	LSB	
1		0									CPU RAM is directly addressed >8300 through >837F
2		1	0	V	I	Address					V --> 0=CPU RAM, 1=VDP RAM I --> 0=Direct, 1=Indirect
3		1	1	V	I	Address					Same as number 2, but indicated.
						Address					Index
4		1	0	V	I	1	1	1	1		Extended area at 0 through 65535 Address with offset >8300, i.e. >DD00 corresponds to address >6000.
						Address					
5		1	1	V	I	1	1	1	1		Like number 4, only indicated.
						Address					
						Address					
						Index					

THE GROM 0

There are several versions of GROM 0 which differ especially in the power up routine. The listing indicates the most important differences. The attached hex dump listing can serve as comparison.

GROM 0 contains the power up routine, several mathematical routines and the cassette DSR. We do not comment on the mathematical routines since their use is described in the handbook of the Editor/Assembler.

At first the power up routine sets several pointers. Then the power up routines of the several GROM'S and of the DSR are executed. At last the Master Title Screen is produced. The build up of a program list is produced by pushing a key and by pushing of the corresponding key the chosen program is started.

The cassette routine, starting at G>1310 functions like an DSR and thus gives to the interested user exact information, how DSR work for the Periphery.

```

*****
*
*           CONSOLE GROM 0 ANALYSIS TI99/4A
*
*           9.6.84 H. Martin
*
*****

```

Header:

```

0000 : DATA >AA02 Header
0002 : DATA >0000
0004 : DATA >0000 Power-up (isn't here)
0006 : DATA >0000 Program (isn't here)
0008 : DATA >1310 DSR
000A : DATA >1320 Subroutine
000C : DATA >0000 Interrupt (isn't here)
000E : DATA >0000

```

Jump table:

```

0010 : BR GROM@>03D9 GROM DSRLNK
0012 : BR GROM@>0439 GSR return
0014 : BR GROM@>09A5 Number to string
0016 : BR GROM@>0393 Load standard character set
0018 : BR GROM@>039B Load small capital letters
001A : BR GROM@>0443 Warning information
001C : BR GROM@>0446 Error information
001E : BR GROM@>0449 Execution of a Basic program in GROM
0020 : BR GROM@>004F Power-up routine
0022 : BR GROM@>11FA Greatest integer
0024 : BR GROM@>0C7E Involution routine
0026 : BR GROM@>0D55 Square root
0028 : BR GROM@>0DB0 Exponent
002A : BR GROM@>0E60 LOG
002C : BR GROM@>0EF5 COS
002E : BR GROM@>0EFD SIN
0030 : BR GROM@>0F5B TAN
0032 : BR GROM@>0F7C ATN
0034 : BR GROM@>03CB Accept tone
0036 : BR GROM@>03D3 Bad response tone
0038 : B GROM@>4D12 Get string space
003B : BR GROM@>125A Bit reversal routine
003D : BR GROM@>0414 GSRLNK (same as DSRLNK, but only routines in GROM)
003F : B GROM@>2844 Reserved space in VDP RAM
0042 : B GROM@>37B4 Set program pointers
0045 : DATA >60
0046 : DATA >0D
0047 : DATA >00
0048 : DATA >11
0049 : DATA >00
004A : BR GROM@>03BF Load lower case set
004C : DATA >B0 Here change in some GROMS: Set pointers to the
004D : DATA >B0 character blocks. The other routines are shifted
004E : DATA >70 3 bytes upwards.

```

Power up routine

```

004F : DCLR @>83CE Clear sound bytes
0052 : ST @>9400,>70 Load speech write
0057 : ST @>8400,>9F Set sound generators
005B : ST @>8400,>BF
005F : ST @>8400,>DF
0063 : ST @>8400,>FF
0067 : DST @>8372,>FF7E Load data/substack
006B : MOVE >0007 TO REG>01 FROM GROM@>044E Load VDP register
0071 : CLR @>8300
0073 : MOVE >0071 TO @>8301 FROM @>8300 Clear scratch pad >00->71
0078 : MOVE >003E TO @>8382 FROM @>8300 >82->C0

```

```

007E : MOVE    >000B TO @>8374 FROM @>8300          >74->7F
0083 : MOVE    >0008 TO @>83C2 FROM @>8300          >C2->CA
0089 : DST     @>8303,>0308      9901 Set CRU
008D : I/O    @>8302,>03
0090 : DST     @>8303,>1001
0094 : I/O    @>8302,>03
0097 : ST     @>8303,>18
009A : I/O    @>8302,>03
009D : INV    @>8300
009F : ST     @>8303,>02
00A2 : I/O    @>8302,>03
00A5 : ST     @>8303,>01
00A8 : I/O    @>8302,>03
00AB : DST     @>8303,>1602
00AF : I/O    @>8302,>03
00B2 : CALL   GROM@>03CB      Print accept tone
00B5 : CLR    VDP@>0000      Check VDP RAM
00B8 : ST     @>8370,>10
00BB : ST     VDP*>8370,>A0
00BF : CZ     VDP@>0000
00C2 : BR     GROM@>00D9
00C4 : MOVE   >0001 TO REG>01 FROM GROM@>044C
00CA : CLR    VDP*>8370
00CD : ADD    @>8370,@>8370    4, 8 or 16K?
00D0 : CEQ    @>8370,>40
00D3 : BR     GROM@>00BB
00D5 : ST     @>83FD,>08      System flags (16k flag)
00D9 : DDEC   @>8370        Pointer end VDP RAM
00DB : MOVE   >0001 TO REG>01 FROM GROM@>0241
00E1 : CLR    VDP@>0000
00E4 : MOVE   >0FFF TO VDP@>0001 FROM VDP@>0000 Clear VDP
00EB : MOVE   >0020 TO VDP@>0380 FROM GROM@>0455 Load color table
00F2 : MOVE   >0200 TO VDP@>0900 FROM GROM@>04B0 Standard character set
00F9 : MOVE   >0050 TO VDP@>0808 FROM GROM@>094C Special signs
0100 : ALL    >20          Clear screen
0102 : ST     @>837E,>05      Load line screen
0105 : ST     @>8374,@>837E Keyboard mode 5
0108 : SCAN
0109 : DEC    @>837E
010B : BR     GROM@>0105      Check all keyboard modes
010D : DCLR   @>837E
010F : ST     @>8375,>60      >60 on ASCII value key
0112 : FMT    Colorbar on screen
0113 : ... FOR 02
0114 : ... 01('@>8375')
0116 : ... END FOR GROM@>0114
0119 : ... 11^
011A : ... 1E<
011B : ... FOR 02
011C : ... 01('@>8375')
011E : ... END FOR GROM@>011C
0121 : ... END FMT
0122 : SUB    @>837E,>12
0125 : ADD    @>8375,>08
0128 : CEQ    @>8375,>E0      All fields?
012B : BR     GROM@>0112
012D : CEQ    @>837E,>03
0130 : BR     GROM@>010F
0132 : DCLR   @>837E
0134 : FMT    Letters on screen
0135 : ... 05^
0136 : ... 0F<
0137 : ... '>01,>02,>03'    TI characters
013B : ... 1D<
013C : ... '>04,>05,>06'
0140 : ... 1D<

```

```

0141 : ... '>07,>08,>09'
0145 : ... 08^
0146 : ... 10<
0147 : ... ':READY-PRESS ANY KEY TO BEGIN:'
0164 : ... END FMT
0165 : MOVE >0011 TO VDP@>0128 FROM GROM@>0492 Texas Instruments
016C : MOVE >0018 TO VDP@>02C4 FROM GROM@>048B c Texas Instruments
0173 : MOVE >000D TO VDP@>016A FROM GROM@>04A3 Home Computer
017A : ST @>8343,>10
017D : CALL GROM@>0379 Check if GROM exists on >6000
0180 : DCLR @>83D0 GROM search pointer clear
0183 : CLR @>8355
0185 : ST @>836D,>04
0188 : XML >19 Search power-up LINK's and execute DSR ROM's
018A : BS GROM@>0188
018C : DST @>8372,>0080 Stacks initialize
0190 : DST *>8372,>019F Data stack new GROM address
0195 : XML >1A Execute power-up LINK's GROM
0197 : DST *>8373,*>8372 Data from data stack to substack(new RTN address!!)
019C : DECT @>8372 Data stack new value (>FE!!)
019E : RTN

```

Go on after initializing:

```

019F : DCZ @>83D0 GROM search pointer 0
01A2 : BR GROM@>018C No, start power LINK GROM
01A4 : MOVE >0001 TO REG>01 FROM GROM@>044D Load VDP register 1
01AA : CLR @>8374 Mode 0
01AC : RAND >FF Random number in >8378
01AE : SCAN
01AF : BR GROM@>01AC Wait for pressed key
01B1 : CALL GROM@>03CB Accept tone
01B4 : ALL >20 Clear screen
01B6 : ST @>8372,>FE Data stack >FE
01B9 : ST @>836D,>06 Program LINK
01BC : CLR @>836C
01BE : CLR @>83FB
01C1 : MOVE >001E TO VDP@>0400 FROM GROM@>6000
01C8 : ST @>83FB,>04 GRMRD +4
01CC : MOVE >001E TO VDP@>0420 FROM GROM@>6000
01D3 : CLR @>8358 Oh, test on "Super Modul Expander"
01D5 : CLR @>8359
01D7 : B GROM@>01DC
01DA : INC @>8359
01DC : CGT @>8359,>1D
01DF : BS GROM@>01ED
01E1 : CEQ VDP@>0400(@>8358),VDP@>0420(@>8358) Are GROM's parallel?
01E8 : BR GROM@>01EF
01EA : B GROM@>01DA
01ED : BR GROM@>01FD
01EF : INCT @>8372
01F1 : DCLR *>8372
01F4 : INCT @>8372
01F6 : DST *>8372,>12A1 >12A1 on data stack
01FB : INC @>836C 1st program
01FD : CEQ @>6000,>AA ROM Header? OH!!! / Not at GROM V2.2
0202 : BR GROM@>0224 /
0204 : DST @>8358,@>6006 Next program start / The other shifted
0209 : DCZ @>8358 / accordingly
020B : BS GROM@>0224 /
020D : INCT @>8372 /
020F : DST *>8372,>FFFF Flag on stack / This change eliminates
0214 : INCT @>8372 / the ROM moduls
0216 : DST *>8372,@>8358 Address on stack /
021A : INC @>836C 2nd program? /
021C : DST @>8358,@>0000(@>8358) /
0222 : BR GROM@>0209 /

```

```

0224 : INCT  @>8372          Increase stack
0226 : DCLR  *>8372          Search program in GROM
0229 : XML   >1A             GSRLNK
022B : BS    GROM@>0224      Loop till all
022D : DECT  @>8372
022F : DCEQ  @>8302,>12A1
0233 : BR    GROM@>0240
0235 : MOVE  >0001 TO @>8359 FROM GROM@>6000 GROM here?
023B : CEQ   @>8359,>AA
023E : BR    GROM@>01B6      New start
0240 : MOVE  >0001 TO REG>01 FROM GROM@>044E Screen enable
0246 : FMT                    Load screen
0247 : ... 01^
0248 : ... 02<
0249 : ... '>01,>02,>03'      TI characters
024D : ... 1D<
024E : ... '>04,>05,>06'
0252 : ... 1D<
0253 : ... '>07,>08,>09'
0257 : ... 01^
0258 : ... 1F<
0259 : ... ':PRESS:'
025F : ... END FMT
0260 : MOVE  >0011 TO VDP@>0028 FROM GROM@>0492 Texas Instruments
0267 : MOVE  >000D TO VDP@>0068 FROM GROM@>04A3 Home Computer
026E : DST   @>8352,>00E4      Address screen
0272 : ST    @>8358,>30        Number
0275 : CZ    @>836C           No program?
0277 : BR    GROM@>0290      Jump, if program
0279 : FMT                    No program, then message: Insert cartridge
027A : ... XPT=>02
027C : ... ':INSERT CARTRIDGE:'
028D : ... END FMT
028E : BR    GROM@>02EC
0290 : INC   @>8358
0292 : ST    VDP*>8352,@>8358 Write number
0296 : DINCT @>8352          VDP address
0298 : MOVE  >0003 TO VDP*>8352 FROM GROM@>0949 Text "FOR"
029F : DADD  @>8352,>0004      VDP address
02A3 : DST   @>836A,*>8372
02A7 : DECT  @>8372
02A9 : DST   @>835C,*>8372
02AD : DECT  @>8372
02AF : DADD  @>836A,>0004
02B3 : CLR   @>835E
02B5 : CZ    @>835C
02B7 : BS    GROM@>02CD
02B9 : MOVE  >0001 TO @>835F FROM @>0000(@>836A)      Length byte and
02C1 : DINC  @>836A
02C3 : MOVE  @>835E TO VDP*>8352 FROM @>0000(@>836A)  Text from ROM
02CB : BR    GROM@>02DD
02CD : MOVE  >0001 TO @>835F FROM GROM@>0000(@>836A)  Length byte and text
02D4 : DINC  @>836A
02D6 : MOVE  @>835E TO VDP*>8352 FROM GROM@>0000(@>836A) from GROM
02DD : DADD  @>8352,>003A      New VDP address
02E1 : CGE   @>8372,>00        Stack 0
02E4 : BS    GROM@>0290      No, next program
02E6 : ST    @>8343,>13        >6013
02E9 : CALL  GROM@>0379      on substack, if GROM exists on >6000
02EC : MOVE  >0001 TO REG>01 FROM GROM@>044D Screen enable
02F2 : CLR   @>8374          Mode 0
02F4 : RAND  >FF            Random number
02F6 : SCAN                    Keyboard scanning
02F7 : BR    GROM@>02F4      till key is pressed
02F9 : SUB   @>8375,>31        Integer from ASCII
02FC : CHE   @>8375,@>836C    Wrong key

```

```

02FF : BR      GROM@>0307
0301 : CALL    GROM@>03D3      Bad tone
0304 : B       GROM@>02F4      New key
0307 : CALL    GROM@>03CB      Accept tone
030A : SUB     @>836C,@>8375
030D : DEC     @>836C
030F : SLL    @>836C,>02
0312 : ST     @>8378,*>836C  Flag in >8378
0316 : INCT   @>836C
0318 : DST    @>835C,*>836C  Start address in>835C
031C : DINCT  @>835C
031E : DST    @>8372,>9E80    New stack pointer
0322 : CZ     @>8378          Check flag
0324 : BS     GROM@>032F
0326 : DST    @>8380,@>0000(@>835C) Start address from ROM
032D : BR     GROM@>0337
032F : MOVE   >0002 TO @>8380 FROM GROM@>0000(@>835C) Start from GROM
0337 : ALL    >20              Clear screen
0339 : DCZ    @>83CE          Wait for sound byte
033C : BR     GROM@>0339
033E : DCGT   @>8370,>1000    Check VDP RAM
0342 : BR     GROM@>0353
0344 : DST    @>8300,@>8370
0347 : DSUB   @>8300,>0FFF
034B : MOVE   @>8300 TO VDP@>1000 FROM VDP@>0FFF Clear VDP
0353 : CLR     @>8300
0355 : MOVE   >006F TO @>8301 FROM @>8300 CPU RAM >00->6F
035A : MOVE   >003C TO @>8384 FROM @>8300 and >84->C0 clear
0360 : CLR     @>8374          Mode 0
0362 : MOVE   >001F TO VDP@>0381 FROM VDP@>0380 Load color table
0369 : DCLR   @>8382
036C : CZ     @>8378          GROM?
036E : BS     GROM@>0378      Yes, execute program over GPL RTN
0370 : DECT   @>8373          No, stack -2
0372 : DST    @>8300,@>8380    Fetch address in >8300 (for XML)
0376 : XML    >F0             and execute
0378 : RTN

```

Check if GROM on >6000

```

0379 : ST     @>8342,>60
037C : MOVE   >0002 TO @>8328 FROM GROM@>6000
0382 : CEQ    @>8328,>AA      GROM here?
0385 : BR     GROM@>0392
0387 : CGE    @>8329,>00      Number <>0?
038A : BS     GROM@>0392
038C : INCT   @>8373
038E : DST    *>8373,@>8342  on stack
0392 : RTN

```

Load standard character set

```

0393 : MOVE   >0200 TO VDP*>834A FROM GROM@>04B0
039A : RTN

```

Load small capital character set

```

039B : DST    @>83D0,>06B0    Address GROM
03A0 : ST     @>83D2,>40      Number
03A4 : CLR    VDP*>834A      Clear 1st byte in VDP
03A7 : MOVE   >0007 TO VDP@>0001(@>834A) FROM GROM@>0000(@>83D0)
03B0 : DADD   @>834A,>0008    VDP address+8
03B4 : DADD   @>83D0,>0007    GROM address+7
03B9 : DEC    @>83D2
03BC : BR     GROM@>03A4      0?
03BE : RTN

```

Load lower case character set

```

03BF : DST    @>83D0,>0870    GROM address

```

```

03C4 : ST      @>83D2,>1F      Number
03C8 : B      GROM@>03A4      Execute

Accept Tone
03CB : DST    @>8358,>0475      Load tone
03CF : I/O    @>8358,>00      Print
03D2 : RTN

Bad response tone
03D3 : DST    @>8358,>0480      Load tone
03D7 : BR     GROM@>03CF      Execute

GPL DSRLNK:
03D9 : FETC   @>836D          Fetch data
03DB : CLR    @>8354
03DD : ST     @>8355,VDP*>8356 Fetch length byte name
03E1 : CLR    @>8358
03E3 : DST    @>8352,@>8356
03E6 : DINC   @>8352
03E8 : CEQ    @>8358,@>8355 Length = length of name?
03EB : BS     GROM@>03F7
03ED : CEQ    VDP*>8352,>2E Point?
03F1 : BS     GROM@>03F7      Yes, go on
03F3 : INC    @>8358          Length DSR name+1
03F5 : BR     GROM@>03E6      Go on
03F7 : CZ     @>8358          Length 0?
03F9 : BS     GROM@>0435      Yes, end with condition bit
03FB : ST     @>8355,@>8358 Length on >8355
03FE : CGE    @>8355,>08      Longer than 8?
0401 : BS     GROM@>0435      Yes, end with set condition bit
0403 : CLR    @>8354
0405 : DCLR   @>83D0          Clear GROM search pointer
0408 : DINC   @>8356          Beginning of name
040A : MOVE   @>8354 TO @>834A FROM VDP*>8356 Fetch name on FAC
040F : DADD   @>8356,@>8354 Left pointing!
0412 : XML    >19            Execute with following RTN (if found) otherwise
                                go on with GSRLNK

GSRLNK:
0414 : INCT   @>8373          GROM read data on substack
0416 : DST    *>8373,@>83FA
041B : XML    >1A            GSRLNK
041D : BR     GROM@>0429
041F : INCT   @>8373
0421 : DST    *>8373,*>8372 Data stack on substack
0426 : DECT   @>8372
0428 : RTN
0429 : DCZ    @>83D0          GROM search pointer 0?
042C : BR     GROM@>041B

042E : DST    @>83FA,*>8373 GROM read address from substack
0433 : DECT   @>8373
0435 : CEQ    @>8300,@>8300
0438 : RTNC   Return condition bit is set

0439 : DECT   @>8373
043B : DST    @>83FA,*>8373 Fetch R13 GPLWS from substack
0440 : DECT   @>8373
0442 : RTN    Return

0443 : B      GROM@>284C
0446 : B      GROM@>284E
0449 : B      GROM@>2010

VDP register data:
044C : DATA  >80
044D : DATA  >60

```


044E : DATA >20
044F : DATA >F0
0450 : DATA >0E
0451 : DATA >F9
0452 : DATA >86
0453 : DATA >F8
0454 : DATA >F7

Color table title screen

0455 : DATA >17,>17,>17,>17,>17,>17,>17,>17
045D : DATA >17,>17,>17,>17,>06,>03,>01,>0B
0465 : DATA >0C,>0D,>0F,>04,>02,>0D,>08,>0E
046D : DATA >05,>09,>0A,>06,>27,>27,>22,>22

Sound list accept tone

0475 : DATA >06
0476 : DATA >BF
0477 : DATA >DF
0478 : DATA >FF
0479 : DATA >80
047A : DATA >05
047B : DATA >92
047C : DATA >0A
047D : DATA >01
047E : DATA >9F
047F : DATA >00

Sound list bad tone

0480 : DATA >06
0481 : DATA >BF
0482 : DATA >DF
0483 : DATA >FF
0484 : DATA >80
0485 : DATA >20
0486 : DATA >90
0487 : DATA >0A
0488 : DATA >01
0489 : DATA >9F
048A : DATA >00

Text titel screen:

048B : TEXT '>0A,:1981 TEXAS INSTRUMENTSHOME COMPUTER:'

Standard charcter set data:

04B0 : DATA >00,>00,>00,>00,>00,>00,>00,>00
04B8 : DATA >20,>20,>20,>20,>20,>20,>00,>20
04C0 : DATA >48,>48,>48,>00,>00,>00,>00,>00
04C8 : DATA >00,>48,>FC,>48,>48,>FC,>48,>00
04D0 : DATA >10,>3C,>50,>38,>14,>78,>10,>00
04D8 : DATA >C0,>C4,>08,>10,>20,>40,>8C,>0C
04E0 : DATA >60,>90,>90,>60,>60,>94,>88,>74
04E8 : DATA >08,>10,>20,>00,>00,>00,>00,>00
04F0 : DATA >08,>10,>20,>20,>20,>20,>10,>08
04F8 : DATA >40,>20,>10,>10,>10,>10,>20,>40
0500 : DATA >00,>00,>48,>30,>CC,>30,>48,>00
0508 : DATA >00,>00,>10,>10,>7C,>10,>10,>00
0510 : DATA >00,>00,>00,>00,>00,>30,>10,>20
0518 : DATA >00,>00,>00,>00,>7C,>00,>00,>00
0520 : DATA >00,>00,>00,>00,>00,>00,>30,>30
0528 : DATA >00,>04,>08,>10,>20,>40,>80,>00
0530 : DATA >38,>44,>44,>44,>44,>44,>44,>38
0538 : DATA >10,>30,>50,>10,>10,>10,>10,>7C
0540 : DATA >78,>84,>04,>08,>10,>20,>40,>FC
0548 : DATA >78,>84,>04,>38,>04,>04,>84,>78
0550 : DATA >0C,>14,>24,>44,>84,>FC,>04,>04
0558 : DATA >F8,>80,>80,>F8,>04,>04,>84,>78

```

0560 : DATA >78,>80,>80,>F8,>84,>84,>84,>78
0568 : DATA >FC,>04,>04,>08,>10,>20,>40,>40
0570 : DATA >78,>84,>84,>78,>84,>84,>84,>78
0578 : DATA >78,>84,>84,>84,>7C,>04,>04,>78
0580 : DATA >00,>30,>30,>00,>00,>30,>30,>00
0588 : DATA >00,>30,>30,>00,>00,>30,>10,>20
0590 : DATA >00,>08,>10,>20,>40,>20,>10,>08
0598 : DATA >00,>00,>00,>7C,>00,>7C,>00,>00
05A0 : DATA >00,>40,>20,>10,>08,>10,>20,>40
05A8 : DATA >38,>44,>04,>08,>10,>10,>00,>10
05B0 : DATA >00,>78,>84,>9C,>A4,>98,>80,>7C
05B8 : DATA >78,>84,>84,>84,>FC,>84,>84,>84
05C0 : DATA >F8,>44,>44,>78,>44,>44,>44,>F8
05C8 : DATA >78,>84,>80,>80,>80,>80,>84,>78
05D0 : DATA >F8,>44,>44,>44,>44,>44,>44,>F8
05D8 : DATA >FC,>80,>80,>F0,>80,>80,>80,>FC
05E0 : DATA >FC,>80,>80,>F0,>80,>80,>80,>80
05E8 : DATA >78,>84,>80,>80,>9C,>84,>84,>78
05F0 : DATA >84,>84,>84,>FC,>84,>84,>84,>84
05F8 : DATA >7C,>10,>10,>10,>10,>10,>10,>7C
0600 : DATA >04,>04,>04,>04,>04,>84,>84,>78
0608 : DATA >88,>90,>A0,>C0,>A0,>90,>88,>84
0610 : DATA >40,>40,>40,>40,>40,>40,>40,>7C
0618 : DATA >84,>CC,>B4,>84,>84,>84,>84,>84
0620 : DATA >84,>C4,>A4,>94,>8C,>84,>84,>84
0628 : DATA >FC,>84,>84,>84,>84,>84,>84,>FC
0630 : DATA >F8,>84,>84,>84,>F8,>80,>80,>80
0638 : DATA >78,>84,>84,>84,>84,>94,>88,>74
0640 : DATA >F8,>84,>84,>84,>F8,>90,>88,>84
0648 : DATA >78,>84,>80,>78,>04,>04,>84,>78
0650 : DATA >7C,>10,>10,>10,>10,>10,>10,>10
0658 : DATA >84,>84,>84,>84,>84,>84,>84,>78
0660 : DATA >44,>44,>44,>44,>28,>28,>10,>10
0668 : DATA >84,>84,>84,>84,>84,>B4,>CC,>84
0670 : DATA >84,>84,>48,>30,>30,>48,>84,>84
0678 : DATA >44,>44,>44,>28,>10,>10,>10,>10
0680 : DATA >FC,>04,>08,>10,>20,>40,>80,>FC
0688 : DATA >38,>20,>20,>20,>20,>20,>20,>38
0690 : DATA >00,>80,>40,>20,>10,>08,>04,>00
0698 : DATA >70,>10,>10,>10,>10,>10,>10,>70
06A0 : DATA >10,>28,>44,>82,>00,>00,>00,>00
06A8 : DATA >00,>00,>00,>00,>00,>00,>00,>FC

```

Small capital letters set (7 bytes per letter):

```

06B0 : DATA >00,>00,>00,>00,>00,>00,>00,>10
06B8 : DATA >10,>10,>10,>10,>00,>10,>28,>28
06C0 : DATA >28,>00,>00,>00,>00,>28,>28,>7C
06C8 : DATA >28,>7C,>28,>28,>38,>54,>50,>38
06D0 : DATA >14,>54,>38,>60,>64,>08,>10,>20
06D8 : DATA >4C,>0C,>20,>50,>50,>20,>54,>48
06E0 : DATA >34,>08,>08,>10,>00,>00,>00,>00
06E8 : DATA >08,>10,>20,>20,>20,>10,>08,>20
06F0 : DATA >10,>08,>08,>08,>10,>20,>00,>28
06F8 : DATA >10,>7C,>10,>28,>00,>00,>10,>10
0700 : DATA >7C,>10,>10,>00,>00,>00,>00,>00
0708 : DATA >30,>10,>20,>00,>00,>00,>7C,>00
0710 : DATA >00,>00,>00,>00,>00,>00,>00,>30
0718 : DATA >30,>00,>04,>08,>10,>20,>40,>00
0720 : DATA >38,>44,>44,>44,>44,>44,>38,>10
0728 : DATA >30,>10,>10,>10,>10,>38,>38,>44
0730 : DATA >04,>08,>10,>20,>7C,>38,>44,>04
0738 : DATA >18,>04,>44,>38,>08,>18,>28,>48
0740 : DATA >7C,>08,>08,>7C,>40,>78,>04,>04
0748 : DATA >44,>38,>18,>20,>40,>78,>44,>44
0750 : DATA >38,>7C,>04,>08,>10,>20,>20,>20
0758 : DATA >38,>44,>44,>38,>44,>44,>38,>38

```

0760 : DATA >44,>44,>3C,>04,>08,>30,>00,>30
 0768 : DATA >30,>00,>30,>30,>00,>00,>30,>30
 0770 : DATA >00,>30,>10,>20,>08,>10,>20,>40
 0778 : DATA >20,>10,>08,>00,>00,>7C,>00,>7C
 0780 : DATA >00,>00,>20,>10,>08,>04,>08,>10
 0788 : DATA >20,>38,>44,>04,>08,>10,>00,>10
 0790 : DATA >38,>44,>5C,>54,>5C,>40,>38,>38
 0798 : DATA >44,>44,>7C,>44,>44,>44,>78,>24
 07A0 : DATA >24,>38,>24,>24,>78,>38,>44,>40
 07A8 : DATA >40,>40,>44,>38,>78,>24,>24,>24
 07B0 : DATA >24,>24,>78,>7C,>40,>40,>78,>40
 07B8 : DATA >40,>7C,>7C,>40,>40,>78,>40,>40
 07C0 : DATA >40,>3C,>40,>40,>5C,>44,>44,>38
 07C8 : DATA >44,>44,>44,>7C,>44,>44,>44,>38
 07D0 : DATA >10,>10,>10,>10,>10,>38,>04,>04
 07D8 : DATA >04,>04,>04,>44,>38,>44,>48,>50
 07E0 : DATA >60,>50,>48,>44,>40,>40,>40,>40
 07E8 : DATA >40,>40,>7C,>44,>6C,>54,>54,>44
 07F0 : DATA >44,>44,>44,>64,>64,>54,>4C,>4C
 07F8 : DATA >44,>7C,>44,>44,>44,>44,>44,>7C
 0800 : DATA >78,>44,>44,>78,>40,>40,>40,>38
 0808 : DATA >44,>44,>44,>54,>48,>34,>78,>44
 0810 : DATA >44,>78,>50,>48,>44,>38,>44,>40
 0818 : DATA >38,>04,>44,>38,>7C,>10,>10,>10
 0820 : DATA >10,>10,>10,>44,>44,>44,>44,>44
 0828 : DATA >44,>38,>44,>44,>44,>28,>28,>10
 0830 : DATA >10,>44,>44,>44,>54,>54,>54,>28
 0838 : DATA >44,>44,>28,>10,>28,>44,>44,>44
 0840 : DATA >44,>28,>10,>10,>10,>10,>7C,>04
 0848 : DATA >08,>10,>20,>40,>7C,>38,>20,>20
 0850 : DATA >20,>20,>20,>38,>00,>40,>20,>10
 0858 : DATA >08,>04,>00,>38,>08,>08,>08,>08
 0860 : DATA >08,>38,>00,>10,>28,>44,>00,>00
 0868 : DATA >00,>00,>00,>00,>00,>00,>00,>7C

Lower case set:

0870 : DATA >00,>20,>10,>08,>00,>00,>00,>00
 0878 : DATA >00,>38,>44,>7C,>44,>44,>00,>00
 0880 : DATA >78,>24,>38,>24,>78,>00,>00,>3C
 0888 : DATA >40,>40,>40,>3C,>00,>00,>78,>24
 0890 : DATA >24,>24,>78,>00,>00,>7C,>40,>78
 0898 : DATA >40,>7C,>00,>00,>7C,>40,>78,>40
 08A0 : DATA >40,>00,>00,>3C,>40,>5C,>44,>38
 08A8 : DATA >00,>00,>44,>44,>7C,>44,>44,>00
 08B0 : DATA >00,>38,>10,>10,>10,>38,>00,>00
 08B8 : DATA >08,>08,>08,>48,>30,>00,>00,>24
 08C0 : DATA >28,>30,>28,>24,>00,>00,>40,>40
 08C8 : DATA >40,>40,>7C,>00,>00,>44,>6C,>54
 08D0 : DATA >44,>44,>00,>00,>44,>64,>54,>4C
 08D8 : DATA >44,>00,>00,>7C,>44,>44,>44,>7C
 08E0 : DATA >00,>00,>78,>44,>78,>40,>40,>00
 08E8 : DATA >00,>38,>44,>54,>48,>34,>00,>00
 08F0 : DATA >78,>44,>78,>48,>44,>00,>00,>3C
 08F8 : DATA >40,>38,>04,>78,>00,>00,>7C,>10
 0900 : DATA >10,>10,>10,>00,>00,>44,>44,>44
 0908 : DATA >44,>38,>00,>00,>44,>44,>28,>28
 0910 : DATA >10,>00,>00,>44,>44,>54,>54,>28
 0918 : DATA >00,>00,>44,>28,>10,>28,>44,>00
 0920 : DATA >00,>44,>28,>10,>10,>10,>00,>00
 0928 : DATA >7C,>08,>10,>20,>7C,>18,>20,>20
 0930 : DATA >40,>20,>20,>18,>10,>10,>10,>00
 0938 : DATA >10,>10,>10,>30,>08,>08,>04,>08
 0940 : DATA >08,>30,>00,>20,>54,>08,>00,>00
 0948 : DATA >00

0949 : TEXT ':FOR:'

Special characters for TI symbol:

094C : DATA >01,>03,>03,>03,>03,>03,>03
0954 : DATA >FC,>04,>05,>05,>04,>06,>02,>0C
095C : DATA >00,>80,>40,>40,>80,>00,>0C,>12
0964 : DATA >FF,>80,>C0,>40,>60,>38,>1C,>0E
096C : DATA >19,>21,>21,>3D,>05,>05,>05,>C4
0974 : DATA >BA,>8A,>8A,>BA,>A1,>A1,>A1,>22
097C : DATA >03,>01,>00,>00,>00,>00,>00,>00
0984 : DATA >E2,>31,>10,>18,>0C,>07,>03,>00
098C : DATA >4C,>90,>20,>40,>40,>20,>E0,>00

Copyright sign:

0994 : DATA >3C,>42,>99,>A1,>A1,>99,>42,>3C

099C : MOVE >001A TO VDP@>03C0 FROM @>8310 >8310->832A save in VDP
09A2 : DCLR @>8352
09A4 : RTN

Number to string (CNS):

09A5 : CALL GROM@>099C
09A8 : MOVE >000A TO @>831A FROM @>834A
09AD : DST @>8358,>2020
09B1 : CGE @>834A,>00
09B4 : BS GROM@>09BB
09B6 : DNEG @>834A
09B8 : ST @>8359,>2D
09BB : ST @>8316,>5A
09BE : DCZ @>834A
09C0 : BR GROM@>09DE
09C2 : DST *>8316,>3000
09C7 : INC @>8316
09C9 : CGT @>8355,>00
09CC : BR GROM@>0A9B
09CE : ST @>8310,>01
09D1 : CGE @>8357,>00
09D4 : BR GROM@>09D9
09D6 : ADD @>8310,@>8357
09D9 : CALL GROM@>0C42
09DC : BR GROM@>0A9B
09DE : CALL GROM@>0BCF
09E1 : CZ @>8355
09E3 : BR GROM@>0AEB
09E5 : CHE @>8377,>0A
09E8 : BS GROM@>0A00
09EA : ST @>8310,@>834A
09ED : ADD @>8310,>0C
09F0 : CZ *>8310
09F3 : BR GROM@>0A00
09F5 : INC @>8310
09F7 : CGE @>8310,>52
09FA : BR GROM@>09F0
09FC : CLR @>8318
09FE : BR GROM@>0A20
0A00 : ST @>8312,>05
0A03 : CGT @>8377,>09
0A06 : BS GROM@>0A1A
0A08 : CGE @>8377,>FC
0A0B : BR GROM@>0A1A
0A0D : ST @>8312,>09
0A10 : CGT @>8377,>FE
0A13 : BS GROM@>0A1A
0A15 : INC @>8312
0A17 : ADD @>8312,@>8377
0A1A : CALL GROM@>0B6F
0A1D : ST @>8318,>FF
0A20 : CGT @>8377,>09

0A23 : BS GROM@>0A60
0A25 : CGT @>8377,>FA
0A28 : BS GROM@>0A55
0A2A : CGE @>8377,>F6
0A2D : BR GROM@>0A60
0A2F : ST @>8310,>52
0A32 : ST @>8312,@>8314
0A35 : ADD @>8312,>04
0A38 : DECT @>8312
0A3A : DEC @>8310
0A3C : CZ *>8310
0A3F : BS GROM@>0A38
0A41 : ST @>8311,*>8310
0A45 : CLR @>8310
0A47 : DIV @>8310,>0A
0A4A : CZ @>8311
0A4C : BR GROM@>0A50
0A4E : DEC @>8312
0A50 : CGT @>8312,@>8377
0A53 : BS GROM@>0A60
0A55 : ST @>8312,>0C
0A58 : CALL GROM@>0BE8
0A5B : CALL GROM@>0C55
0A5E : BR GROM@>0A9B
0A60 : ST @>8312,>08
0A63 : ST @>8315,>03
0A66 : SUB @>8315,@>8314
0A69 : CALL GROM@>0C01
0A6C : CALL GROM@>0C55
0A6F : DST *>8316,>452B
0A74 : CZ @>8376
0A76 : BS GROM@>0A7F
0A78 : DST *>8316,>452D
0A7D : DABS @>8376
0A7F : INCT @>8316
0A81 : DST *>8316,>2A2A
0A86 : CHE @>8377,>64
0A89 : BS GROM@>0A96
0A8B : DIV @>8376,>0A
0A8E : DADD @>8376,>3030
0A92 : DST *>8316,@>8376
0A96 : INCT @>8316
0A98 : CLR *>8316
0A9B : ST @>8356,>59
0A9E : ST @>8312,@>8359
0AA1 : ST *>8356,>20
0AA5 : INC @>8356
0AA7 : CEQ *>8356,>30
0AAB : BS GROM@>0AA1
0AAD : CZ *>8356
0AB0 : BS GROM@>0AC6
0AB2 : CEQ *>8356,>45
0AB6 : BS GROM@>0AC6
0AB8 : DCEQ *>8356,>2E00
0ABD : BS GROM@>0AC6
0ABF : DCEQ *>8356,>2E45
0AC4 : BR GROM@>0ACF
0AC6 : DECT @>8356
0AC8 : DST *>8356,>2030
0ACD : BR GROM@>0AD5
0ACF : DEC @>8356
0AD1 : ST *>8356,@>8312
0AD5 : ST @>8355,@>8356
0AD8 : INC @>8356
0ADA : CZ *>8356
0ADD : BR GROM@>0AD8

```

0ADF : SUB    @>8356,@>8355
0AE2 : MOVE   >001A TO @>8310 FROM VDP@>03C0
0AE8 : DCLR   @>8352
0AEA : RTNC

0AEB : CGE    @>8377,@>8355
0AEE : BS     GROM@>0B3E
0AF0 : CALL   GROM@>0B2F
0AF3 : CGE    @>8312,>FF
0AF6 : BR     GROM@>0B54
0AF8 : CGE    @>8312,@>8355
0AFB : BR     GROM@>0B02
0AFD : ST     @>8312,@>8355
0B00 : DEC    @>8312
0B02 : CALL   GROM@>0B6F
0B05 : CGE    @>8377,@>8355
0B08 : BS     GROM@>0B3E
0B0A : CALL   GROM@>0B2F
0B0D : ADD    @>8312,>03
0B10 : CGE    @>8312,>03
0B13 : BR     GROM@>0B54
0B15 : ST     @>8310,@>8355
0B18 : INCT   @>8310
0B1A : CGT    @>8312,@>8310
0B1D : BR     GROM@>0B22
0B1F : ST     @>8312,@>8310
0B22 : SUB    @>8312,@>8314
0B25 : CALL   GROM@>0BE8
0B28 : CGE    @>8357,>00
0B2B : BR     GROM@>0A5B
0B2D : BR     GROM@>0A9B
0B2F : ST     @>8312,@>8355
0B32 : CGE    @>8357,>00
0B35 : BR     GROM@>0B3A
0B37 : ST     @>8312,@>8357
0B3A : ADD    @>8312,@>8377
0B3D : RTN

0B3E : CZ     @>8356
0B40 : BR     GROM@>0B58
0B42 : ST     *>8316,>45
0B46 : INC    @>8316
0B48 : DEC    @>8355
0B4A : BR     GROM@>0B42
0B4C : CLR    *>8316
0B4F : ST     @>8356,>59
0B52 : BR     GROM@>0AD5
0B54 : CZ     @>8356
0B56 : BS     GROM@>09C2
0B58 : MOVE   >000A TO @>834A FROM @>831A
0B5D : ST     @>8312,@>8356
0B60 : DEC    @>8312
0B62 : CALL   GROM@>0B6F
0B65 : ST     @>8312,@>8356
0B68 : INCT   @>8312
0B6A : SUB    @>8312,@>8314
0B6D : BR     GROM@>0A63
0B6F : SUB    @>8377,@>8312
0B72 : SUB    @>8312,@>8314
0B75 : SRL    @>8312,>01
0B78 : INCT   @>8312
0B7A : AND    @>8312,>7F
0B7D : CGT    @>8312,>07
0B80 : BS     GROM@>0BCF
0B82 : ST     @>8310,>31
0B85 : CLOG   @>8377,>01

```

0B88	:	BS	GROM@>0B8D	
0B8A	:	ST	@>8310,>04	
0B8D	:	ST	@>8314,>4A	
0B90	:	DCLR	@>8375	
0B92	:	ST	@>8377,@>834A	
0B95	:	ST	@>8318,@>834A	
0B98	:	ADD	@>8314,@>8312	
0B9B	:	ADD	*>8314,@>8310	
0B9F	:	ST	@>8354,@>8312	
0BA2	:	XML	>02	
0BA4	:	CLR	@>8312	
0BA6	:	CEQ	@>8314,>4B	
0BA9	:	BR	GROM@>0BB0	
0BAB	:	CEQ	@>8318,@>834A	
0BAE	:	BR	GROM@>0BC3	
0BB0	:	CEQ	@>8310,>04	
0BB3	:	BR	GROM@>0BC5	
0BB5	:	ST	@>8313,*>8314	
0BB9	:	DIV	@>8312,>0A	
0BBC	:	MUL	@>8312,>0A	
0BBF	:	ST	*>8314,@>8313	
0BC3	:	INC	@>8314	
0BC5	:	CLR	*>8314	
0BC8	:	INC	@>8314	
0BCA	:	CGE	@>8314,>52	
0BCD	:	BR	GROM@>0BC5	
0BCF	:	ST	@>8376,@>834A	Exponent
0BD2	:	SUB	@>8376,>40	*2 (100!)
0BD5	:	SLL	@>8376,>01	
0BD8	:	DSRA	@>8376,>0008	
0BDC	:	CLR	@>8314	
0BDE	:	CGE	@>834B,>0A	10?
0BE1	:	BR	GROM@>0BE7	
0BE3	:	INC	@>8377	Exponent +1
0BE5	:	INC	@>8314	Set flag
0BE7	:	RTN		
0BE8	:	SUB	@>8377,@>8314	
0BEB	:	ST	@>8315,@>8377	
0BEE	:	ADD	@>8315,>03	
0BF1	:	CGE	@>8377,>00	
0BF4	:	BS	GROM@>0C01	
0BF6	:	ST	@>8310,>FF	
0BF9	:	SUB	@>8310,@>8377	
0BFC	:	CALL	GROM@>0C42	
0BFF	:	CLR	@>8315	
0C01	:	DCLR	@>8352	
0C03	:	ST	@>8313,>4B	
0C06	:	CALL	GROM@>0C2F	
0C09	:	CLR	@>8310	
0C0B	:	ST	@>8311,*>8313	
0C0F	:	INC	@>8313	
0C11	:	DIV	@>8310,>0A	
0C14	:	DADD	@>8310,>3030	
0C18	:	CALL	GROM@>0C29	
0C1B	:	ST	@>8310,@>8311	
0C1E	:	CALL	GROM@>0C29	
0C21	:	BR	GROM@>0C09	
0C23	:	ST	@>8310,>2E	
0C26	:	ST	@>8317,@>8316	
0C29	:	ST	*>8316,@>8310	
0C2D	:	INC	@>8316	
0C2F	:	DEC	@>8315	
0C31	:	BS	GROM@>0C23	
0C33	:	CEQ	@>8316,>69	

```

0C36 : BS      GROM@>0C3C
0C38 : DEC      @>8312
0C3A : BR      GROM@>0C41
0C3C : DECT     @>8373
0C3E : CLR      *>8316
0C41 : RTN

0C42 : ST      *>8316,>2E
0C46 : INC      @>8316
0C48 : DEC      @>8310
0C4A : CGT      @>8310,>00
0C4D : BR      GROM@>0C3E
0C4F : ST      *>8316,>30
0C53 : BR      GROM@>0C46
0C55 : DEC      @>8316
0C57 : CEQ      *>8316,>30
0C5B : BS      GROM@>0C55
0C5D : INC      @>8316
0C5F : CZ      @>8318
0C61 : BR      GROM@>0C3E
0C63 : ST      @>8316,@>8317
0C66 : BR      GROM@>0C3E

0C68 : DADD     @>836E,>0008   Increase value stack pointer
0C6C : MOVE     >0008 TO VDP*>836E FROM @>834A   FAC on value stack
0C72 : RTN

0C73 : MOVE     >0008 TO @>834A FROM VDP*>836E   Fetch stack entry on FAC
0C79 : DADD     @>836E,>FFF8   New value stack pointer
0C7D : RTN

```

Involution routine:

```

0C7E : CALL     GROM@>11F2
0C81 : DCZ      @>834A
0C83 : BS      GROM@>0D3D
0C85 : DCZ      @>835C
0C87 : BS      GROM@>0D35
0C89 : DADD     @>836E,>0008
0C8D : CLR      @>8375
0C8F : CALL     GROM@>0C68
0C92 : CALL     GROM@>11FA
0C95 : CLR      @>8375
0C97 : CALL     GROM@>11DC
0C9A : XML      >0F
0C9C : BR      GROM@>0D47
0C9E : CALL     GROM@>0C68
0CA1 : CLR      @>8354
0CA3 : XML      >12
0CA5 : DABS     @>834A
0CA7 : DST      VDP@>03DB,@>834A
0CAB : CALL     GROM@>0C73
0CAE : CZ      @>8354
0CB0 : BR      GROM@>0CFE
0CB2 : CALL     GROM@>11DC
0CB5 : CALL     GROM@>0C68
0CB8 : DDEC     VDP@>03DB
0CBB : BS      GROM@>0CE0
0CBD : CLOG     VDP@>03DC,>01
0CC1 : BS      GROM@>0CC9
0CC3 : XML      >0D
0CC5 : DADD     @>836E,>0008
0CC9 : DSRL     VDP@>03DB,>0001
0CCE : DCZ      VDP@>03DB
0CD1 : BS      GROM@>0CE0
0CD3 : CALL     GROM@>11DC
0CD6 : CALL     GROM@>0C68

```



```

0CD9 : XML      >0D
0CDB : CALL     GROM@>11DC
0CDE : BR       GROM@>0CBD
0CE0 : DADD     @>836E,>FFF0
0CE4 : CGE     VDP@>0008(@>836E),>00
0CE9 : BS      GROM@>0CFD
0CEB : CZ      @>8354
0CED : BR      GROM@>0CF9
0CEF : MOVE    >0008 TO @>835C FROM GROM@>106D
0CF5 : XML     >09
0CF7 : BR      GROM@>0CFD
0CF9 : DCLR    @>834A
0CFB : CLR     @>8354
0CFD : RTN

0CFE : CGE     VDP*>836E,>00
0D02 : BS      GROM@>0D1B
0D04 : ST      @>8356,@>834A
0D07 : ABS     @>8356
0D09 : CGT     @>8356,>46
0D0C : BS      GROM@>0D1B
0D0E : ST      @>8375,@>8356
0D11 : ADD     @>8375,>0B
0D14 : ST      @>8375,*>8375
0D18 : SRC     @>8375,>01
0D1B : ST      VDP@>03DC,@>8375
0D1F : CALL    GROM@>11DC
0D22 : DABS    @>834A
0D24 : CALL    GROM@>0E60
0D27 : XML     >0D
0D29 : CALL    GROM@>0DB0
0D2C : CGE     VDP@>03DC,>00
0D30 : BS      GROM@>0D34
0D32 : DNEG    @>834A
0D34 : RTN

0D35 : CGE     @>834A,>00      FAC 0?
0D38 : BR      GROM@>0D44      No, jump
0D3A : DCLR    @>834A          Set FAC and FAC+1
0D3C : RTN

0D3D : MOVE    >0008 TO @>834A FROM GROM@>1041  Number 1 on FAC
0D43 : RTN

0D44 : XML     >05              Set overflow
0D46 : RTN

0D47 : CGE     VDP*>836E,>00
0D4B : BS      GROM@>0D1B
0D4D : DADD     @>836E,>FFF8
0D51 : ST      @>8354,>05
0D54 : RTN

Square root:
0D55 : CALL    GROM@>099C
0D58 : XML     >03
0D5A : BS      GROM@>0AE2
0D5C : GT
0D5D : BR      GROM@>0DAB
0D5F : ST      @>8314,@>834A
0D62 : ST      @>834A,>3F
0D65 : ADD     @>8314,>C1
0D68 : DSRA    @>8314,>0008
0D6C : DSLR    @>8314,>0001
0D70 : CALL    GROM@>0C68
0D73 : CALL    GROM@>0C68

```

```

0D76 : DST    @>8312,>1027
0D7A : CALL   GROM@>11A9
0D7D : CALL   GROM@>11DC
0D80 : DST    @>8312,>1041
0D84 : CALL   GROM@>11A9
0D87 : XML    >0E
0D89 : ST     @>8312,>03
0D8C : MOVE   >0008 TO @>835C FROM VDP*>836E
0D92 : CALL   GROM@>0C68
0D95 : XML    >09
0D97 : XML    >0B
0D99 : MOVE   >0008 TO @>835C FROM GROM@>0FDF
0D9F : XML    >08
0DA1 : DEC    @>8312
0DA3 : BR     GROM@>0D8C
0DA5 : DADD   @>836E,>FFF8
0DA9 : BR     GROM@>0E39
0DAB : ST     @>8354,>04
0DAE : BR     GROM@>0AE2

```

Exponent :

```

0DB0 : CALL   GROM@>099C
0DB3 : MOVE   >0008 TO @>835C FROM GROM@>0FEF
0DB9 : XML    >08
0DBB : CALL   GROM@>0C68
0DBE : CALL   GROM@>11FA
0DC1 : MOVE   >0008 TO @>835C FROM GROM@>0FD7
0DC7 : XML    >0A
0DC9 : BS     GROM@>0DE3
0DCB : GT
0DCC : BR     GROM@>0DD5
0DCE : DNEG   @>835C
0DD0 : XML    >0A
0DD2 : GT
0DD3 : BR     GROM@>0DE3
0DD5 : DADD   @>836E,>FFF8
0DD9 : DST    @>8376,@>834A
0DDC : CLR    @>8375
0DDE : XML    >04
0DE0 : B      GROM@>0AE2
0DE3 : CALL   GROM@>0C68
0DE6 : XML    >12
0DE8 : DST    @>8314,@>834A
0DEB : CALL   GROM@>0C73
0DEE : DSL    @>8314,>0001
0DF2 : XML    >0C
0DF4 : MOVE   >0008 TO @>835C FROM GROM@>0FDF
0DFA : XML    >0A
0DFC : GT
0DFD : BS     GROM@>0E05
0DFE : DNEG   @>835C
0E01 : XML    >06
0E03 : DINC   @>8314
0E05 : CALL   GROM@>0C68
0E08 : DST    @>8312,>1053
0E0C : CALL   GROM@>1199
0E0F : CALL   GROM@>11DC
0E12 : DST    @>8312,>106D
0E16 : CALL   GROM@>11A2
0E19 : CALL   GROM@>11F2
0E1C : DADD   @>836E,>0008
0E20 : CALL   GROM@>0C68
0E23 : XML    >06
0E25 : MOVE   >0008 TO @>831A FROM @>834A
0E2A : CALL   GROM@>0C73
0E2D : CALL   GROM@>11DC

```

0E30 : XML >0C
 0E32 : MOVE >0008 TO @>835C FROM @>831A
 0E37 : XML >09
 0E39 : CLOG @>8315,>01
 0E3C : BS GROM@>0E46
 0E3E : MOVE >0008 TO @>835C FROM GROM@>0FE7
 0E44 : XML >08
 0E46 : MOVE >0008 TO @>835C FROM GROM@>1041
 0E4C : CLOG @>8315,>02
 0E4F : BS GROM@>0E54
 0E51 : ST @>835D,>0A
 0E54 : DSRA @>8314,>0002
 0E58 : ADD @>835C,@>8315
 0E5B : XML >08
 0E5D : B GROM@>0AE2

LOG:
 0E60 : XML >03
 0E62 : BS GROM@>0EF1
 0E64 : GT
 0E65 : BR GROM@>0EF1
 0E67 : CALL GROM@>099C
 0E6A : CALL GROM@>0BCF
 0E6D : CZ @>8314
 0E6F : BR GROM@>0E81
 0E71 : MOVE >0008 TO @>835C FROM GROM@>1041
 0E77 : ST @>835D,>0A
 0E7A : XML >08
 0E7C : CALL GROM@>0BCF
 0E7F : DDEC @>8376
 0E81 : DINC @>8376
 0E83 : ST @>834A,>3F
 0E86 : DST VDP@>03DA,@>8376
 0E8A : MOVE >0008 TO @>835C FROM GROM@>0FE7
 0E90 : XML >08
 0E92 : CALL GROM@>1177
 0E95 : CALL GROM@>0C68
 0E98 : DST @>8312,>108F
 0E9C : CALL GROM@>1199
 0E9F : CALL GROM@>11DC
 0EA2 : DST @>8312,>10B9
 0EA6 : CALL GROM@>11A2
 0EA9 : XML >0E
 0EAB : CALL GROM@>0C68
 0EAE : DST @>835C,VDP@>03DA
 0EB2 : MOVE >0006 TO @>835E FROM GROM@>0FE1
 0EB8 : DCZ @>835C
 0EBA : BS GROM@>0ECE
 0EBC : DABS @>835C
 0EBE : DCGT @>835C,>0063
 0EC2 : BS GROM@>0EE3
 0EC4 : ST @>835C,>40
 0EC7 : CZ VDP@>03DA
 0ECA : BS GROM@>0ECE
 0ECC : DNEG @>835C
 0ECE : MOVE >0008 TO @>834A FROM GROM@>0FDF
 0ED4 : XML >07
 0ED6 : MOVE >0008 TO @>835C FROM GROM@>0FF7
 0EDC : XML >08
 0EDE : XML >0B
 0EE0 : B GROM@>0AE2
 0EE3 : DSUB @>835C,>0064
 0EE7 : ST @>835E,@>835D
 0EEA : DST @>835C,>4101
 0EEE : B GROM@>0EC7
 0EF1 : ST @>8354,>06

```

0EF4 : RTN

COS:
0EF5 : MOVE >0008 TO @>835C FROM GROM@>0FFF
0EFB : XML >06

SIN:
0EFD : CALL GROM@>099C
0F00 : MOVE >0008 TO @>835C FROM GROM@>1007
0F06 : XML >08
0F08 : ST VDP@>03DA,@>834A
0F0C : DABS @>834A
0F0E : CGT @>834A,>44
0F11 : BR GROM@>0F18
0F13 : ST @>8354,>07
0F16 : BR GROM@>0AE2
0F18 : CALL GROM@>0C68
0F1B : CALL GROM@>11FA
0F1E : CLR @>8316
0F20 : DCZ @>834A
0F22 : BS GROM@>0F39
0F24 : ST @>8377,@>834A
0F27 : SUB @>8377,>46
0F2A : CGT @>8377,>00
0F2D : BS GROM@>0F36
0F2F : ADD @>8377,>51
0F32 : ST @>8316,*>8377
0F36 : AND @>8316,>03
0F39 : XML >0C
0F3B : CLOG @>8316,>01
0F3E : BS GROM@>0F48
0F40 : MOVE >0008 TO @>835C FROM GROM@>1041
0F46 : XML >07
0F48 : SRL @>8316,>01
0F4B : CZ @>8316
0F4D : BS GROM@>0F52
0F4F : INV VDP@>03DA
0F52 : DST @>8312,>10E3
0F56 : CALL GROM@>1199
0F59 : BR GROM@>0FCD

TAN:
0F5B : CALL GROM@>0C68
0F5E : CALL GROM@>0EFD
0F61 : CALL GROM@>11DC
0F64 : CALL GROM@>0EF5
0F67 : CALL GROM@>11F2
0F6A : CEQ @>8354,>07
0F6D : BS GROM@>0F75
0F6F : XML >03
0F71 : BS GROM@>0F76
0F73 : XML >09
0F75 : RTN

0F76 : ST @>8375,@>835C
0F79 : XML >05
0F7B : RTN

ATN:
0F7C : CALL GROM@>099C
0F7F : ST VDP@>03DA,@>834A
0F83 : DABS @>834A
0F85 : DCLR @>8316
0F87 : MOVE >0008 TO @>835C FROM GROM@>1017
0F8D : XML >0A
0F8F : BS GROM@>0FB9

```

```

0F91 : H
0F92 : BS      GROM@>0FB9
0F94 : MOVE    >0008 TO @>835C FROM GROM@>101F
0F9A : XML     >0A
0F9C : H
0F9D : BS      GROM@>0FB2
0F9F : MOVE    >0008 TO @>835C FROM GROM@>1041
0FA5 : DST     @>835C,>BFFF
0FA9 : XML     >09
0FAB : DST     @>8316,>0FFF
0FAF : B       GROM@>0FB9
0FB2 : CALL    GROM@>1177
0FB5 : DST     @>8316,>100F
0FB9 : DST     @>8312,>1125
0FBD : CALL    GROM@>1199
0FC0 : DCZ     @>8316
0FC2 : BS      GROM@>0FCD
0FC4 : MOVE    >0008 TO @>835C FROM GROM@>0000(@>8316)
0FCB : XML     >06
0FCD : CGE     VDP@>03DA,>00
0FD1 : BS      GROM@>0AE2
0FD3 : DNEG    @>834A
0FD5 : BR      GROM@>0AE2

```

Various constants :

```

0FD7 : DATA  >41,>01,>1B,>00,>00,>00,>00,>00
0FDF : DATA  >3F,>32,>00,>00,>00,>00,>00,>00
0FE7 : DATA  >40,>03,>10,>16,>4D,>42,>01,>45
0FEF : DATA  >3F,>2B,>2A,>5E,>30,>13,>03,>19
0FF7 : DATA  >40,>02,>1E,>19,>55,>09,>1D,>5E
0FFF : DATA  >40,>01,>39,>07,>60,>20,>43,>5F
1007 : DATA  >3F,>3F,>42,>13,>4D,>17,>43,>3A
100F : DATA  >3F,>4E,>35,>62,>10,>21,>61,>2D
1017 : DATA  >3F,>29,>2A,>0D,>38,>17,>49,>0A
101F : DATA  >40,>02,>29,>2A,>0D,>38,>17,>49
1027 : DATA  >3F,>3A,>51,>16,>5A,>00,>00,>00
102F : DATA  >3F,>34,>43,>57,>32,>00,>00,>00
1037 : DATA  >3E,>3A,>51,>14,>00,>00,>00,>00
103F : DATA  >80,>00
1041 : DATA  >40,>01,>00,>00,>00,>00,>00,>00
1049 : DATA  >3F,>09,>63,>63,>50,>00,>00,>00
1051 : DATA  >80,>00,>40,>12,>1F,>17,>3C,>0F
1059 : DATA  >5C,>4B
105B : DATA  >41,>08,>1F,>28,>43,>15,>1D,>25
1063 : DATA  >41,>33,>4E,>09,>13,>5B,>33,>3E
106B : DATA  >80,>00
106D : DATA  >40,>01,>00,>00,>00,>00,>00,>00
1075 : DATA  >41,>01,>3B,>25,>29,>34,>24,>03
107D : DATA  >41,>1B,>09,>1F,>45,>28,>55,>10
1085 : DATA  >41,>2C,>61,>3F,>23,>39,>28,>3A
108D : DATA  >80,>00,>3F,>23,>43,>05,>0A,>1E
1095 : DATA  >58,>2C,>BF,>F5,>62,>1E,>21,>1F
109D : DATA  >24,>58,>40,>3F,>4D,>36,>52,>1C
10A5 : DATA  >56,>11,>BE,>FF,>08,>53,>47,>16
10AD : DATA  >23,>3A,>40,>39,>5E,>49,>51,>26
10B5 : DATA  >2C,>2C,>80,>00,>40,>01,>00,>00
10BD : DATA  >00,>00,>00,>00,>BF,>F3,>0D,>19
10C5 : DATA  >61,>48,>58,>2E,>40,>2F,>2D,>12
10CD : DATA  >16,>24,>02,>3D,>BF,>C0,>07,>40
10D5 : DATA  >3A,>07,>34,>38,>40,>1C,>61,>24
10DD : DATA  >5A,>45,>16,>16,>80,>00,>C4,>FA
10E5 : DATA  >2C,>49,>10,>00,>00,>00,>3C,>05
10ED : DATA  >44,>52,>03,>21,>1A,>58,>C2,>FD
10F5 : DATA  >3B,>58,>09,>0B,>46,>1F,>3E,>01
10FD : DATA  >3C,>2C,>0B,>44,>2E,>62,>C1,>D2
1105 : DATA  >51,>4B,>29,>1F,>06,>02,>3F,>07

```

```

110D : DATA >60,>5C,>3E,>3E,>2D,>3E,>C0,>C0
1115 : DATA >3B,>40,>09,>4B,>06,>16,>40,>01
111D : DATA >39,>07,>60,>20,>43,>5F,>80,>00
1125 : DATA >C0,>FE,>35,>39,>12,>4F,>58,>14
112D : DATA >3F,>05,>02,>4F,>0D,>54,>26,>55
1135 : DATA >C0,>FA,>32,>45,>63,>5E,>01,>28
113D : DATA >3F,>07,>43,>25,>0C,>2B,>5B,>40
1145 : DATA >C0,>F7,>08,>5F,>2F,>5B,>60,>48
114D : DATA >3F,>0B,>0B,>0A,>31,>5C,>32,>35
1155 : DATA >C0,>F2,>1C,>39,>0C,>45,>4B,>60
115D : DATA >3F,>13,>63,>63,>63,>61,>59,>60
1165 : DATA >C0,>DF,>21,>21,>21,>21,>20,>19
116D : DATA >40,>01,>00,>00,>00,>00,>00,>00
1175 : DATA >80,>00

1177 : CALL GROM@>0C68 FAC on VDP stack
117A : CALL GROM@>118E Add 0,5
117D : CALL GROM@>118E Add 0,5
1180 : CALL GROM@>11DC Last value from stack and new value on stack
1183 : MOVE >0008 TO @>835C FROM GROM@>1041 Number 1
1189 : XML >06 FADD
118B : XML >0E SDIV
118D : RTN

118E : MOVE >0008 TO @>835C FROM GROM@>0FDF Number 0.5 on ARG
1194 : DNEG @>835C -
1196 : XML >06 FADD
1198 : RTN

1199 : CALL GROM@>0C68 FAC on value stack
119C : CALL GROM@>11A2
119F : XML >0D SMULT
11A1 : RTN

11A2 : MOVE >0008 TO @>835C FROM @>834A
11A7 : XML >08
11A9 : CALL GROM@>0C68
11AC : MOVE >0008 TO @>834A FROM GROM@>0000(@>8312)
11B3 : BR GROM@>11C6
11B5 : MOVE >0008 TO @>835C FROM VDP*>836E
11BB : XML >08
11BD : MOVE >0008 TO @>835C FROM GROM@>0000(@>8312)
11C4 : XML >06
11C6 : DADD @>8312,>0008
11CA : MOVE >0002 TO @>835C FROM GROM@>0000(@>8312)
11D1 : DCEQ @>835C,>8000
11D5 : BR GROM@>11B5
11D7 : DADD @>836E,>FFF8
11DB : RTN

11DC : CALL GROM@>0C68 FAC on VDP stack
11DF : DADD @>836E,>FFF8
11E3 : MOVE >0008 TO @>834A FROM VDP*>836E Number before last on FAC
11E9 : MOVE >0008 TO VDP*>836E FROM VDP@>0008(@>836E) Last number on stack
11F1 : RTN

11F2 : MOVE >0008 TO @>835C FROM VDP*>836E
11F8 : BR GROM@>11D7

Greatest integer:
11FA : ST @>8375,@>834A
11FD : DABS @>834A
11FF : ST @>8377,@>834A
1202 : CLR @>8376

```

```

1204 : CGE    @>8377,>40
1207 : BR     GROM@>1247
1209 : CGT    @>8377,>45
120C : BS     GROM@>123D
120E : ST     @>8355,@>8377
1211 : SUB    @>8355,>46
1214 : ST     @>8354,@>8355
1217 : CLR    @>8356
1219 : ST     @>8357,>52
121C : ADD    @>8357,@>8355
121F : OR     @>8356,*>8357
1223 : CLR    *>8357
1226 : INC    @>8357
1228 : INC    @>8355
122A : BR     GROM@>121F
122C : CGE    @>8375,>00
122F : BS     GROM@>123D
1231 : CZ     @>8356
1233 : BS     GROM@>123D
1235 : ADD    @>8354,>07
1238 : XML    >02
123A : B      GROM@>1244
123D : CGE    @>8375,>00
1240 : BS     GROM@>1244
1242 : DNEG   @>834A
1244 : CLR    @>8354
1246 : RTN

```

```

1247 : DCLR   @>834A
1249 : CGE    @>8375,>00
124C : BS     GROM@>1252
124E : DST    @>834A,>BFFF
1252 : DCLR   @>834C
1254 : DCLR   @>834E
1256 : DCLR   @>8350
1258 : BR     GROM@>1244

```

Bit reversal routine:

```

125A : MOVE   >0040 TO @>8300 FROM GROM@>1263   Execute program from GROM
1260 : XML    >F0
1262 : RTN

```

Program for bit reversal routine:

```

1263 : DATA   >83,>02,>C0,>60,>83,>4A,>C0,>A0   >8302, MOV @>834A,6, MOV @>834C,1
126B : DATA   >83,>4C,>D7,>E0,>83,>E3,>D7,>C1   MOVB @>83E3,*15, MOVB 1,*15
1273 : DATA   >02,>61,>40,>00,>D0,>EF,>FB,>FE   ORI 1,>4000, MOVB @>FBFE(15),3
127B : DATA   >02,>05,>00,>08,>09,>14,>0A,>13   LI 5,>0008, SRL 4,1, SLA 3,1
1283 : DATA   >17,>02,>02,>24,>80,>00,>06,>05   JNC      , AI 4,>8000, DEC 5
128B : DATA   >16,>F9,>D7,>E0,>83,>E3,>D7,>C1   JNE, MOVB @>83E3,*15, MOVB 1,*1 5
1293 : DATA   >02,>41,>3F,>FF,>DB,>C4,>FF,>FE   ANDI 1,>3FFF, MOVB 4,@>FFFE(15)
129B : DATA   >05,>81,>06,>02,>16,>E6,>04,>5B   INC 1, DEC 2, JNE      , B *11

```

```

12A3 : DATA   >01
12A4 : DATA   >B1

```

```

12A5 : DATA   >15
12A6 : TEXT    ':REVIEW MODULE LIBRARY:'

```

```

12BB : DATA   >00   Empty space till
130F : DATA   >00

```

DSR table :

```

1310 : DATA   >1318           Next DSR
1312 : DATA   >1326           Entry point
1314 : DATA   >03
1315 : TEXT    ':CS1:'

```

1318 : DATA >0000
131A : DATA >132C
131C : DATA >03
131D : TEXT ':CS2:'

Subprogram table:

1320 : DATA >0000
1322 : DATA >1573
1324 : DATA >01
1325 : TEXT '>03' Subprogram writes text

Cassette DSR:

1326 : DST @>8366,>0016 CS1
132A : BR GROM@>1330
132C : DST @>8366,>0017 CS2
1330 : ST @>835A,@>8373 >835A Substack
1333 : DSUB @>8356,@>8354 Compute address length byte (FRAGE)
1336 : AND VDP@>FFF7(@>8356),>1F Clear PAB error
133C : MOVE >000A TO @>834A FROM VDP@>FFF6(@>8356) PAB on FAC
1344 : DST @>835E,@>834C Buffer address
1347 : CASE @>834A Test on I/O op code
1349 : BR GROM@>1387 Open
134B : BR GROM@>140E Close
134D : BR GROM@>13CF Read
134F : BR GROM@>13DA Write
1351 : BR GROM@>1387 Restore
1353 : BR GROM@>13F2 Load
1355 : BR GROM@>1489 Save
1357 : BR GROM@>140E Delete
1359 : BR GROM@>135D Scratch record
135B : BR GROM@>1380 Status

Scratch record

135D : OR VDP@>FFF7(@>8356),>60 Error illegal operation
1363 : CALL GROM@>1549 CRU reset
1366 : CALL GROM@>1516 Scroll
1369 : CLR @>837F
136B : ST @>8373,@>835A Old substack
136E : CALL GROM@>0012 Return over GSR return

1371 : OR VDP@>FFF7(@>8356),>C0 Device error
1377 : CALL GROM@>14A9 Write text
137A : DATA >04 "Press Cassette Stop"
137B : CALL GROM@>1528 Scan keyboard
137E : BR GROM@>1363 End

Status

1380 : CLR VDP@>FFFC(@>8356) Status=>00
1385 : BR GROM@>1363 End

Open, Restore

1387 : CZ @>834E Logical record length
1389 : BR GROM@>1391
138B : ST VDP@>FFFA(@>8356),>40 Decimal 64 default value
1391 : ADD VDP@>FFFA(@>8356),>3F
1397 : AND VDP@>FFFA(@>8356),>C0 Round to integer number 64
139D : CLOG @>834B,>15 Check flag input
13A0 : BS GROM@>13B5
13A2 : CEQ @>8367,>17 CS2?
13A5 : BS GROM@>135D Illegal operation
13A7 : CLOG @>834B,>13 Variable input
13AA : BR GROM@>135D Illegal operation
13AC : CALL GROM@>1503 Turn on motor, rewind cassette
13AF : CALL GROM@>14A9 Press cassette play
13B2 : DATA >0E
13B3 : BR GROM@>13C1

13B5 : CLOG @>834B,>02 Output
 13B8 : BS GROM@>135D Illegal operation
 13BA : CALL GROM@>1503 Turn on motor, rewind
 13BD : CALL GROM@>14A9 Text record
 13C0 : DATA >0C
 13C1 : CALL GROM@>1528 Keyboard scanning
 13C4 : CALL GROM@>1549 Turn on motor
 13C7 : CALL GROM@>1562 Set pointer, wait
 13CA : CALL GROM@>155E Turn off motor
 13CD : BR GROM@>1366 End

Read

13CF : ST @>8362,>05 Read
 13D2 : ST VDP@>FFFB(@>8356),@>834E Record length
 13D8 : BR GROM@>13DD Execute

Write

13DA : ST @>8362,>04 Write
 13DD : CLR @>835C
 13DF : ST @>835D,VDP@>FFFA(@>8356)
 13E5 : CALL GROM@>1549 Set CRU, turn on motor
 13E8 : I/O @>835C,@>8362 Output
 13EB : BS GROM@>1371 Error
 13ED : CALL GROM@>155E Turn off motor
 13F0 : BR GROM@>136E End

Load:

13F2 : CEQ @>8367,>17 CS2?
 13F5 : BS GROM@>135D Illegal operation
 13F7 : CALL GROM@>1503 Turn on motor, rewind text with key
 13FA : CALL GROM@>149F Text play with tone
 13FD : DATA >00 Reading
 13FE : CZ @>83CE Number of sound byte
 1401 : BR GROM@>13FE
 1403 : I/O @>835C,>05 Read data block
 1406 : BS GROM@>1455 Error
 1408 : CALL GROM@>14A9 Text data o.k
 140B : DATA >18
 140C : CLR @>8353

Close, Delete

140E : CALL GROM@>155E Turn on motor
 1411 : CALL GROM@>14A9 Text stop
 1414 : DATA >04
 1415 : CALL GROM@>1528 Keyboard
 1418 : CZ @>8353 Load flag
 141A : BS GROM@>1363 End
 141C : CEQ VDP@>FFF6(@>8356),>06 Save?
 1422 : BR GROM@>1363 End
 1424 : CEQ @>8367,>17 CS2?
 1427 : BS GROM@>1363 End
 1429 : CALL GROM@>14A9 Text check tape
 142C : DATA >10
 142D : SCAN
 142E : BR GROM@>142D No key
 1430 : DST @>837E,>171B YPT, XPT
 1434 : ST @>837D,@>8375 Write key value
 1437 : ADD @>837D,@>8352 Add offset
 143A : CEQ @>8375,>4E N
 143D : BS GROM@>1363 End
 143F : CEQ @>8375,>59 Y
 1442 : BR GROM@>142D None of both keys
 1444 : CALL GROM@>1503 Turn on motor, rewind
 1447 : CALL GROM@>149F Text checking
 144A : DATA >08
 144B : CZ @>83CE Wait for time

```

144E : BR      GROM@>144B
1450 : I/O     @>835C,>06      Check
1453 : BR      GROM@>1408      End with data o.k
1455 : CLOG    @>837C,>01      GPL status byte
1458 : BS      GROM@>1463
145A : CALL    GROM@>155E      Turn off motor
145D : CALL    GROM@>14A9      Text error
1460 : DATA   >02
1461 : BR      GROM@>146A
1463 : CALL    GROM@>155E      Turn off motor
1466 : CALL    GROM@>14A9      Text no data
1469 : DATA   >1A
146A : CEQ     VDP@>FFF6(@>8356),>06 Save?
1470 : BS      GROM@>1478
1472 : CALL    GROM@>14FE      Press R to read
1475 : DATA   >14
1476 : BR      GROM@>147C
1478 : CALL    GROM@>14FE      Press R to record
147B : DATA   >1E
147C : CALL    GROM@>14FE      C to check
147F : DATA   >1C
1480 : CALL    GROM@>14FE      E to exit
1483 : DATA   >16
1484 : CALL    GROM@>1528      Scan keyboard
1487 : BR      GROM@>133C      From beginning

Save
1489 : CALL    GROM@>1503      Turn on motor, rewind
148C : CALL    GROM@>1499      Text press record
148F : DATA   >06            Recording
1490 : CALL    GROM@>1562      Wait and set pointer
1493 : I/O     @>835C,>04      Save
1496 : B       GROM@>140E      End with scanning

1499 : CALL    GROM@>14A9      Text press record
149C : DATA   >0C
149D : BR      GROM@>14A3      Go on
149F : CALL    GROM@>14A9      Text play
14A2 : DATA   >0E
14A3 : CALL    GROM@>1528      Scan keyboard
14A6 : CALL    GROM@>1549      Turn on keyboard

14A9 : MOVE    >02C0 TO VDP@>0000 FROM VDP@>0040  Scroll 2 lines
14B0 : FMT
14B1 : ... YPT=>16
14B3 : ... XPT=>00
14B5 : ... BIAS=(@>8352)
14B7 : ... 20': : '
14B9 : ... ': *:'
14BD : ... 1D': : '
14BF : ... END FMT
14C0 : CLR     @>8362
14C2 : FETC    @>8363      Fetch data for text
14C4 : DST     @>8364,>02E4  Address VDP
14C8 : ST      @>834A,>03    Subprogram name
14CB : DST     @>8354,>0001  Length name subprogram
14CF : ST      @>836D,>0A    Data GSRLNK
14D2 : DCLR    @>83D0      GROM search pointer 0
14D5 : CALL    GROM@>003D    GSRLNK (write text)
14D8 : CZ      @>8375      Keyboard input nessesary?
14DA : BS      GROM@>1513    No, end with tone
14DC : DST     @>837E,>171B  Set YPT and XPT
14E0 : FMT
14E1 : ... BIAS=(@>8352)
14E3 : ... ':CS1:'

```

```

14E7 : ... END FMT
14E8 : CEQ    @>8367,>16      CS1?
14EB : BS     GROM@>14F0      Yes, jump
14ED : INC    VDP@>02FD      Write 2
14F0 : CEQ    @>8375,>FE      Flag >FE (R to read)
14F3 : BS     GROM@>1513      End
14F5 : DST    @>8362,>0012    Then press enter
14F9 : CALL   GROM@>1516      Scroll
14FC : BR     GROM@>14C4      Write text
14FE : CALL   GROM@>1516      Scroll
1501 : BR     GROM@>14C0      Write text
1503 : CALL   GROM@>1549      Turn on motor
1506 : CALL   GROM@>14A9      Write text
1509 : DATA  >0A           Rewind
150A : CALL   GROM@>1528      Keyboard scanning
150D : CALL   GROM@>155E      Turn off motor
1510 : DST    @>835C,@>8350
1513 : B      GROM@>0034      Accept tone (Trick with RTN!)

1516 : MOVE   >02E0 TO VDP@>0000 FROM VDP@>0020  Scroll one line
151D : FMT
151E : ... YPT=>17
1520 : ... XPT=>00
1522 : ... BIAS=(@>8352)
1524 : ... 20': : '
1526 : ... END FMT
1527 : RTN

1528 : SCAN           Keyboard scanning
1529 : BR     GROM@>1528
152B : ST     @>8358,@>8373  Stack pointer
152E : ST     @>8373,@>835A  Old substack pointer
1531 : CEQ    @>8375,>45      E Exit
1534 : BS     GROM@>1371
1536 : CEQ    @>8375,>43      C Check
1539 : BS     GROM@>1444
153B : CEQ    @>8375,>52      R Record or Read
153E : BS     GROM@>133C
1540 : ST     @>8373,@>8358  Old substack pointer again
1543 : CEQ    @>8375,>0D      Enter
1546 : BR     GROM@>1528      None of the keys
1548 : RTN

1549 : ST     @>836C,>FF      Turn on motor
154C : DST    @>8368,@>8366  CRU basis address
154F : DST    @>836A,>016C
1553 : I/O    @>8368,>03      CRU output
1556 : CLR    @>8379          VDP interrupt timer
1558 : CGT    @>8379,>1E
155B : BR     GROM@>1558      Time delay
155D : RTN

155E : CLR    @>836C          Turn off motor
1560 : BR     GROM@>154C      Set CRU
1562 : CLR    @>8362
1564 : CLR    @>8379
1566 : CGT    @>8379,>3C      VDP interrupt timer
1569 : BR     GROM@>1566      Loop
156B : INC    @>8362
156D : CGT    @>8362,>0A      Waiting loop 10 times
1570 : BR     GROM@>1564
1572 : RTN

Subprogram, write text >8364=VDP address, >8358 Pointer to text
1573 : MOVE   >0002 TO @>8358 FROM GROM@>15A0(@>8362) Fetch pointer to text

```

```

157A : MOVE    >0002 TO @>8362 FROM GROM@>0000(@>8358) First 2 bytes in >8362
1581 : ST      @>8375,@>8363 ASCII key
1584 : DSRL    @>8362,>0008 Number bytes
1588 : MOVE    @>8362 TO VDP*>8364 FROM GROM@>0002(@>8358) Write text
158F : DADD    @>8362,@>8364 New address <8362
1592 : ADD     VDP*>8364,@>8352 Add screen offset
1596 : DINC    @>8364 Increase address
1598 : DCGE    @>8364,@>8362 End ?
159B : BR     GROM@>1592
159D : CALL    GROM@>0012 End over GSR return

```

Text :

```

15A0 : DATA  >15C0 = '>07,>00,:READING:'
15A2 : DATA  >15C9 = '>16,>00,:ERROR DETECTED IN DATA:'
15A4 : DATA  >15E1 = '>13,>FF,:PRESS CASSETTE STOP:'
15A6 : DATA  >15F6 = '>09,>00,:RECORDING:'
15A8 : DATA  >1601 = '>08,>00,:CHECKING:'
15AA : DATA  >160B = '>14,>FF,:REWIND CASSETTE TAPE:'
15AC : DATA  >1621 = '>15,>FF,:PRESS CASSETTE RECORD:'
15AE : DATA  >1638 = '>13,>FF,:PRESS CASSETTE PLAY:'
15B0 : DATA  >164D = '>14,>00,:CHECK TAPE (Y OR N)?:'
15B2 : DATA  >1663 = '>10,>00,:THEN PRESS ENTER:'
15B4 : DATA  >1675 = '>0F,>FE,:PRESS R TO READ:'
15B6 : DATA  >1686 = '>0F,>00,:PRESS E TO EXIT:'
15B8 : DATA  >1697 = '>07,>00,:DATA OK:'
15BA : DATA  >16A0 = '>15,>00,:ERROR - NO DATA FOUND:'
15BC : DATA  >16B7 = '>10,>00,:PRESS C TO CHECK:'
15BE : DATA  >16C9 = '>11,>00,:PRESS R TO RECORD:'

```

```

16DC : DATA  >00,>00,>00,>00

```

Keyboard table joysticks

```

16E0 : DATA  >00,>00,>00,>00
16E4 : DATA  >00,>00,>00,>00,>00,>00,>04,>04
16EC : DATA  >04,>FC,>04,>00,>00,>00,>FC,>04
16F4 : DATA  >FC,>FC,>FC,>00,>00,>00,>00,>04
16FC : DATA  >00,>FC,>00,>00

```

Keyboard table lower case

```

1700 : DATA  >FF,>FF,>FF,>FF
1704 : DATA  >FF,>0D,>20,>3D,>78,>77,>73,>32
170C : DATA  >39,>6F,>6C,>2E,>63,>65,>64,>33
1714 : DATA  >38,>69,>6B,>2C,>76,>72,>66,>34
171C : DATA  >37,>75,>6A,>6D,>62,>74,>67,>35
1724 : DATA  >36,>79,>68,>6E,>7A,>71,>61,>31
172C : DATA  >30,>70,>3B,>2F

```

Keyboard table SHIFT

```

1730 : DATA  >FF,>FF,>FF,>FF
1734 : DATA  >FF,>0D,>20,>2B,>58,>57,>53,>40
173C : DATA  >28,>4F,>4C,>3E,>43,>45,>44,>23
1744 : DATA  >2A,>49,>4B,>3C,>56,>52,>46,>24
174C : DATA  >26,>55,>4A,>4D,>42,>54,>47,>25
1754 : DATA  >5E,>59,>48,>4E,>5A,>51,>41,>21
175C : DATA  >29,>50,>3A,>2D

```

Keyboard table FCTN

```

1760 : DATA  >FF,>FF,>FF,>FF
1764 : DATA  >FF,>0D,>20,>05,>0A,>7E,>08,>04
176C : DATA  >0F,>27,>C2,>B9,>60,>0B,>09,>07
1774 : DATA  >06,>3F,>C1,>B8,>7F,>5B,>7B,>02
177C : DATA  >01,>5F,>C0,>C3,>BE,>5D,>7D,>0E
1784 : DATA  >0C,>C6,>BF,>C4,>5C,>B9,>7C,>03
178C : DATA  >BC,>22,>BD,>BA

```

Keyboard table CNTRL

```

1790 : DATA  >FF,>FF,>FF,>FF
1794 : DATA  >FF,>0D,>20,>9D,>98,>97,>93,>B2
179C : DATA  >9F,>8F,>8C,>9B,>83,>85,>84,>B3
17A4 : DATA  >9E,>89,>8B,>80,>96,>92,>86,>B4

```

17AC : DATA >B7,>95,>8A,>8D,>82,>94,>87,>B5
17B4 : DATA >B6,>99,>88,>8E,>9A,>91,>81,>B1
17BC : DATA >B0,>90,>9C,>BB
Keyboard table mode 1 and 2
17c0 : DATA >FF,>FF,>FF,>FF
17C4 : DATA >FF,>FF,>FF,>FF,>00,>04,>02,>07
17CC : DATA >09,>06,>0C,>0D,>0E,>05,>03,>08
17D4 : DATA >08,>05,>03,>0E,>0D,>06,>0C,>09
17DC : DATA >07,>04,>02,>00,>10,>0B,>11,>0A
17E4 : DATA >13,>12,>01,>0F,>0F,>12,>01,>13
17EC : DATA >0A,>0B,>11,>10

17F0 : DATA >00,>00,>00,>00
17F4 : DATA >00,>00,>00,>00,>00,>00,>00,>00
17FC : DATA >00,>00,>D1,>FF

HEXDUMP of GROM 0 (>0000 bis >17FF)

```

0000G AA02 0000 0000 0000 1310 1320 0000 0000
0010G 43D9 4439 49A5 4393 439B 4443 4446 4449 C D9I C C DCDFDI
0020G 404F 51FA 4C7E 4D55 4DB0 4E60 4EF5 4EFD @QQ L~MUM N`N N
0030G 4F5B 4F7C 43CB 43D3 054D 1252 5A44 1405 O[O|C C M RZD
0040G 2844 0537 B460 0D00 1100 43BF B0B0 7087 (D 7 ` C p
0050G 80CE BE8F 1100 70BE 8100 9FBE 8100 BFBE p
0060G 8100 DFBE 8100 FFBF 72FF 7E39 0007 0104 r ~9
0070G 4E86 0035 0071 0100 3500 3E80 8200 3500 N 5 q 5 > 5
0080G 0B74 0035 0008 80C2 00BF 0303 08F6 0203 t 5
0090G BF03 1001 F602 03BE 0318 F602 0384 00BE
00A0G 0302 F602 03BE 0301 F602 03BF 0316 02F6
00B0G 0203 0603 CB86 A000 BE70 10BE B070 A08E p p
00C0G A000 40D9 3900 0101 044C 86B0 70A0 7070 @ 9 L p pp
00D0G D670 4040 BBBE 80FD 0893 7039 0001 0102 p@@ p9
00E0G 4186 A000 350F FFA0 01A0 0031 0020 A380 A 5 1
00F0G 0455 3102 00A9 0004 B031 0050 A808 094C U1 1 P L
0100G 0720 BE7E 05BC 747E 0392 7E41 0587 7EBE ~ t~ ~A ~
0110G 7560 08C1 E075 FB01 14B0 9DC1 E075 FB01 u` u u
0120G 1CFB A67E 12A2 7508 D675 E041 12D6 7E03 ~ u u A ~
0130G 410F 877E 08A4 8E02 0102 039C 0204 0506 A ~
0140G 9C02 0708 09A7 8F1B 5245 4144 592D 5052 READY-PR
0150G 4553 5320 414E 5920 4B45 5920 544F 2042 ESS ANY KEY TO B
0160G 4547 494E FB31 0011 A128 0492 3100 18A2 EGIN 1 ( 1
0170G C404 8B31 000D A16A 04A3 BE43 1006 0379 1 j C y
0180G 8780 D086 55BE 6D04 0F19 6188 BF72 0080 U m a r
0190G BF90 7201 9F0F 1ABD 9073 9072 9672 008F r s r r
01A0G 80D0 418C 3900 0101 044D 8674 02FF 0341 A 9 M t A
01B0G AC06 03CB 0720 BE72 FEBE 6D06 866C 8680 r m l
01C0G FB31 001E A400 6000 BE80 FB04 3100 1EA4 1 ` 1
01D0G 2060 0086 5886 5905 01DC 9059 CE59 1D61 ` X Y Y Y a
01E0G EDD4 E400 58E4 2058 41EF 0501 DA41 FD94 X X A A
01F0G 7287 9072 9472 BF90 7212 A190 6CD6 8FDD r r r r l
0200G 00AA 4224 BD58 8FDD 068F 5862 2494 72BF B$ X Xb$ r
0210G 9072 FFFF 9472 BD90 7258 906C BD58 CF7D r r rX l X }
0220G 0058 4209 9472 8790 720F 1A62 2496 72D7 XB r r b$ r
0230G 0212 A142 4031 0001 5960 00D6 59AA 41B6 B@1 Y` Y A
0240G 3900 0101 044E 08A0 8102 0102 039C 0204 9 N
0250G 0506 9C02 0708 09A0 9E04 5052 4553 53FB PRESS
0260G 3100 11A0 2804 9231 000D A068 04A3 BF52 1 ( 1 h R
0270G 00E4 BE58 308E 6C42 9008 FF02 0F49 4E53 X0 lB INS
0280G 4552 5420 4341 5254 5249 4447 45FB 42EC ERT CARTRIDGE B
0290G 9058 BCB0 5258 9552 3100 03B0 5209 49A3 X RX R1 R I
02A0G 5200 04BD 6A90 7296 72BD 5C90 7296 72A3 R j r r \ r r
02B0G 6A00 0486 5E8E 5C62 CD35 0001 5FCF 7D00 j ^ \b 5 _ }
02C0G 6A91 6A34 5EB0 52CF 7D00 6A42 DD33 0001 j j4^ R } jB 3
02D0G 5F00 006A 916A 325E B052 0000 6AA3 5200 _ j j2^ R j R
02E0G 3AD2 7200 6290 BE43 1306 0379 3900 0101 : r b C y9
02F0G 044D 8674 02FF 0342 F4A6 7531 C875 6C43 M t B ul ulC
0300G 0706 03D3 0502 F406 03CB A46C 7592 6CE2 lu l
0310G 6C02 BC78 906C 946C BD5C 906C 955C BF72 1 x l l \ l \ r
0320G 9E80 8E78 632F BD80 80CF 7D00 5C43 3733 xc/ } \C73
0330G 0002 8080 0000 5C07 208F 80CE 4339 CF70 \ C9 p
0340G 1000 4353 BD00 70A7 000F FF34 00AF 1000 CS p 4
0350G AF0F FF86 0035 006F 0100 3500 3C80 8400 5 o 5 <
0360G 8674 3500 1FA3 81A3 8087 8082 8E78 6378 t5 xcx
0370G 9673 BD00 8080 0FF0 00BE 4260 3100 0228 s B`1 (
0380G 6000 D628 AA43 92D2 2900 6392 9473 BD90 ` ( C ) c s
0390G 7342 0031 0200 B04A 04B0 00BF 80D0 06B0 sB 1 J
03A0G BE80 D240 86B0 4A33 0007 E001 4A00 00D0 @ J3 J
03B0G A34A 0008 A380 D000 0792 80D2 43A4 00BF J C
03C0G 80D0 0870 BE80 D21F 0503 A4BF 5804 75F6 p X u
03D0G 5800 00BF 5804 8043 CF88 6D86 54BC 55B0 X X C m T U
03E0G 5686 58BD 5256 9152 D458 5563 F7D6 B052 V X RV R XUc R
03F0G 2E63 F790 5843 E68E 5864 35BC 5558 D255 .c XC Xd5 UX U

```

0400G	0864	3586	5487	80D0	9156	3454	4AB0	56A1	d5 T V4TJ V
0410G	5654	0F19	9473	BD90	7380	FA0F	1A44	2994	VT s s D)
0420G	73BD	9073	9072	9672	008F	80D0	441B	BD80	s s r r D
0430G	FA90	7396	73D4	0000	0196	73BD	80FA	9073	s s s s
0440G	9673	0005	284C	0528	4E05	2010	8060	20F0	s (L (N `
0450G	0EF9	86F8	F717	1717	1717	1717	1717	1717	
0460G	1706	0301	0B0C	0D0F	0402	0D08	0E05	090A	
0470G	0627	2722	2206	BFDF	FF80	0592	0A01	9F00	' '' ''
0480G	06BF	DFFF	8020	900A	019F	000A	3139	3831	1981
0490G	2020	5445	5841	5320	494E	5354	5255	4D45	TEXAS INSTRUME
04A0G	4E54	5348	4F4D	4520	434F	4D50	5554	4552	NTSHOME COMPUTER
04B0G	0000	0000	0000	0000	2020	2020	2020	0020	
04C0G	4848	4800	0000	0000	0048	FC48	48FC	4800	HHH H HH H
04D0G	103C	5038	1478	1000	C0C4	0810	2040	8C0C	<P8 x @
04E0G	6090	9060	6094	8874	0810	2000	0000	0000	` `` t
04F0G	0810	2020	2020	1008	4020	1010	1010	2040	@ @
0500G	0000	4830	CC30	4800	0000	1010	7C10	1000	H0 OH
0510G	0000	0000	0030	1020	0000	0000	7C00	0000	0
0520G	0000	0000	0000	3030	0004	0810	2040	8000	00 @
0530G	3844	4444	4444	4438	1030	5010	1010	107C	8DDDDDD8 OP
0540G	7884	0408	1020	40FC	7884	0438	0404	8478	x @ x 8 x
0550G	0C14	2444	84FC	0404	F880	80F8	0404	8478	\$D x
0560G	7880	80F8	8484	8478	FC04	0408	1020	4040	x x @@
0570G	7884	8478	8484	8478	7884	8484	7C04	0478	x x xx x
0580G	0030	3000	0030	3000	0030	3000	0030	1020	00 00 00 0
0590G	0008	1020	4020	1008	0000	007C	007C	0000	@ @
05A0G	0040	2010	0810	2040	3844	0408	1010	0010	@ @8D
05B0G	0078	849C	A498	807C	7884	8484	FC84	8484	x x
05C0G	F844	4478	4444	44F8	7884	8080	8080	8478	DDxDDD x x
05D0G	F844	4444	4444	44F8	FC80	80F0	8080	80FC	DDDDDD
05E0G	FC80	80F0	8080	8080	7884	8080	9C84	8478	x x
05F0G	8484	84FC	8484	8484	7C10	1010	1010	107C	
0600G	0404	0404	0484	8478	8890	A0C0	A090	8884	x
0610G	4040	4040	4040	407C	84CC	B484	8484	8484	@@@@@@
0620G	84C4	A494	8C84	8484	FC84	8484	8484	84FC	
0630G	F884	8484	F880	8080	7884	8484	8494	8874	x t
0640G	F884	8484	F890	8884	7884	8078	0404	8478	x x x
0650G	7C10	1010	1010	1010	8484	8484	8484	8478	x
0660G	4444	4444	2828	1010	8484	8484	84B4	CC84	DDDD((
0670G	8484	4830	3048	8484	4444	4428	1010	1010	H00H DDD(
0680G	FC04	0810	2040	80FC	3820	2020	2020	2038	@ 8 8
0690G	0080	4020	1008	0400	7010	1010	1010	1070	@ P P
06A0G	1028	4482	0000	0000	0000	0000	0000	00FC	(D
06B0G	0000	0000	0000	0010	1010	1010	0010	2828	((
06C0G	2800	0000	0028	287C	287C	2828	3854	5038	((((((8TP8
06D0G	1454	3860	6408	1020	4C0C	2050	5020	5448	T8`d L PP TH
06E0G	3408	0810	0000	0000	0810	2020	2010	0820	4
06F0G	1008	0808	1020	0028	107C	1028	0000	1010	((
0700G	7C10	1000	0000	0000	3010	2000	0000	7C00	0
0710G	0000	0000	0000	0030	3000	0408	1020	4000	00 @
0720G	3844	4444	4444	3810	3010	1010	1038	3844	8DDDDDD8 0 88D
0730G	0408	1020	7C38	4404	1804	4438	0818	2848	8D D8 (H
0740G	7C08	087C	4078	0404	4438	1820	4078	4444	@x D8 @xDD
0750G	387C	0408	1020	2020	3844	4438	4444	3838	8 8DD8DD88
0760G	4444	3C04	0830	0030	3000	3030	0000	3030	DD< 0 00 00 00
0770G	0030	1020	0810	2040	2010	0800	007C	007C	0 @
0780G	0000	2010	0804	0810	2038	4404	0810	0010	8D
0790G	3844	5C54	5C40	3838	4444	7C44	4444	7824	8D\T\@88DD DDDx\$
07A0G	2438	2424	7838	4440	4040	4438	7824	2424	\$8\$ \$x8D@@@D8x\$\$\$
07B0G	2424	787C	4040	7840	407C	7C40	4078	4040	\$ \$x @x@@ @x@@
07C0G	403C	4040	5C44	4438	4444	447C	4444	4438	@<@@\DD8DDD DDD8
07D0G	1010	1010	1038	0404	0404	0444	3844	4850	8 D8DHP
07E0G	6050	4844	4040	4040	4040	7C44	6C54	5444	`PHD@@@ D1TTD
07F0G	4444	4464	6454	4C4C	447C	4444	4444	447C	DDddTLLD DDDD
0800G	7844	4478	4040	4038	4444	4454	4834	7844	xDDx@@@8DDDDTH4xD
0810G	4478	5048	4438	4440	3804	4438	7C10	1010	DxPHD8D@8 D8

0820G	1010	1044	4444	4444	4438	4444	4428	2810	DDDDDD8DDD((
0830G	1044	4444	5454	5428	4444	2810	2844	4444	DDDTT(DD((DD
0840G	4428	1010	1010	7C04	0810	2040	7C38	2020	D(@ 8
0850G	2020	2038	0040	2010	0804	0038	0808	0808	8 @ 8
0860G	0838	0010	2844	0000	0000	0000	0000	007C	8 (D
0870G	0020	1008	0000	0000	0038	447C	4444	0000	8D DD
0880G	7824	3824	7800	003C	4040	403C	0000	7824	x\$8\$x <@@@< x\$
0890G	2424	7800	007C	4078	407C	0000	7C40	7840	\$\$x @x@ @x@
08A0G	4000	003C	405C	4438	0000	4444	7C44	4400	@ <@\D8 DD DD
08B0G	0038	1010	1038	0000	0808	0848	3000	0024	8 8 H0 \$
08C0G	2830	2824	0000	4040	4040	7C00	0044	6C54	(0(\$ @@@@ DLT
08D0G	4444	0000	4464	544C	4400	007C	4444	447C	DD DdTLD DDD
08E0G	0000	7844	7840	4000	0038	4454	4834	0000	xDx@@ 8DTH4
08F0G	7844	7848	4400	003C	4038	0478	0000	7C10	xDxHD <@8 x
0900G	1010	1000	0044	4444	4438	0000	4444	2828	DDDD8 DD((
0910G	1000	0044	4454	5428	0000	4428	1028	4400	DDTT(D((D
0920G	0044	2810	1010	0000	7C08	1020	7C18	2020	D(
0930G	4020	2018	1010	1000	1010	1030	0808	0408	@ 0
0940G	0830	0020	5408	0000	0046	4F52	0103	0303	0 T FOR
0950G	0303	0303	FC04	0505	0406	020C	0080	4040	@@
0960G	8000	0C12	FF80	C040	6038	1C0E	1921	213D	@`8 !=
0970G	0505	05C4	BA8A	8ABA	A1A1	A122	0301	0000	"
0980G	0000	0000	E231	1018	0C07	0300	4C90	2040	1 L @
0990G	4020	E000	3C42	99A1	A199	423C	3500	1AA3	@ <B B<5
09A0G	C010	8752	0006	099C	3500	0A1A	4ABF	5820	R 5 J X
09B0G	20D2	4A00	69BB	834A	BE59	2DBE	165A	8F4A	J i J Y- Z J
09C0G	49DE	BF90	1630	0090	16CE	5500	4A9B	BE10	I 0 U J
09D0G	01D2	5700	49D9	A010	5706	0C42	4A9B	060B	W I W BJ
09E0G	CF8E	554A	EBCA	770A	6A00	BC10	4AA2	100C	UJ w j J
09F0G	8E90	104A	0090	10D2	1052	49F0	8618	4A20	J RI J
0A00G	BE12	05CE	7709	6A1A	D277	FC4A	1ABE	1209	w j w J
0A10G	CE77	FE6A	1A90	12A0	1277	060B	6FBE	18FF	w j w o
0A20G	CE77	096A	60CE	77FA	6A55	D277	F64A	60BE	w j` w jU w J`
0A30G	1052	BC12	14A2	1204	9612	9210	8E90	106A	R j
0A40G	38BC	1190	1086	10AE	100A	8E11	4A50	9212	8 JP
0A50G	CC12	776A	60BE	120C	060B	E806	0C55	4A9B	wj` UJ
0A60G	BE12	08BE	1503	A415	1406	0C01	060C	55BF	U
0A70G	9016	452B	8E76	6A7F	BF90	1645	2D81	7694	E+ vj E- v
0A80G	16BF	9016	2A2A	CA77	646A	96AE	760A	A376	** wdj v v
0A90G	3030	BD90	1676	9416	8690	16BE	5659	BC12	00 v VY
0AA0G	59BE	9056	2090	56D6	9056	306A	A18E	9056	Y V V V0j V
0AB0G	6AC6	D690	5645	6AC6	D790	562E	006A	C6D7	j VEj V. j
0AC0G	9056	2E45	4ACF	9656	BF90	5620	304A	D592	V.EJ V V 0J
0AD0G	56BC	9056	12BC	5556	9056	8E90	564A	D8A4	V V UV V VJ
0AE0G	5655	3500	1A10	A3C0	8752	01D0	7755	6B3E	VU5 R wUk>
0AF0G	060B	2FD2	12FF	4B54	D012	554B	02BC	1255	/ KT UK U
0B00G	9212	060B	6FD0	7755	6B3E	060B	2FA2	1203	o w k> /
0B10G	D212	034B	54BC	1055	9410	CC12	104B	22BC	KT K"
0B20G	1210	A412	1406	0BE8	D257	004A	5B4A	9BBC	W J[J
0B30G	1255	D257	004B	3ABC	1257	A012	7700	8E56	U W K: W w V
0B40G	4B58	BE90	1645	9016	9255	4B42	8690	16BE	KX E UKB
0B50G	5659	4AD5	8E56	69C2	3500	0A4A	1ABC	1256	VYJ Vi 5 J V
0B60G	9212	060B	6FBC	1256	9412	A412	144A	63A4	o Jc
0B70G	7712	A412	14E6	1201	9412	B212	7FCE	1207	w
0B80G	6BCF	BE10	31DA	7701	6B8D	BE10	04BE	144A	k 1 w k J
0B90G	8775	BC77	4ABC	184A	A014	12A0	9014	10BC	u wJ
0BA0G	5412	0F02	8612	D614	4B4B	B0D4	184A	4BC3	T KK JK
0BB0G	D610	044B	C5BC	1390	14AE	120A	AA12	0ABC	K
0BC0G	9014	1390	1486	9014	9014	D214	524B	C5BC	RK
0BD0G	764A	A676	40E2	7601	DF76	0008	8614	D24B	vJ v@ v v K
0BE0G	0A4B	E790	7790	1400	A477	14BC	1577	A215	K w w w
0BF0G	03D2	7700	6C01	BE10	FFA4	1077	060C	4286	w l w B
0C00G	1587	52BE	134B	060C	2F86	10BC	1190	1390	R K /
0C10G	13AE	100A	A310	3030	060C	29BC	1011	060C	0)
0C20G	294C	09BE	102E	BC17	16BC	9016	1090	1692)L .
0C30G	156C	23D6	1669	6C3C	9212	4C41	9673	8690	l# il LA s

0C40G	1600	BE90	162E	9016	9210	CE10	004C	3E8E				L>
0C50G	9016	304C	4692	16D6	9016	306C	5590	168E	OLF		01U	
0C60G	184C	3EBC	1617	4C3E	A36E	0008	3500	08B0	L>	L	n	5
0C70G	6E4A	0035	0008	4AB0	6EA3	6EFF	F800	0611	nJ	5	J	n
0C80G	F28F	4A6D	3D8F	5C6D	35A3	6E00	0886	7506	Jm=	\	5	n
0C90G	0C68	0611	FA86	7506	11DC	0F0F	4D47	060C	h	u		MG
0CA0G	6886	540F	1281	4ABD	A3DB	4A06	0C73	8E54	h	T	J	J
0CB0G	4CFE	0611	DC06	0C68	93A3	DB6C	E0DA	A3DC	L			l
0CC0G	016C	C90F	0DA3	6E00	08E7	A3DB	0001	8FA3	l	n		
0CD0G	DB6C	E006	11DC	060C	680F	0D06	11DC	4CBD	l	h		L
0CE0G	A36E	FFF0	D2E0	086E	006C	FD8E	544C	F931	n		l	TL
0CF0G	0008	5C10	6D0F	094C	FD87	4A86	5400	D2B0	\	m		J
0D00G	6E00	6D1B	BC56	4A80	56CE	5646	6D1B	BC75	n	m	VJ	V
0D10G	56A2	750B	BC75	9075	EA75	01BC	A3DC	7506	V	u	u	u
0D20G	11DC	814A	060E	600F	0D06	0DB0	D2A3	DC00	J			
0D30G	6D34	834A	00D2	4A00	4D44	874A	0031	0008	m	4	J	J
0D40G	4A10	4100	0F05	00D2	B06E	006D	1BA3	6EFF	J	A		n
0D50G	F8BE	5405	0006	099C	0F03	6AE2	0A4D	ABBC	T		j	M
0D60G	144A	BE4A	3FA2	14C1	DF14	0008	E314	0001	J	J?		
0D70G	060C	6806	0C68	BF12	1027	0611	A906	11DC	h	h		
0D80G	BF12	1041	0611	A90F	0EBE	1203	3500	085C	A		5	\
0D90G	B06E	060C	680F	090F	0B31	0008	5C0F	DF0F	n	h	1	\
0DA0G	0892	124D	8CA3	6EFF	F84E	39BE	5404	4AE2	M	n	N9	T
0DB0G	0609	9C31	0008	5C0F	EF0F	0806	0C68	0611	1	\		h
0DC0G	FA31	0008	5C0F	D70F	0A6D	E30A	4DD5	835C	1	\	m	M
0DD0G	0F0A	0A4D	E3A3	6EFF	F8BD	764A	8675	0F04	M	n	vJ	u
0DE0G	050A	E206	0C68	0F12	BD14	4A06	0C73	E314	h		J	s
0DF0G	0001	0F0C	3100	085C	0FDF	0F0A	0A6E	0583	1			n
0E00G	5C0F	0691	1406	0C68	BF12	1053	0611	9906	\	h	S	
0E10G	11DC	BF12	106D	0611	A206	11F2	A36E	0008		m		n
0E20G	060C	680F	0635	0008	1A4A	060C	7306	11DC	h	5	J	s
0E30G	0F0C	3500	085C	1A0F	09DA	1501	6E46	3100	5	\		nF1
0E40G	085C	0FE7	0F08	3100	085C	1041	DA15	026E	\	1	\	A
0E50G	54BE	5D0A	DF14	0002	A05C	150F	0805	0AE2	T]	\	
0E60G	0F03	6EF1	0A4E	F106	099C	060B	CF8E	144E	n	N		N
0E70G	8131	0008	5C10	41BE	5D0A	0F08	060B	CF93	1	\	A]
0E80G	7691	76BE	4A3F	BDA3	DA76	3100	085C	0FE7	v	v	J?	v1
0E90G	0F08	0611	7706	0C68	BF12	108F	0611	9906	w	h		\
0EA0G	11DC	BF12	10B9	0611	A20F	0E06	0C68	BD5C				h
0EB0G	A3DA	3100	065E	0FE1	8F5C	6ECE	815C	CF5C	1	^	\	n
0EC0G	0063	6EE3	BE5C	408E	A3DA	6ECE	835C	3100	cn	\	@	n
0ED0G	084A	0FDF	0F07	3100	085C	0FF7	0F08	0F0B	J	1	\	1
0EE0G	050A	E2A7	5C00	64BC	5E5D	BF5C	4101	050E		\	d	^
0EF0G	C7BE	5406	0031	0008	5C0F	FF0F	0606	099C	T	1	\	A
0F00G	3100	085C	1007	0F08	BCA3	DA4A	814A	CE4A	1	\		J
0F10G	444F	18BE	5407	4AE2	060C	6806	11FA	8616	DO	T	J	h
0F20G	8F4A	6F39	BC77	4AA6	7746	CE77	006F	36A2	Jo	9	wJ	wF
0F30G	7751	BC16	9077	B216	030F	0CDA	1601	6F48	wQ	w		oH
0F40G	3100	085C	1041	0F07	E616	018E	166F	5284	1	\	A	oR
0F50G	A3DA	BF12	10E3	0611	994F	CD06	0C68	060E			O	h
0F60G	FD06	11DC	060E	F506	11F2	D654	076F	750F			T	ou
0F70G	036F	760F	0900	BC75	5C0F	0500	0609	9CBC	ov		u	\
0F80G	A3DA	4A81	4A87	1631	0008	5C10	170F	0A6F	J	J	1	\
0F90G	B909	6FB9	3100	085C	101F	0F0A	096F	B231	o	1	\	o
0FA0G	0008	5C10	41BF	5CBF	FF0F	09BF	160F	FF05	\	A	\	
0FB0G	0FB9	0611	77BF	1610	0FBF	1211	2506	1199	w			%
0FC0G	8F16	6FCD	3300	085C	0000	160F	06D2	A3DA	o	3	\	
0FD0G	006A	E283	4A4A	E241	011B	0000	0000	003F	j	JJ	A	?
0FE0G	3200	0000	0000	0040	0310	164D	4201	453F	2		@	MB
0FF0G	2B2A	5E30	1303	1940	021E	1955	091D	5E40	+	+	^	@
1000G	0139	0760	2043	5F3F	3F42	134D	1743	3A3F	9	\	C	??
1010G	4E35	6210	2161	2D3F	292A	0D38	1749	0A40	N5b	!	a	??
1020G	0229	2A0D	3817	493F	3A51	165A	0000	003F)	*	8	I
1030G	3443	5732	0000	003E	3A51	1400	0000	0080	4CW2		>	Q
1040G	0040	0100	0000	0000	003F	0963	6350	0000	@		?	ccP
1050G	0080	0040	121F	173C	0F5C	4B41	081F	2843	@		<	\

1060G	151D	2541	334E	0913	5B33	3E80	0040	0100	%A3N [3> @
1070G	0000	0000	0041	013B	2529	3424	0341	1B09	A ;%)4\$ A
1080G	1F45	2855	1041	2C61	3F23	3928	3A80	003F	E(U A,a?#9(: ?
1090G	2343	050A	1E58	2CBF	F562	1E21	1F24	5840	#C X, b ! \$X@
10A0G	3F4D	3652	1C56	11BE	FF08	5347	1623	3A40	?M6R V SG #: @
10B0G	395E	4951	262C	2C80	0040	0100	0000	0000	9^IQ&, , @
10C0G	00BF	F30D	1961	4858	2E40	2F2D	1216	2402	aHX. @/ - \$
10D0G	3DBF	C007	403A	0734	3840	1C61	245A	4516	= @: 48@ a\$ZE
10E0G	1680	00C4	FA2C	4910	0000	003C	0544	5203	,I < DR
10F0G	211A	58C2	FD3B	5809	0B46	1F3E	013C	2C0B	! X ;X F > < ,
1100G	442E	62C1	D251	4B29	1F06	023F	0760	5C3E	D.b QK) ? ` \>
1110G	3E2D	3EC0	C03B	4009	4B06	1640	0139	0760	>-> ;@ K @ 9 `
1120G	2043	5F80	00C0	FE35	3912	4F58	143F	0502	C_ 59 OX ?
1130G	4F0D	5426	55C0	FA32	4563	5E01	283F	0743	O T&U 2Ec^ (? C
1140G	250C	2B5B	40C0	F708	5F2F	5B60	483F	0B0B	% +[@ _/[`H?`
1150G	0A31	5C32	35C0	F21C	390C	454B	603F	1363	1\25 9 EK`? c
1160G	6363	6159	60C0	DF21	2121	2120	1940	0100	ccaY` !!!! @
1170G	0000	0000	0080	0006	0C68	0611	8E06	118E	h
1180G	0611	DC31	0008	5C10	410F	060F	0E00	3100	1 \ A 1
1190G	085C	0FDF	835C	0F06	0006	0C68	0611	A20F	\ \ h
11A0G	0D00	3500	085C	4A0F	0806	0C68	3300	084A	5 \J h3 J
11B0G	0000	1251	C635	0008	5CB0	6E0F	0833	0008	Q 5 \ n 3
11C0G	5C00	0012	0F06	A312	0008	3300	025C	0000	\ 3 \
11D0G	12D7	5C80	0051	B5A3	6EFF	F800	060C	68A3	\ Q n h
11E0G	6EFF	F835	0008	4AB0	6E35	0008	B06E	E008	n 5 J n5 n
11F0G	6E00	3500	085C	B06E	51D7	BC75	4A81	4ABC	n 5 \ nQ uJ J
1200G	774A	8676	D277	4052	47CE	7745	723D	BC55	wJ v w@RG wEr= U
1210G	77A6	5546	BC54	5586	56BE	5752	A057	55B4	w UF TU V WR WU
1220G	5690	5786	9057	9057	9055	521F	D275	0072	V W W W UR u r
1230G	3D8E	5672	3DA2	5407	0F02	0512	44D2	7500	= Vr= T D u
1240G	7244	834A	8654	0087	4AD2	7500	7252	BF4A	rD J T J u rR J
1250G	BFFF	874C	874E	8750	5244	3100	4000	1263	L N PRD1 @ c
1260G	0FF0	0083	02C0	6083	4AC0	A083	4CD7	E083	` J L
1270G	E3D7	C102	6140	00D0	EFFB	FE02	0500	0809	a@
1280G	140A	1317	0202	2480	0006	0516	F9D7	E083	\$
1290G	E3D7	C102	413F	FFDB	C4FF	FE05	8106	0216	A?
12A0G	E604	5B01	B115	5245	5649	4557	204D	4F44	[REVIEW MOD
12B0G	554C	4520	4C49	4252	4152	5900	0000	0000	ULE LIBRARY
12C0G	0000	0000	0000	0000	0000	0000	0000	0000	
12D0G	0000	0000	0000	0000	0000	0000	0000	0000	
12E0G	0000	0000	0000	0000	0000	0000	0000	0000	
12F0G	0000	0000	0000	0000	0000	0000	0000	0000	
1300G	0000	0000	0000	0000	0000	0000	0000	0000	
1310G	1318	1326	0343	5331	0000	132C	0343	5332	& CS1 , CS2
1320G	0000	1573	0103	BF66	0016	5330	BF66	0017	s f S0 f
1330G	BC5A	73A5	5654	B2EF	FFF7	561F	3500	0A4A	Zs VT V 5 J
1340G	FFFF	F656	BD5E	4C8A	4A53	8754	0E53	CF53	V ^L JS T S S
1350G	DA53	8753	F254	8954	0E53	5D53	80B6	FFFF	S S T T S]S
1360G	F756	6006	1549	0615	1686	7FBC	735A	0600	V` I sZ
1370G	12B6	FFFF	F756	C006	14A9	0406	1528	5363	V (Sc
1380G	86EF	FFFC	5653	638E	4E53	91BE	FFFF	FA56	VSc NS V
1390G	40A2	FFFF	FA56	3FB2	FFFF	FA56	C0DA	4B15	@ V? V K
13A0G	73B5	D667	1773	5DDA	4B13	535D	0615	0306	s g s] K S]
13B0G	14A9	0E53	C1DA	4B02	735D	0615	0306	14A9	S K s]
13C0G	0C06	1528	0615	4906	1562	0615	5E53	66BE	(I b ^Sf
13D0G	6205	BCEF	FFFF	564E	53DD	BE62	0486	5CBC	b VNS b \
13E0G	5DEF	FFFA	5606	1549	F45C	6273	7106	155E] V I \bsq ^
13F0G	536E	D667	1773	5D06	1503	0614	9F00	8E80	Sn g s]
1400G	CE53	FEF6	5C05	7455	0614	A918	8653	0615	S \ tU S
1410G	5E06	14A9	0406	1528	8E53	7363	D6EF	FFF6	^ (Ssc
1420G	5606	5363	D667	1773	6306	14A9	1003	542D	V Sc g sc T-
1430G	BF7E	171B	BC7D	75A0	7D52	D675	4E73	63D6	~ }u }R uNsc
1440G	7559	542D	0615	0306	149F	088E	80CE	544B	uYT- TK
1450G	F65C	0654	08DA	7C01	7463	0615	5E06	14A9	\ T tc ^
1460G	0254	6A06	155E	0614	A91A	D6EF	FFF6	5606	Tj ^ V
1470G	7478	0614	FE14	547C	0614	FE1E	0614	FE1C	tx T

1480G	0614	FE16	0615	2853	3C06	1503	0614	9906
1490G	0615	62F6	5C04	0514	0E06	14A9	0C54	A306
14A0G	14A9	0E06	1528	0615	4935	02C0	A000	A040
14B0G	08FE	16FF	00FD	525F	2002	2020	2A5C	20FB
14C0G	8662	8863	BF64	02E4	BE4A	03BF	5400	01BE
14D0G	6D0A	8780	D006	003D	8E75	7513	BF7E	171B
14E0G	08FD	5202	4353	31FB	D667	1674	F090	A2FD
14F0G	D675	FE75	13BF	6200	1206	1516	54C4	0615
1500G	1654	C006	1549	0614	A90A	0615	2806	155E
1510G	BD5C	5005	0034	3502	E0A0	00A0	2008	FE17
1520G	FF00	FD52	5F20	FB00	0355	28BC	5873	BC73
1530G	5AD6	7545	7371	D675	4374	44D6	7552	733C
1540G	BC73	58D6	750D	5528	00BE	6CFF	BD68	66BF
1550G	6A01	6CF6	6803	8679	CE79	1E55	5800	866C
1560G	554C	8662	8679	CE79	3C55	6690	62CE	620A
1570G	5564	0033	0002	5815	A062	3300	0262	0000
1580G	58BC	7563	E762	0008	3262	B064	0002	58A1
1590G	6264	A0B0	6452	9164	D164	6255	9206	0012
15A0G	15C0	15C9	15E1	15F6	1601	160B	1621	1638
15B0G	164D	1663	1675	1686	1697	16A0	16B7	16C9
15C0G	0700	5245	4144	494E	4716	0045	5252	4F52
15D0G	2044	4554	4543	5445	4420	494E	2044	4154
15E0G	4113	FF50	5245	5353	2043	4153	5345	5454
15F0G	4520	5354	4F50	0900	5245	434F	5244	494E
1600G	4708	0043	4845	434B	494E	4714	FF52	4557
1610G	494E	4420	4341	5353	4554	5445	2054	4150
1620G	4515	FF50	5245	5353	2043	4153	5345	5454
1630G	4520	5245	434F	5244	13FF	5052	4553	5320
1640G	4341	5353	4554	5445	2050	4C41	5914	0043
1650G	4845	434B	2054	4150	4520	2859	204F	5220
1660G	4E29	3F10	0054	4845	4E20	5052	4553	5320
1670G	454E	5445	520F	FE50	5245	5353	2052	2054
1680G	4F20	5245	4144	0F00	5052	4553	5320	4520
1690G	544F	2045	5849	5407	0044	4154	4120	4F4B
16A0G	1500	4552	524F	5220	2D20	4E4F	2044	4154
16B0G	4120	464F	554E	4410	0050	5245	5353	2043
16C0G	2054	4F20	4348	4543	4B11	0050	5245	5353
16D0G	2052	2054	4F20	5245	434F	5244	0000	0000
16E0G	0000	0000	0000	0000	0000	0404	04FC	0400
16F0G	0000	FC04	FCFC	FC00	0000	0004	00FC	0000
1700G	FFFF	FFFF	FF0D	203D	7877	7332	396F	6C2E
1710G	6365	6433	3869	6B2C	7672	6634	3775	6A6D
1720G	6274	6735	3679	686E	7A71	6131	3070	3B2F
1730G	FFFF	FFFF	FF0D	202B	5857	5340	284F	4C3E
1740G	4345	4423	2A49	4B3C	5652	4624	2655	4A4D
1750G	4254	4725	5E59	484E	5A51	4121	2950	3A2D
1760G	FFFF	FFFF	FF0D	2005	0A7E	0804	0F27	C2B9
1770G	600B	0907	063F	C1B8	7F5B	7B02	015F	C0C3
1780G	BE5D	7D0E	0CC6	BFC4	5CB9	7C03	BC22	BDBA
1790G	FFFF	FFFF	FF0D	209D	9897	93B2	9F8F	8C9B
17A0G	8385	84B3	9E89	8B80	9692	86B4	E795	8A8D
17B0G	8294	87B5	B699	888E	9A91	81B1	E090	9CBB
17C0G	FFFF	FFFF	FFFF	FFFF	0004	0207	0906	0C0D
17D0G	0E05	0308	0805	030E	0D06	0C09	0704	0200
17E0G	100B	110A	1312	010F	0F12	0113	0A0B	1110
17F0G	0000	0000	0000	0000	0000	0000	0000	D1FF

```

(S<
b \      T
( I5    @
R_     *\<
b c d   J T
m      = uu ~
R CS1  g t
u u b   T
T I     ( ^
\P 45
R_     U( Xs s
Z uEsq uCtD uRs<
sX u U( l hf
j l h y y UX l
UL b y y<Uf b b
Ud 3 X b3 b
X uc b 2b d X
bd dR d dbU
! 8
M c u
READING ERROR
DETECTED IN DAT
A PRESS CASSETT
E STOP RECORDIN
G CHECKING REW
IND CASSETTE TAP
E PRESS CASSETT
E RECORD PRESS
CASSETTE PLAY C
HECK TAPE (Y OR
N)? THEN PRESS
ENTER PRESS R T
O READ PRESS E
TO EXIT DATA OK
ERROR - NO DAT
A FOUND PRESS C
TO CHECK PRESS
R TO RECORD
=xws29ol.
ced38ik,vrf47ujm
btg56yhnzqa10p;/
+XWS@(OL>
CED#*IK<VRF$&UJM
BTG%^YHNZQA!)P:-
~
'
?  [ {
] } \ | "

```

THE BASIC GROMS

Essential parts of Basic are located in GROMS 1 and 2 . Among them are the routines for input of commands, the crunching of input lines into Basic programs, the start of execution of Basic programs, several Basic commands and all subprograms which can be activated by Basic command CALL.

Besides the possibility of putting Basic programs into GROMS which can be used i modules (i.e. Module Data management and Analysis), the TI Basic mostly uses th VDP RAM. The use of the memory space starts at the top with the program containing the program lines. These program lines are composed of a byte which indicates the length and then of a crunched line, i.e. not the input line is stored but a shorter version (so-called tokens) in which all Basic commands are crunched into 1 byte. The end of the line is indicated by a byte with value >00. The value of the tokens can be seen on the tables starting at GROM address >2870.

There are 3 additional tokens:

C7 means a following string in parenthesis;

C8 means a string without parenthesis and

C9 means that a line number (2 bytes) follows.

The last line is always stored at the lower end i.e. the lines are not sorted according to their numbers. The second part of the program consists of the line list. It consists of 2 bytes for the line number and 2 bytes for a pointer indicating to the start of the line in RAM.

Then follows the symbol list also known as variable list. It contains all the important information for the variables (structure see below). Following that is the space for the Peripheral Access Blocks (PAB), which are necessary for each file, and finally the space to store the string variables.

Basic also uses a stack for temporary storage of values. It starts at the top of the RAM used by VDP for the screen display and grows to the top. Often, the GPL command PARSE is used in the Basic routines to obtain the value of a variable or of an expression.

Basic also uses extensively the Scratch Pad RAM >8300 through >83FF. In details:

>8300->8317 Generally used for temporary storage.
>8318 Pointer to start RAM space used for strings(String Space), high address.
>831A Pointer to end of RAM space used for strings.
(low address)
>831C Temporary pointer to string.
>831E Step for MUM mode.
>8320 Pointer to start of screen input.
>8322 Error-Code for the transfer between Assembler routines and GPL-routines.

>8324 Pointer to start of Basic value stack and of character definitions.
 >8326 Return address for the return of Basic Assembler routines to GPL.
 >8328 Address for jump table in GROM for the Basic commands.
 >832A Pointer to end of screen input (may also be located outside of screen area).
 >832C Pointer to text in a Basic line.(Text-Pointer)
 >832E Pointer to line in line list just executed.
 >8330 Pointer to start line list (low address is essentially the end of a line list since here the highest line number is located.
 >8332 Pointer to end of line list (high address).
 >8334 Pointer to next data element.
 >8336 Pointer to line in line list for next data element.
 >8338 Pointer to actual position in crunch buffer.
 >833A Pointer to free space used by PAB's.
 >833C Pointer to first entry in PAB list(there is an entry for each file).
 >833E Pointer to first entry into symbol list(variable list).
 >8340 Pointer to free space which can be used by symbol list.
 >8342 Executed character. >8343 Value of option base.
 >8344 Run-flag.
 >8346 Start line number.
 >834A->836D FAC used as for the assembler routines.
 >836E Pointer to upper end of Basic value stack.
 >8370 Pointer to end of usable VDP RAM.
 >8372->837F Is used by GPL.
 >8388 Various flags.
 >8389 GROM/VDP flag(0=VDP).
 >838A->83BF Subroutine and data stack for GPL.
 >83C0->83FF GPL working space.

To this day, no versions are known of these Basic GROMS. Therefore we can do without the Hex-dumps of the sample GROMS.

THE BASIC VALUE STACK

The entries vary depending on what is stored on the stack. Every entry has a length of 8 bytes.

Kind	1st word	2nd word	3rd word	4th word
Numeric expression	8 bytes floating point number			
String expression	>001C	>6500	Pointer	Length of
Numeric variable	Pointer to entry symbol table	>0000	Pointer to value	>0000
String variable	Pointer to entry symbol table	>6500	Pointer to value	Length of string
Numeric data field	Pointer to entry symbol table	Hbyte=0 Lbyte= Number of dim.	Pointer to value	>0000
String data field	Pointer to entry symbol table	Hbyte=>65 Lbyte= number of dim.	Pointer to value	>0000
GOSUB	Pointer to return row in row list	>6600		
FOR	Pointer to entry symbol table	>6700	Pointer to value	Pointer to row in row list
	- - - - -	Value of increase	- - - - -	- - - - -
	- - - - -	Limit of Loop	- - - - -	- - - - -
DEF	Pointer to row	>68 plus >00 for numeric >80 for string	Old pointer symbol table	Old pointer Free space string

THE SYMBOL TABLE

The entries in the symbol table are constructed as follows:

1st word

Byte 1: Bit 0 (MSB): String flag, when set, is a string flag for
Bit 1: user defined functions.
Bit 2 - 4 : Not used
Bit 5 - 7 : Number of dimensions i.e. number of
parameters.

Byte 2: Bit 0 - 3 : Not used
Bit 4 - 7 : Length of variable names

2nd word: Pointer to next entry in symbol table.
(0000 at the end)

3rd word: Pointer to name of variable

4th word and on: Space for value of variable.

Details:

DEF: 1 word pointer on definition
String: 1 word pointer to value of string
Numeric: 8 bytes for value
Data field: 1 word each with limit in each dimension.
At numeric data field follows space for
values of all elements.(8 Bytes each)
At string data field each element is
followed by one pointer (2 Bytes) to the
values of the corresponding string.

THE PAB LIST

Each entry for a file consists of the following details:

Byte 0 and 1: Pointer to next entry in PAB-list
Byte 2: File number
Byte 3: Internal offset (is used to write into PAB buffer)
Byte 4: I/O Op Code for DSR
Byte 5: Flag byte
Byte 6,7: Pointer to buffer
Byte 8: Maximum length of data block
Byte 9: Length of actual data block
Byte 10,11: Number of data block (only for relative files)
Byte 12: Screen offset
Byte 13: Length of file name
Byte 14: File name

```

*****
*
*                BASIC GROM ANALYSIS
*
*                30.12.84  H. MARTIN
*
*****

```

Header:

```

2000 : DATA  >AA02
2002 : DATA  >0100
2004 : DATA  >0000
2006 : DATA  >214D      Program
2008 : DATA  >0000
200A : DATA  >4D1A      Subprograms
200C : DATA  >0000
200E : DATA  >0000

2010 : BR      GROM@>2417  Execution of a Basic program in GROM
2012 : BR      GROM@>2195  Return Basic
2014 : BR      GROM@>260B  "Crunch" input line
2016 : BR      GROM@>266C  Routine syntax error output
2018 : BR      GROM@>267E  Sets back cursor position after an error
201A : BR      GROM@>2192  2nd entry point Basic
201C : BR      GROM@>27F1  Load character block and color VDP
201E : BR      GROM@>236D  Shift blocks in VDP RAM
2020 : BR      GROM@>26AB  Reset length byte

```

Text (with screen offset):

```

2022 : TEXT    '>8A,>80,>B7,>A1,>B2,>AE,>A9,>AE,>A7,>9A'      * WARNING
202C : TEXT    '>13,>A9,>AE,>A3,>AF,>B2,>B2'                  INCORRECT STATEMENT
                >A5,>A3,>B4,>80,>B3,>B4,>A1,>B4,>A5,>AD,>A5,>AE,>B4'
2040 : TEXT    '>08,>A2,>A1,>A4,>80,>AE,>A1 >AD,>A5'          BAD NAME
2049 : TEXT    '>0B,>AD,>A5,>AD,>AF,>B2,>B9,>80,>A6,>B5,>AC,>AC,   MEMORY FULL
2055 : TEXT    '>0E,>A3,>A1,>AE,>87,>B4,>80,>A3,>AF,>AE,>B4,>A9,>AE,>B5,>A5'
                CAN'T CONTINUE
2064 : TEXT    '>09,>A2,>A1,>A4,>80,>B6,>A1,>AC,>B5,>A5'          BAD VALUE
206E : TEXT    '>0E,>AE,>B5,>AD,>A2,>A5,>B2,>80,>B4,>AF,>AF,>80,>A2,>A9,>A7'
                NUMBER TOO BIG
207D : TEXT    '>16,>B3,>B4,>B2,>A9,>AE,>A7,>8D,>AE,>B5,>AD,>A2,>A5,>B2,>80,>AD
                >A9,>B3,>AD,>A1,>B4,>A3,>A8'          STRING NUMBER MISMATCH

2094 : TEXT    '>0C,>A2,>A1,>A4 >80,>A1,>B2,>A7,>B5,>AD,>A5,>AE,>B4'
                BAD ARGUMENT
20A1 : TEXT    '>0D,>A2,>A1,>A4,>80,>B3,>B5,>A2,>B3,>A3,>B2,A9,>B0,>B4'
                BAD SUBSCRIPT
20AF : TEXT    '>0D,>AE,>A1,>AD,>A5,>80,>A3,>AF,>AE,>A6,>AC,>A9,>A3,>B4'
                NAME CONFLICT
20BD : TEXT    '>0D,>A3,>A1,>AE,>87,>B4,>80,>A4,>AF,>80,>B4,>A8,>A1,>B4'
                CAN'T DO THAT
20CB : TEXT    '>B4,>A9,>80,>A2,>A1,>B3,>A9,>A3,>80,>B2,>A5,>A1,>A4,>B9'
                TI BASIC READY
20D9 : TEXT    '>0F,>A2,>A1,>A4,>80,>AC,>A9,>AE,>A5,>80,>AE,>B5,>AD,>A2,>A5,>B2'
                BAD LINE NUMBER
20E9 : TEXT    '>8A,>80,>A2,>B2,>A5,>A1,>AB,>B0,>AF,>A9,>AE,>B4,>80,>A1,>B4,>80'
                * BREAKPOINT IN
20F9 : TEXT    '>0E,>A6,>AF,>B2,>8D,>AE,>A5,>B8,>B4,>80,>A5,>B2,>B2,>AF,>B2'
                FOR-NEXT ERROR
2108 : TEXT    '>B4,>B2,>B9,>80,>A1,>A7,>A1,>A9,>AE,>9A,>80'
                TRY AGAIN:
2113 : TEXT    '>09,>A9,>8F,>AF,>80,>A5,>B2,>B2,>AF,>B2'          I/O ERROR
211D : TEXT    '>0A,>A6,>A9,>AC,>A5,>80,>A5,>B2,>B2,>AF,>B2'      FILE ERROR
2128 : TEXT    '>0B,>A9,>AE,>B0,>B5,>B4,>80,>A5,>B2,>B2,>AF,>B2'  INPUT ERROR
2134 : TEXT    '>0A,>A4,>A1,>B4,>A1,>80 >A5,>B2,>B2,>AF,>B2'    DATA ERROR
213F : TEXT    '>0D,>AC,>A9,>AE,>A5,>80,>B4,>AF,>AF,>80,>AC,>AF,>AE,>A7'
                LINE TOO LONG

```



```

Program :
214D : DATA >0000
214F : DATA >216F
2151 : DATA >08
2152 : TEXT ':TI BASIC:'

```

```

215A : BR GROM@>222B Resets pointer
Cursor and space:
215C : DATA >00,>7C,>7C,>7C,>7C,>7C,>7C,>7C
2164 : DATA >00,>00,>00,>00,>00,>00,>00,>00
VDP register data:
216C : DATA >F0
216D : DATA >0C
216E : DATA >F8

```

```

Begin Basic:
216F : ST @>8373,>88 Substack
2172 : CALL GROM@>27E3 Prepare VDP
2175 : MOVE >000E TO VDP@>02C2 FROM GROM@>20CB TI Basic ready
217C : CLR @>8388
217F : CALL GROM@>4012 PAB existing, close files
2182 : ST @>8334,>FF Pointer to current data
2185 : DST @>836E,>06F8 Value stack pointer
2189 : DST @>8324,@>836E Basis value stack = top character table
218C : DST @>8332,@>8370 End line table
218F : DST @>8330,@>8332 Start line table
2192 : CALL GROM@>222B Set pointer
Entry return Basic:
2195 : AND @>8388,>F7 Clear bit 4
2199 : OR @>8388,>20 Set bit 2
219D : ST @>8374,>05 Keyboard mode
21A0 : SCAN
21A1 : ST @>8373,>88 Pointer substack
21A4 : DST @>8320,>02E2 Screen output start address
21A8 : CLOG @>8388,>01 Numeric mode?
21AC : BS GROM@>21D6 No, jump
21AE : DADD @>8346,@>8348 Step to line number
21B1 : CGE @>8346,>00 Stil positive?
21B4 : BS GROM@>21BC
21B6 : AND @>8388,>FE Clear flag bit
21BA : BR GROM@>21D6
21BC : DCEQ @>8330,@>8332 Exist line table?
21BF : BS GROM@>21C9 No, jump
21C1 : DST @>8344,@>8346 Line number on pointer
21C4 : CALL GROM@>283E Compute line pointer
21C7 : BS GROM@>2399 Found line, jump
21C9 : CALL GROM@>4D00 Scroll 1 line
21CC : DST @>835E,@>8346
21CF : CALL GROM@>2842 Line number on screen
21D2 : DINC @>8320 Screen address +1
21D4 : BR GROM@>21D9 Go on
21D6 : CALL GROM@>4D00 Scroll
21D9 : ST VDP@>02E1,>9E Cursor
21DD : CALL GROM@>2832 Accept input line
21E0 : AND @>8321,>E0 On full line
21E3 : INCT @>8321 +2
21E5 : CALL GROM@>2457 Crunch input line
21E8 : DCZ @>8344 Line number 0?
21EA : BS GROM@>2203 Yes, jump
21EC : CLOG @>8388,>01 NUM mode ?
21F0 : BR GROM@>21FC No, jump
21F2 : CEQ @>8375,>0D Enter key?
21F5 : BS GROM@>21FC Yes, jump
21F7 : CEQ @>8342,>01 Line cleared?
21FA : BS GROM@>2388 Then jump, clear line

```

```

21FC : CALL  GROM@>26B4      Insert line
21FF : BS    GROM@>21A4      New beginning
2201 : BR    GROM@>2192      New beginning, but without clearing of pointers
2203 : CEQ   @>8342,>01      Length of input 1, i.e. only length byte
2206 : BS    GROM@>21A4      Then start again
2208 : CH    VDP@>0320,>09   Token >09?
220C : BS    GROM@>2266      Yes, then arrange line
220E : DST   @>832C,>0321    Fetch 1st byte of line
2212 : XML   >1B
2214 : CASE  VDP@>0320      Depending on token for direct mode
2217 : BR    GROM@>224D      RUN
2219 : BR    GROM@>216F      NEW
221B : BR    GROM@>2268      CONTINUE
221D : BR    GROM@>2245      LIST
221F : BR    GROM@>2342      BYE
2221 : BR    GROM@>228C      NUM
2223 : BR    GROM@>22A7      OLD
2225 : BR    GROM@>22AA      RESEQUENCE
2227 : BR    GROM@>229F      SAVE
2229 : BR    GROM@>2377      EDIT

222B : DST   @>8340,@>8330    Beginning line table = free space
222E : DCEQ  @>8340,@>8370    Equal to highest address ?
2231 : BS    GROM@>2235
2233 : DDEC  @>8340          -1
2235 : DCLR  @>833E          Clear symbol table
2237 : DST   @>8318,@>8340    Free space in end string space
223A : DDEC  @>8318          -1
223C : DST   @>831A,@>8318    Start string space
223F : CLR   @>8343          Option base 0
2241 : DCLR  VDP@>03EC
2244 : RTN

List:
2245 : DCEQ  @>8330,@>8332    Exists line table?
2248 : BS    GROM@>2346      No, jump
224A : B     GROM@>4018      Execute list
Run:
224D : DCEQ  @>8330,@>8332    Exists line table?
2250 : BS    GROM@>2346      No, jump
2252 : CALL  GROM@>4D10      Run subprogram
2255 : DCLR  @>8344          Clear flag
2257 : CALL  GROM@>4012      Close all open files
225A : INC   @>8344          Flag
225C : DCLR  VDP@>03EC      Entry GROM execution
225F : DST   @>836E,>06F8     Top value stack
2263 : DST   @>8324,@>836E     Basis value stack
2266 : BR    GROM@>2828      Execute prescan
Continue:
2268 : CALL  GROM@>282C      Only spaces in line?
226B : BR    GROM@>266C      No, error
226D : DCZ   VDP@>03EC      Continue flag?
2270 : BS    GROM@>2285      Cleared, error
2272 : AND   @>8388,>DF      Clear bit 2
2276 : CALL  GROM@>4D00      Scroll
2279 : DCLR  @>832E
227B : DEX   VDP@>03EC,@>832E  Restore pointer
227F : ST    @>8344,>FF      Run flag
2282 : B     GROM@>4D0C      Continue
2285 : CALL  GROM@>284E      Error
2288 : DATA >20           Can't continue
2289 : DATA >55
228A : BR    GROM@>21A4

Number:
228C : CALL  GROM@>2840      Fetch details
228F : DST   @>8346,@>8314    Start line

```

```

2292 : DST    @>8348,@>831E  Step
2295 : OR     @>8388,>01      NUM mode
2299 : DST    @>8320,>02E2  Start screen input
229D : BR     GROM@>21BC   Go on
Save:
229F : DCEQ  @>8330,@>8332  Is there any program at all?
22A2 : BS    GROM@>2346   No, error
22A4 : B     GROM@>4014   Execute
Old:
22A7 : B     GROM@>4016
Resequenece:
22AA : DCEQ  @>8330,@>8332  Program existing?
22AD : BS    GROM@>2346   No, error
22AF : CALL  GROM@>2840   Fetch details for line number and step
22B2 : DST    @>834A,@>8332  End line list
22B5 : DSUB  @>834A,@>8330  Minus beginning line list
22B8 : DSRL  @>834A,>0002  Now number of lines (/4)
22BC : DMUL  @>834A,@>831E  * Step
22BF : CZ    @>834B     No overflow, jump
22C1 : BS    GROM@>22CA
22C3 : CALL  GROM@>284E
22C6 : DATA >20        Bad line number
22C7 : DATA >D9
22C8 : BR    GROM@>21A4
22CA : DADD  @>8314,@>834C  Plus start line number
22CD : CARY  Overflow?
22CE : BS    GROM@>22C3   Yes, error
22D0 : CH    @>8314,>7F   Negative?
22D3 : BS    GROM@>22C3   Yes, error
22D5 : CLR   @>8350
22D7 : DST    @>834A,@>8332  End line list
22DA : DINCT @>834A      +2
22DC : CEQ   VDP*>834A,>C7  String?
22E0 : BS    GROM@>22E8
22E2 : CEQ   VDP*>834A,>C8  String?
22E6 : BR    GROM@>22F3   No, jump
22E8 : DINC  @>834A     On length byte
22EA : ST    @>8351,VDP*>834A  Fetch length byte
22EE : DADD  @>834A,@>8350  Behind the string
22F1 : BR    GROM@>231E   Go on
22F3 : CEQ   VDP*>834A,>C9  Token for line number ?
22F7 : BR    GROM@>231E   No, go on
22F9 : DINC  @>834A
22FB : DST    @>834E,@>8314  Highest line number (computed before)
22FE : DST    @>834C,@>8330  Start line list
2301 : DCEQ  VDP*>834C,VDP*>834A  Scan line list for right line
2306 : BS    GROM@>2318
2308 : DSUB  @>834E,@>831E  Right line number for next line
230B : DADD  @>834C,>0004  Next line
230F : DCH   @>834C,@>8332  End of line list reached?
2312 : BR    GROM@>2301
2314 : DST    @>834E,>7FFF   Not found, error value
2318 : DST    VDP*>834A,@>834E  New line number in program line
231C : DINC  @>834A
231E : DINC  @>834A     Next address
2320 : CZ    VDP*>834A     Line end ?
2323 : BR    GROM@>22DC   No, go on
2325 : DCEQ  @>834A,@>8370  Arrived at end of program
2328 : BR    GROM@>22DA   No, go on
232A : DST    @>834A,@>8330  Start line number
232D : DST    @>834C,@>8314  Highest new line number
2330 : DST    VDP*>834A,@>834C  Line number in line list
2334 : DSUB  @>834C,@>831E  Next line number
2337 : DADD  @>834A,>0004  Next entry in line list
233B : DCH   @>834A,@>8332  End of list reached?
233E : BR    GROM@>2330   No, go on

```

```

2340 : BR      GROM@>21A4      Back to Basic
Bye:
2342 : CALL    GROM@>4012      Close files
2345 : EXIT                                Software reset

2346 : CALL    GROM@>4D00      Scroll
2349 : DCLR    @>8344          Clear program flag
234B : CALL    GROM@>284E      Error
234E : DATA    >20           Can't do that
234F : DATA    >BD
2350 : BR      GROM@>267E

2352 : DADD    @>832A,@>8302    Entire length
2355 : DSUB    @>8330,@>832A    Minus beginning of line list
2358 : DCHE    @>8330,>0738    High enough?
235C : BS      GROM@>2368      Yes, return
235E : DADD    @>8330,@>832A    Old value again
2361 : CALL    GROM@>284E      Error
2364 : DATA    >20           Memory full
2365 : DATA    >49
2366 : BR      GROM@>267E
2368 : RTN

2369 : DDEC    @>8306          Next address
236B : DDEC    @>8300
236D : ST      VDP*>8306,VDP*>8300  Shift storage spaces
2372 : DDEC    @>835C          Number
2374 : BR      GROM@>2369      No end, then jump
2376 : RTN

Edit:
2377 : CALL    GROM@>282C      Skip spaces
237A : BS      GROM@>2679      No line number, then error
237C : CALL    GROM@>283C      Fetch line number and convert in integer.
237F : CZ      @>830C          No digits?
2381 : BS      GROM@>2679      Then error
2383 : CALL    GROM@>2856      Space till line end ?
2386 : BR      GROM@>2679      No, error
2388 : DCEQ    @>8330,@>8332    Exists line list ?
238B : BS      GROM@>2346      No, error
238D : CALL    GROM@>283E      Search line in list
2390 : BS      GROM@>2399      Found, go on
2392 : CALL    GROM@>284E      Error
2395 : DATA    >20           Bad line number
2396 : DATA    >D9
2397 : BR      GROM@>267E
2399 : ST      @>8306,>1D      Line width
239C : DST     @>8314,@>832E    Pointer to line in line list
239F : ST      @>8317,>60      Screen offset
23A2 : ST      @>8307,>1C
23A5 : CALL    GROM@>282E      Print the line
23A8 : CH      @>8306,@>8307
23AB : BR      GROM@>23B8
23AD : CALL    GROM@>4D00      Scroll
23B0 : DSUB    @>8320,>0020    Start of screen input
23B4 : DSUB    @>8308,>001C
23B8 : DST     @>835E,@>8320
23BB : AND     @>835F,>E0      Start line
23BE : DADD    @>835E,>007D    Plus maximum length of input
23C2 : DST     @>832A,@>8308
23C5 : DCHE    @>835E,@>832A    Higher then end of screen input ?
23C8 : BS      GROM@>23CE
23CA : DST     @>835E,>02FD    End of screen
23CE : CALL    GROM@>2858      Line editor
23D1 : AND     @>8321,>E0      Input of the whole screen line
23D4 : INCT    @>8321

```

```

23D6 : CLOG @>8388,>01 NUM mode
23DA : BR GROM@>23FA Yes, jump
23DC : CEQ @>8375,>0A Cursor up?
23DF : BR GROM@>23EC No, jump
23E1 : DSUB @>8314,>0004 Next line
23E5 : DCHE @>8314,@>8330 Still over beginning of line list?
23E8 : BR GROM@>23FA No, end with ENTER key
23EA : BR GROM@>23FF
23EC : CEQ @>8375,>0B Cursor down?
23EF : BR GROM@>2403
23F1 : DADD @>8314,>0004 Next line
23F5 : DCH @>8314,@>8332 End of line list
23F8 : BR GROM@>23FF No, go on
23FA : ST @>8375,>0D Trick, simulate ENTER key
23FD : BR GROM@>2403
23FF : DST @>831E,VDP*>8314 Next line number
2403 : CZ @>8360 Flag for change
2405 : BR GROM@>240D Not changed, jump
2407 : CALL GROM@>2457 Crunch line
240A : CALL GROM@>26B4 Insert line
240D : DST @>8344,@>831E Next line number
2410 : CEQ @>8375,>0D ENTER key?
2413 : BR GROM@>238D No, go on in edit mode
2415 : BR GROM@>2195 Otherwise return to Basic

```

Execute Basic program in GROM

```

2417 : FETC @>8330 Fetch line table pointer
2419 : FETC @>8331
241B : FETC @>8332
241D : FETC @>8333
241F : DST @>834A,*>8373 Return address on FAC
2423 : ST @>8373,>8A New set stack
2426 : DST *>8373,@>834A New return address
242A : CALL GROM@>27E5 Prepare VDP with pattern table
242D : DCLR @>834A
242F : DADD VDP*>834A,>6060 Describe screen with Basic offset
2434 : DINCT @>834A
2436 : DCEQ @>834A,>0300
243A : BR GROM@>242F
243C : MOVE >0001 TO REG>01 FROM GROM@>2456 Set VDP register 1
2442 : ST @>8389,>FF Set GROM flag
2446 : DST @>8334,@>8332 Pointer to data element
2449 : DSUB @>8334,>0003
244D : MOVE >0002 TO @>8344 FROM GROM@>0000(@>8334) Fetch 1st line number
2454 : BR GROM@>225C For execution RUN
2456 : DATA >57

```

Crunch input line (Change into token i.e. into Basic format):

```

2457 : CLR @>830C Clear number of digits
2459 : DCLR @>8344 No line number
245B : DST @>8338,>031F Pointer on crunch buffer minus 1
245F : CALL GROM@>282C Skip spaces
2462 : BS GROM@>265B Jump at empty line
2464 : CH @>8342,>39 Character greater than 9?
2467 : BS GROM@>2471 Yes, jump
2469 : CHE @>8342,>30 Character smaller than 0?
246C : BR GROM@>2471 Yes, jump
246E : CALL GROM@>283A Convert line number in hex
2471 : DCLR @>835C
2473 : DCH @>8320,@>832A Start addresss input greater than end address?
2476 : BS GROM@>265B Yes, end
2478 : CALL GROM@>2856 Fetch first character
247B : BS GROM@>265B No more character, end
247D : CLR @>830C Clear number of digits
247F : CHE @>8342,>30 Greater 0?
2482 : BS GROM@>2497 Yes, then jump

```

2484	:	CEQ	@>8342,>2E	Point?
2487	:	BS	GROM@>249C	Yes, then jump
2489	:	CEQ	@>8342,>22	Quotation marks?
248C	:	BR	GROM@>24A1	No, jump
248E	:	CALL	GROM@>2684	Fetch string
2491	:	CALL	GROM@>2830	Fetch next byte
2494	:	B	GROM@>2471	Start again
2497	:	CH	@>8342,>39	Greater 9?
249A	:	BS	GROM@>24AF	Yes, then jump
249C	:	CALL	GROM@>27AF	Crunch string or number
249F	:	BR	GROM@>24F6	
24A1	:	DCLR	@>830C	Length counter
24A3	:	CALL	GROM@>2850	Write byte
24A6	:	DST	@>8302,@>8338	Crunch pointer
24A9	:	CALL	GROM@>2830	Fetch next byte
24Ac	:	B	GROM@>24FE	
24AF	:	CALL	GROM@>2846	Check on symbol name
24B2	:	BR	GROM@>24A1	No, write byte and go on
24B4	:	DCLR	@>830C	Length counter
24B6	:	CALL	GROM@>2850	Byte in VDP
24B9	:	DST	@>8302,@>8338	Crunch pointer
24BC	:	BR	GROM@>24C3	Jump
24BE	:	CALL	GROM@>2850	Byte in VDP
24C1	:	INC	@>830D	
24C3	:	CALL	GROM@>2830	Fetch byte
24C6	:	BS	GROM@>24DA	Line end
24C8	:	CALL	GROM@>2846	Check on symbol name
24CB	:	BS	GROM@>24BE	O.K., go on
24CD	:	CEQ	@>8342,>24	Character \$?
24D0	:	BR	GROM@>24DA	No, jump
24D2	:	CALL	GROM@>2850	Byte in VDP
24D5	:	INC	@>830D	Increase length counter
24D7	:	CALL	GROM@>2830	Fetch byte
24DA	:	CZ	@>830D	Length 0, then if adequate token
24DC	:	BS	GROM@>24F6	Yes, jump
24DE	:	CHE	@>830D,>0A	Greater 10?
24E1	:	BR	GROM@>24FE	No, search token table
24E3	:	CGT	@>830D,>0E	Greater 14?
24E6	:	BR	GROM@>24F6	No, jump
24E8	:	DCLR	@>8344	No line number
24EA	:	CALL	GROM@>284E	Error
24ED	:	DATA	>20	Bad name
24EE	:	DATA	>40	
24EF	:	BR	GROM@>267E	With new beginning
24F1	:	CEQ	@>8301,>02	If character not recognized
24F4	:	BS	GROM@>24E8	then jump error
24F6	:	CZ	@>835C	2 symbols one after the other
24F8	:	BR	GROM@>266C	Incorrect statement
24FA	:	INC	@>835C	1st symbol
24FC	:	BR	GROM@>2473	Go on
24FE	:	DST	@>8300,@>830C	Save symbol length
2501	:	INCT	@>8301	Number of characters
2503	:	SLL	@>830D,>01	Into the table
2506	:	MOVE	>0002 TO @>830C FROM GROM@>285C(@>830C)	Fetch table address
250D	:	DST	@>8306,@>8302	
2510	:	MOVE	@>8300 TO @>834A FROM GROM@>0000(@>830C)	Symbol definiton from tab.
2516	:	CEQ	@>834A,>FF	Table end?
2519	:	BS	GROM@>24F1	Yes, then variable
251B	:	DADD	@>830C,@>8300	Next name
251E	:	ST	@>8304,>4A	FAC
2521	:	CEQ	*>8304,VDP*>8306	Name o.k.?
2526	:	BR	GROM@>250D	No, go on

```

2528 : DINC @>8306 Till all digits
252A : INC @>8304
252C : DCHE @>8338,@>8306 End ?
252F : BS GROM@>2521 No, go on
2531 : DST @>8338,@>8302 Old crunch pointer
2534 : ST @>8302,*>8304 Fetch token
2538 : ST VDP*>8338,@>8302 Token in VDP
253C : CEQ @>8302,>93 Special case DATA
253F : BS GROM@>2604
2541 : CEQ @>8302,>9D Special case CALL
2544 : BS GROM@>263E Jump directly to crunch string
2546 : CEQ @>8302,>9A Special case REM
2549 : BS GROM@>25DF
254B : CH @>8302,>09 Special cases 00 through 09
254E : BR GROM@>25BA
2550 : DST @>835E,>25D5 Address list with token for line number
2554 : BR GROM@>25AA
2556 : CEQ @>835C,@>8302 Is token alright
2559 : BR GROM@>25AA No, search
255B : CEQ @>8302,>B1 Token TO?
255E : BR GROM@>2568 No, jump
2560 : CEQ VDP@>FFFF(@>8338),>85 Token GO?
2566 : BR GROM@>2471 No, go on
2568 : CALL GROM@>2856 Fetch byte
256B : BS GROM@>265B At line end jump
256D : CHE @>8342,>30 Greater or equal 0
2570 : BR GROM@>2471
2572 : CH @>8342,>39 But smaller or equal 9
2575 : BS GROM@>2471 then go on
2577 : DST @>834E,@>8344 Save line number
257A : CALL GROM@>283A Convert in integer
257D : DST @>8344,@>834E Right line number again
2580 : ST @>834E,@>8342
2583 : ST @>8342,>C9 Token for line number
2586 : CALL GROM@>2850 Write byte
2589 : ST @>8342,@>834A
258C : CALL GROM@>2850
258F : ST @>8342,@>834B Line number in VDP RAM
2592 : CALL GROM@>2850
2595 : ST @>8342,@>834E Old byte again
2598 : CEQ @>8342,>2C Character ,?
259B : BR GROM@>2471 No, then go on
259D : ST @>8342,>B3 Token ,
25A0 : CALL GROM@>2850 Write byte
25A3 : CALL GROM@>282C Skip spaces
25A6 : BS GROM@>265B Line end
25A8 : BR GROM@>256D Go on
Scan list of token for line number:
25AA : MOVE >0001 TO @>835C FROM GROM@>0000(@>835E) Fetch token
25B1 : DINC @>835E Next token
25B3 : CEQ @>835C,>FF End ?
25B6 : BR GROM@>2556 No, jump
25B8 : BR GROM@>2471 Go on
25BA : DCEQ @>8338,>0320 Crunch pointer on >0320
25BE : BR GROM@>266C No, syntax error
25C0 : DCZ @>8344 No line number
25C2 : BR GROM@>2346 Scroll with " Can't do that error "
25C4 : CEQ @>8302,>06 Old
25C7 : BS GROM@>2604 Like DATA
25C9 : CEQ @>8302,>08 Save
25CC : BS GROM@>2604 Like DATA
25CE : CEQ @>8302,>03 List
25D1 : BS GROM@>25E9 Special case
25D3 : BR GROM@>265B Finish and return
List of tokens with following line number:
25D5 : DATA >B0,>81,>A1,>B1,>87,>86,>8E,>8F,>94,>FF

```

```

Special case REM:
25DF : CALL GROM@>2850 Write byte
25E2 : CALL GROM@>2830 Fetch next byte
25E5 : BR GROM@>25DF Till end
25E7 : BR GROM@>265B End of line
Special case LIST:
25E9 : CALL GROM@>2856 First character without spaces
25EC : CEQ @>8342,>22 Character "?"
25EF : BR GROM@>2601 No, end
25F1 : CALL GROM@>2684 Fetch string
25F4 : CALL GROM@>282C Skip spaces
25F7 : BS GROM@>265B Jump at line end
25F9 : CEQ @>8342,>3A Character :?
25FC : BR GROM@>266C No, error
25FE : CALL GROM@>282C Skip spaces
2601 : B GROM@>265B
Data:
2604 : ST @>834A,>01 Flag
2607 : DDEC @>8320 Pointer in input line minus 1
2609 : BR GROM@>260E Fetch line

Change input line:
260B : ST @>8300,@>8373 Save substack pointer
260E : CLR @>8311 Length counter
2610 : CALL GROM@>282C Over jump spaces
2613 : BS GROM@>265B
2615 : CEQ @>8342,>2C Point?
2618 : BR GROM@>2624
261A : ST @>8342,>B3 Token ,
261D : CALL GROM@>2850 Write byte
2620 : INC @>8311 Length plus 1
2622 : BR GROM@>2610 Start again
2624 : CEQ @>8342,>22 Quotation mark?
2627 : BR GROM@>262E
2629 : CALL GROM@>2684 Fetch string
262C : BR GROM@>2610 Start again
262E : ST @>8313,>2C Point
2631 : CLR @>8382
2634 : CALL GROM@>2854 Fetch string without ""
2637 : CEQ @>8342,>2C Point?
263A : BS GROM@>2615 Go on
263C : BR GROM@>265B End
263E : CZ @>8342 Line end ?
2640 : BS GROM@>266C Error
2642 : CALL GROM@>2856 First character without spaces
2645 : ST @>834A,>01
2648 : ST @>8313,>28
264B : ST @>8382,>01
264F : CALL GROM@>2854 Fetch string without ""
2652 : BS GROM@>265B
2654 : DDEC @>8320 Beginning input minus 1
2656 : CALL GROM@>282C Skip spaces
2659 : BR GROM@>24A1 Not line end, then go on in crunch routine
265B : CLR @>8342
265D : CALL GROM@>2850 Byte into VDP
2660 : DST @>8302,@>8338 Crunch pointer
2663 : DSUB @>8302,>031F
2667 : ST @>8342,@>8303 Compute length
266A : BR GROM@>2850 Byte in VDP and end

Syntax error routine:
266C : CLOG @>8388,>20 Bit 2 set?
2670 : BR GROM@>2677 Yes, jump
2672 : ST @>8373,@>8300 Save subroutine stack
2675 : BR GROM@>26CE Return with condition bit set

```


2677	:	DCLR	@>8344	Clear run flag
2679	:	CALL	GROM@>284E	Error
267C	:	DATA	>20	Incorrect statement
267D	:	DATA	>2C	
267E	:	DSUB	@>8346,@>8348	
2681	:	B	GROM@>21A1	
2684	:	ST	@>8342,>C7	String token
2687	:	CALL	GROM@>2850	Write
268A	:	CALL	GROM@>26AB	Save crunch pointer for length byte
268D	:	CALL	GROM@>2830	Fetch character
2690	:	BS	GROM@>266C	No more, then jump
2692	:	CEQ	@>8342,>22	Quotation mark
2695	:	BS	GROM@>269E	
2697	:	CALL	GROM@>2852	Increase length byte, write byte and new byte
269A	:	BS	GROM@>266C	
269C	:	BR	GROM@>2692	Till the end of strings
269E	:	CALL	GROM@>2830	Fetch character
26A1	:	BS	GROM@>26AA	Till line end jump
26A3	:	CEQ	@>8342,>22	Another quotation mark?
26A6	:	BS	GROM@>2697	Yes, then go on in string
26A8	:	DDEC	@>8320	
26AA	:	RTN		
26AB	:	DINC	@>8338	Crunch pointer +1
26AD	:	DST	@>8302,@>8338	Save crunch pointer
26B0	:	CLR	VDP*>8338	Clear position in VDP
26B3	:	RTN		
Insert line in program:				
26B4	:	CALL	GROM@>4012	Close open files
26B7	:	CALL	GROM@>222B	Set pointer new
26BA	:	CLR	@>8302	
26BC	:	ST	@>8303,@>8342	Length byte on >8302 (word)
26BF	:	CEQ	@>8342,>01	Length 0?
26C2	:	BR	GROM@>2703	No, jump
26C4	:	CLOG	@>8388,>01	NUM mode?
26C8	:	BS	GROM@>26D2	No, jump
26CA	:	AND	@>8388,>FE	Clear NUM mode
26CE	:	CEQ	@>8300,@>8300	
26D1	:	RTNC		
26D2	:	DCEQ	@>8330,@>8332	No line list exists?
26D5	:	BS	GROM@>26CE	No,return with condition bit set
26D7	:	CALL	GROM@>283E	Search line number
26DA	:	BR	GROM@>26CE	Not found, end with condition bit set
26DC	:	CALL	GROM@>2838	Clear line
26DF	:	DST	@>8306,@>832E	Pointer in line list
26E2	:	DINC	@>8306	+1
26E4	:	DDECT	@>832E	-2
26E6	:	DST	@>8300,@>832E	Lower address
26E9	:	DDEC	@>8300	
26EB	:	DSUB	@>832E,@>8330	Number
26EE	:	DCZ	@>832E	0, last line?
26F0	:	BS	GROM@>26F8	
26F2	:	DST	@>835C,@>832E	Number
26F5	:	CALL	GROM@>236D	Shift part of line list
26F8	:	DADD	@>8330,>0004	New start line list
26FC	:	DCH	@>8330,@>8370	Higher than top of memory?
26FF	:	BS	GROM@>218C	Doesn't go, reset pointer to line list
2701	:	BR	GROM@>222B	Return with new pointer
2703	:	DST	@>8304,@>8344	Line number
2706	:	DST	@>8306,@>8332	End line list
2709	:	DST	@>832E,@>8332	End line list

```

270C : DCEQ @>8330,@>8332 Does line list not exist ?
270F : BS GROM@>2749 Then jump
2711 : DINC @>832E
2713 : DSUB @>832E,>0004
2717 : DCEQ @>8344,VDP*>832E Is there a line number?
271B : BS GROM@>278A Yes, jump
271D : DST @>832A,>0004 Does not influence status byte
2721 : H Test on H bit
2722 : BR GROM@>272B Line number does not exist
2724 : DCEQ @>832E,@>8330 Reached end of line list?
2727 : BS GROM@>274D Yes, line belongs to end
2729 : BR GROM@>2713 Go on searching
272B : DST @>835C,@>832E Pointer on entry point in line list
272E : DADD @>835C,>0004 +4
2732 : DSUB @>835C,@>8330 Number
2735 : DST @>8316,@>8330 Old pointer beginning of line list
2738 : CALL GROM@>2352 Is there enough space in memory?
273B : DADD @>8330,@>8302 Start address
273E : MOVE @>835C TO VDP*>8330 FROM VDP*>8316 Shift part of line list
2744 : DST @>8306,@>8332 End line list
2747 : BR GROM@>2756
If no program exists till now:
2749 : DST @>832A,>0003
274D : CALL GROM@>2352 Is there enough space in memory?
2750 : DADD @>8330,@>8302 New start line list (old value stil on >8316)
2753 : DST @>832E,@>8330
2756 : DSUB @>8306,@>8302 Minus length
2759 : DINC @>8306 Pointer to new line
275B : MOVE >0004 TO VDP*>832E FROM @>8304 Entry in line list
2761 : DST @>835C,@>8332 End line list
2764 : DSUB @>835C,@>8330 Minus beginning line list
2767 : DINC @>835C +1
2769 : DST @>8316,@>8330 Beginning line list
276C : DINC @>8302 Length plus 1
276E : DSUB @>8330,@>8302 New start line list
2771 : DSUB @>8332,@>8302 New end of line list
2774 : MOVE @>835C TO VDP*>8330 FROM VDP*>8316 Shift line list
277A : DDEC @>8302 Number minus 1
277C : ST VDP@>FFFF(@>8306),@>8303 Write length byte in program part
2782 : MOVE @>8302 TO VDP*>8306 FROM VDP@>0320 Line in program
2788 : BR GROM@>222B Return with new pointer
If line already exists:
278A : DST @>835C,VDP@>0002(@>832E) Pointer to line
278F : AND @>835C,>7F Bit 0 reset
2792 : DDEC @>835C -!
2794 : CLR @>8308
2796 : ST @>8309,VDP*>835C Length
279A : DST @>832A,@>8308
279D : DNEG @>832A Will be cleared
279F : CALL GROM@>2352 Is there enough space in memory?
27A2 : DADD @>8330,@>832A
27A5 : CALL GROM@>2838 Clear line
27A8 : DDECT @>832E Reset pointer again
27AA : DST @>8306,@>8332
27AD : BR GROM@>2756 Go on with inserting of new line

Crunch number
27AF : DINC @>8338
27B1 : ST VDP*>8338,>C8 String token
27B5 : CALL GROM@>26AB Pointer for length byte
27B8 : CALL GROM@>27CB Fetch number
27BB : CEQ @>8342,>2E Point?
27BE : BR GROM@>27C3
27C0 : CALL GROM@>27D5 Go on
27C3 : CEQ @>8342,>45 E?
27C6 : BR GROM@>27E2 No, end

```

```

27C8 : CALL  GROM@>2852      Write byte and increase pointer
27CB : CEQ    @>8342,>2B      Plus
27CE : BS     GROM@>27D5
27D0 : CEQ    @>8342,>2D      Minus
27D3 : BR     GROM@>27D8
27D5 : CALL  GROM@>2852      Write byte, next byte
27D8 : CHE    @>8342,>30      Greater or equal 0 ?
27DB : BR     GROM@>27E2
27DD : CHE    @>8342,>3A      Greater or equal : ?
27E0 : BR     GROM@>27D5
27E2 : RTN

```

Set VDP tables

```

27E3 : ALL    >80           Clear screen
27E5 : DST    @>83C0,>3567   Random number seed
27EA : MOVE   >0010 TO VDP@>03F0 FROM GROM@>215C  Cursor and space
27F1 : DST    @>834A,>0400   Capital letters
27F5 : CALL  GROM@>0018
27F8 : DST    @>834A,>0600   Lower case
27FC : CALL  GROM@>004A
27FF : BACK   >07           Background color
2801 : DST    VDP@>0300,>D000 Sprite end decimal 208
2806 : MOVE   >000D TO VDP@>0302 FROM VDP@>0301  Clear under color table
280D : ST     VDP@>030F,>17   Load color table
2811 : MOVE   >0010 TO VDP@>0310 FROM VDP@>030F
2818 : MOVE   >0003 TO REG>02 FROM GROM@>216C  Load color table
281E : RTN

```

```

281F : DATA  >00,>00,>00,>00
2823 : DATA  >00,>00,>00,>00,>00

```

```

2828 : BR     GROM@>2FFF      Prescan
282A : BR     GROM@>2F43      Bad line number
282C : BR     GROM@>2C75      Skip spaces
282E : BR     GROM@>2DFA      List one line on screen
2830 : BR     GROM@>2CA6      Fetch character from input line
2832 : BR     GROM@>2A42      Line editor
2834 : BR     GROM@>2C36      Starts NUM
2836 : BR     GROM@>2FC4      Finds first token of a line in VDP
2838 : BR     GROM@>2BD6      Clears lines
283A : BR     GROM@>2F12      Search certain line number
283C : BR     GROM@>2EF9      Line number from ASCII in hex
283E : BR     GROM@>2F5D      Finds program line in VDP
2840 : BR     GROM@>2C2B      Starts NUM with error values
2842 : BR     GROM@>2FAF      Line number from hex in ASCII with screen
                                projection.
2844 : BR     GROM@>3493      Fetch space in VDP for string etc.
2846 : BR     GROM@>3450      Check, if variable name
2848 : BR     GROM@>31E5      Sets variable in symbol table
284A : BR     GROM@>322B      Sets dummies in symbol table
284C : BR     GROM@>2D24      Warning
284E : BR     GROM@>2D99      Error
2850 : BR     GROM@>2C84      Stores actual byte in VDP
2852 : BR     GROM@>2CA0      Increase VDP pointer for next character
2854 : BR     GROM@>2CC0      Fetch string without ""
2856 : BR     GROM@>2C7A      Fetch first character from input line without
                                space
2858 : BR     GROM@>2A49      2nd entry point line editor
285A : BR     GROM@>2A4F      3rd entry point line editor

```

Token table pointer:

```

285C : DATA  >2870      1 byte
285E : DATA  >288F      2
2860 : DATA  >289C      3
2862 : DATA  >291D      4
2864 : DATA  >2973      5

```

2866 : DATA >299E 6
 2868 : DATA >29D0 7
 286A : DATA >29F1 8
 286C : DATA >2A16 9
 286E : DATA >2A2B 10

1 byte:

2870 : TEXT ' :) : , >B6 ,
 : (: , >B7 ,
 : & : , >B8 ,
 : ^ : , >C5 ,
 : = : , >BE ,
 : * : , >C3 ,
 : / : , >C4 ,
 : + : , >C1 ,
 : - : , >C2 ,
 : < : , >BF ,
 : > : , >C0 ,
 : : : , >B5 ,
 : ; : , >B4 ,
 : # : , >FD ,
 : , : , >B3 '

288E : DATA >FF

2 bytes:

288F : TEXT ' : GO : , >85 ,
 : IF : , >84 ,
 : ON : , >9B ,
 : TO : , >B1 '

289B : DATA >FF

3 bytes:

289C : TEXT ' : DEF : , >89 ,
 : DIM : , >8A ,
 : END : , >8B ,
 : EOF : , >CA ,
 : FOR : , >8C ,
 : LET : , >8D ,
 : REM : , >9A ,
 : SUB : , >A1 ,
 : TAB : , >FC ,
 : ABS : , >CB ,
 : ATN : , >CC ,
 : COS : , >CD ,
 : EXP : , >CE ,
 : INT : , >CF ,
 : LOG : , >D0 ,
 : RND : , >D7 ,
 : SGN : , >D1 ,
 : SIN : , >D2 ,
 : SQR : , >D3 ,
 : TAN : , >D4 ,
 : LEN : , >D5 ,
 : POS : , >D9 ,
 : VAL : , >DA ,
 : ASC : , >DC ,
 : REC : , >DE ,
 : NEW : , >01 ,
 : RUN : , >00 ,
 : CON : , >02 ,
 : NUM : , >05 ,
 : RES : , >07 ,
 : BYE : , >04 ,
 : OLD : , >06 '

291C : DATA >FF

4 bytes:

291D : TEXT ' : BASE : , >F1 ,
 : DATA : , >93 ,

```

:EDIT: ,>09,
:ELSE: ,>81,
:GOTO: ,>86,
:NEXT: ,>96,
:READ: ,>97,
:STEP: ,>B2,
:STOP: ,>98,
:THEN: ,>B0,
:CHR$: ,>D6,
:SEG$: ,>D8,
:STR$: ,>DB,
:LIST: ,>03,
:SAVE: ,>08,
:CALL: ,>9D,
:OPEN: ,>9F'
2972 : DATA >FF
5 bytes:
2973 : TEXT ' :BREAK: ,>8E,
:GOSUB: ,>87,
:FIXED: ,>FA,
:INPUT: ,>92,
:PRINT: ,>9C,
:TRACE: ,>90,
:CLOSE: ,>A0'
299D : DATA >FF
6 bytes:
299E : TEXT ' :OPTION: ,>9E,
:RETURN: ,>88,
:NUMBER: ,>05,
:OUTPUT: ,>F7,
:APPEND: ,>F9,
:UPDATE: ,>F8,
:DELETE: ,>99'
29CF : DATA >FF
7 bytes:
29D0 : TEXT ' :UNTRACE: ,>91,
:UNBREAK: ,>8F,
:RESTORE: ,>94,
:DISPLAY: ,>A2'
29F0 : DATA >FF
8 bytes:
29F1 : TEXT ' :CONTINUE: ,>02,
:VARIABLE: ,>F3,
:INTERNAL: ,>F5,
:RELATIVE: ,>F4'
2A15 : DATA >FF
9 bytes:
2A16 : TEXT ' :RANDOMIZE: ,>95,
:PERMANENT: ,>FB'
2A2A : DATA >FF
10 bytes:
2A2B : TEXT ' :SEQUENTIAL: ,>F6,
:RESEQUENCE: ,>07,
2A41 : DATA >FF'

```

Line editor:

```

2A42 : DST @>835E,>035D End of input
2A46 : DST @>832A,@>8320 Start of input
2A49 : ST @>8360,>01 Flag
2A4C : DST @>8361,@>8320 Start of input, cursor position
2A4F : CLR @>830D Repeat counter
2A51 : CLR @>8363
2A53 : ST @>8301,>7E Cursor character
2A56 : EX VDP*>8361,@>8301 On screen
2A5A : CLR @>8379 Cursor counter
2A5C : SCAN Keyboard scanning

```

2A5D	:	BS	GROM@>2A78	New key, jump
2A5F	:	INC	@>830D	Repeat counter
2A61	:	CEQ	@>8375,>FF	No key?
2A64	:	BS	GROM@>2A71	Yes, jump
2A66	:	CHE	@>830D,>FE	Repeat counter high enough?
2A69	:	BR	GROM@>2A71	No, go on
2A6B	:	SUB	@>830D,>1E	Minus
2A6E	:	B	GROM@>2A7A	Treat as key input
2A71	:	CH	@>8379,>0E	Cursor counter high?
2A74	:	BR	GROM@>2A5C	No, jump to keyboard scanning
2A76	:	BR	GROM@>2A56	Yes, keyboard scanning with change character
2A78	:	CLR	@>830D	Repeat counter 0
2A7A	:	CEQ	@>8301,>7E	Cursor character on screen?
2A7D	:	BS	GROM@>2A83	No, jump
2A7F	:	EX	VDP*>8361,@>8301	Change character
2A83	:	CHE	@>8375,>20	FCTN key?
2A86	:	BS	GROM@>2B41	No, jump
2A88	:	CEQ	@>8375,>02	Clear key?
2A8B	:	BR	GROM@>2AA1	
2A8D	:	AND	@>8388,>FE	Clear bit 7
2A91	:	CZ	@>8389	GROM?
2A94	:	BR	GROM@>2A56	Yes, go on
2A96	:	CLOG	@>8388,>20	Bit 2 set?
2A9A	:	BR	GROM@>2A9F	Yes, jump
2A9C	:	B	GROM@>4D0E	Program interrupt
2A9F	:	BR	GROM@>2018	Reset cursor position and end
2AA1	:	CEQ	@>8375,>08	Cursor to the right
2AA4	:	BS	GROM@>2BC3	Yes, jump
2AA6	:	CEQ	@>8375,>09	Cursor to the right
2AA9	:	BS	GROM@>2BBF	Yes jump
2AAB	:	CEQ	@>8375,>04	Insert key
2AAE	:	BR	GROM@>2AB3	No, jump
2AB0	:	ST	@>8363,>01	Flag for insert mode
2AB3	:	CEQ	@>8375,>03	Delete key?
2AB6	:	BR	GROM@>2B08	No, jump
2AB8	:	CLR	@>8360	0
2ABA	:	DCEQ	@>832A,@>8361	Cursor at the end?
2ABD	:	BS	GROM@>2B02	Yes end
2ABF	:	CEQ	VDP*>832A,>7F	Edge?
2AC3	:	BR	GROM@>2AC7	
2AC5	:	DDEC	@>832A	Minus 1
2AC7	:	DST	@>835C,@>832A	End of input
2ACA	:	DSUB	@>835C,@>8361	Minus actual cursor position
2ACD	:	MOVE	@>835C TO VDP*>8361 FROM VDP@>0001(@>8361)	All 1 up
2AD4	:	DST	@>835C,@>8361	Actual cursor position
2AD7	:	AND	@>835D,>FC	Test on edge
2ADA	:	OR	@>835D,>1D	
2ADD	:	DCHE	@>835C,@>832A	Already higher than end?
2AE0	:	BS	GROM@>2AEE	
2AE2	:	EX	VDP@>0004(@>835C),VDP*>835C	Exchange edge
2AE8	:	DADD	@>835C,>0020	Next line
2AEC	:	BR	GROM@>2ADD	
2AEE	:	DDEC	@>832A	End minus 1 (is cleared)
2AF0	:	CEQ	VDP*>832A,>7F	Edge?
2AF4	:	BR	GROM@>2AFA	
2AF6	:	DSUB	@>832A,>0004	Skip edge
2AFA	:	CEQ	VDP*>832A,>80	Space at the end
2AFE	:	BS	GROM@>2A51	Yes, go on
2B00	:	DINC	@>832A	+1
2B02	:	ST	VDP*>832A,>80	Set space
2B06	:	BR	GROM@>2A51	Delete end
2B08	:	CEQ	@>8375,>07	Erase key?
2B0B	:	BR	GROM@>2B24	No, jump
2B0D	:	CEQ	VDP*>832A,>7F	Edge?
2B11	:	BS	GROM@>2B17	Yes, skip
2B13	:	ST	VDP*>832A,>80	Set space

2B17	:	DDEC	@>832A	End minus 1
2B19	:	DCHE	@>832A,@>8320	Till the beginning
2B1C	:	BS	GROM@>2B0D	Go on
2B1E	:	DINC	@>832A	New end
2B20	:	CLR	@>8360	
2B22	:	BR	GROM@>2A4C	Start again
2B24	:	CEQ	@>8375,>0D	Enter key
2B27	:	BS	GROM@>2B33	Yes, end
2B29	:	CEQ	@>8375,>0B	Cursor up
2B2C	:	BS	GROM@>2B33	Yes, end
2B2E	:	CEQ	@>8375,>0A	Cursor down
2B31	:	BR	GROM@>2A56	No, more input
2B33	:	DCEQ	@>832A,@>835E	End equal maximum?
2B36	:	BR	GROM@>2B40	No, end
2B38	:	CEQ	VDP*>832A,>80	Yes, write more space
2B3C	:	BS	GROM@>2B40	
2B3E	:	DINC	@>832A	Increase end for 1 point
2B40	:	RTN		End
2B41	:	CZ	@>8363	Insert mode?
2B43	:	BS	GROM@>2B7D	No, jump
2B45	:	DCEQ	@>832A,@>835E	Maximum end reached ?
2B48	:	BS	GROM@>2B54	Yes, jump
2B4A	:	CEQ	VDP*>832A,>7F	Edge?
2B4E	:	BR	GROM@>2B54	No, jump
2B50	:	DADD	@>832A,>0004	Skip edge
2B54	:	DST	@>835C,@>832A	
2B57	:	DCH	@>835C,@>8361	Cursor at the end
2B5A	:	BR	GROM@>2B76	No, jump
2B5C	:	DDEC	@>835C	
2B5E	:	ST	VDP@>0001(@>835C),VDP*>835C	Shift text 1 to the right
2B64	:	CEQ	VDP*>835C,>7F	Edge?
2B68	:	BR	GROM@>2B74	No, jump
2B6A	:	DSUB	@>835C,>0004	Skip edge
2B6E	:	ST	VDP@>0005(@>835C),VDP*>835C	Shift edge
2B74	:	BR	GROM@>2B57	Till beginning
2B76	:	DCHE	@>832A,@>835E	Maximum end reached?
2B79	:	BS	GROM@>2B7D	Yes, jump
2B7B	:	DINC	@>832A	End plus 1
2B7D	:	ADD	@>8375,>60	Add screen offset
2B80	:	ST	VDP*>8361,@>8375	Write character on screen
2B84	:	CLR	@>8360	
2B86	:	DCEQ	@>8361,@>835E	Maximum end reached?
2B89	:	BR	GROM@>2B90	No, jump
2B8B	:	CALL	GROM@>0036	Bad tone
2B8E	:	BR	GROM@>2A56	Next entry
2B90	:	DINC	@>8361	Increase cursor position
2B92	:	CEQ	VDP*>8361,>7F	Edge?
2B96	:	BR	GROM@>2B9C	No, jump
2B98	:	DADD	@>8361,>0004	Skip edge
2B9C	:	DCH	@>8361,@>832A	Cursor over end
2B9F	:	BR	GROM@>2BA4	No, jump
2BA1	:	DST	@>832A,@>8361	New end position
2BA4	:	DCHE	@>832A,>02FE	Outside from screen?
2BA8	:	BR	GROM@>2A56	No, next input
2BAA	:	CALL	GROM@>4D00	Scroll
2BAD	:	DSUB	@>832A,>001C	End minus 1 line
2BB1	:	DSUB	@>8320,>0020	Beginning minus 1 line
2BB5	:	DSUB	@>835E,>0020	Maximum minus 1 line
2BB9	:	DSUB	@>8361,>0020	Cursor position minus 1 line
2BBD	:	BR	GROM@>2A56	Furthe input
2BBF	:	CLR	@>8363	Clear insert mode
2BC1	:	BR	GROM@>2B86	Jump, controll if beyond end
2BC3	:	DCH	@>8361,@>8320	At the beginning?
2BC6	:	BR	GROM@>2BD4	Yes, jump
2BC8	:	DDEC	@>8361	Cursor position minus 1

```

2BCA : CEQ   VDP*>8361,>7F   Edge?
2BCE : BR    GROM@>2BD4     No, jump
2BD0 : DSUB  @>8361,>0004    Skip edge
2BD4 : BR    GROM@>2A51     Next input

```

Routine for clearing lines in VDP RAM:

```

2BD6 : DINCT @>832E           Pointer on line
2BD8 : DST   @>835C,VDP*>832E Fetch pointer to line
2BDC : AND   @>835C,>7F       Clear bit 0
2BDF : DDEC  @>835C           -1, points now to length byte
2BE1 : ST    @>8309,VDP*>835C Length byte
2BE5 : INC   @>8309           +1
2BE7 : CLR   @>8308           Length is word
2BE9 : DST   @>8300,@>8332    End line table
2BEC : DINC  @>8300           +1
2BEE : DDECT @>8300           -2, indicates to pointer in first line
2BF0 : DST   @>8306,VDP*>8300 Pointer on >8306
2BF4 : AND   @>8306,>7F       Clear bit 0
2BF7 : DDEC  @>8306           -1
2BF9 : DCH   @>835C,@>8306    Is line above questioned line?
2BFC : BR    GROM@>2C02     No, jump
2BFE : DADD  VDP*>8300,@>8308 Correct pointer in line list
2C02 : DDECT @>8300           Next line
2C04 : DCEQ  @>8330,@>8300    Already arrived at end of line list
2C07 : BR    GROM@>2BEE     No, go on
2C09 : DST   @>8306,@>835C    Pointer to line
2C0C : DST   @>8300,@>8306    Lowest address for shifting
2C0F : DDEC  @>8300           -1
2C11 : DADD  @>8306,@>8308    Plus number of bytes
2C14 : DDEC  @>8306           Now high address
2C16 : DST   @>835C,@>8300    Reserve number of pointer
2C19 : DSUB  @>835C,@>8330    Minus start line table
2C1C : DINC  @>835C           +1
2C1E : CALL  GROM@>201E     Shift VDP RAM
2C21 : DADD  @>8330,@>8308    New pointer to line list
2C24 : DADD  @>8332,@>8308
2C27 : DADD  @>832E,@>8308
2C2A : RTN

```

```

2C2B : DST   @>8314,>0064     Error value line 100
2C2F : DST   @>831E,>000A     Step 10
2C33 : ST    @>8308,>2C       Decimal 44
2C36 : DDEC  @>8320           Start input screen
2C38 : CALL  GROM@>2C75       To next character
2C3B : BS    GROM@>2C2A
2C3D : CALL  GROM@>2EF9       Convert first number in hex
2C40 : CZ    @>830C           All o.k.?
2C42 : BR    GROM@>2C4F
2C44 : CZ    @>8300
2C46 : BR    GROM@>2C4D       Syntax error
2C48 : CALL  GROM@>2C75       To next character
2C4B : BR    GROM@>2C65
2C4D : BR    GROM@>2016       Syntax error
2C4F : DST   @>8314,@>8344    First number on line
2C52 : CZ    @>8300
2C54 : BR    GROM@>2C60
2C56 : CALL  GROM@>2C75       To next character
2C59 : BS    GROM@>2C2A
2C5B : ST    @>830E,@>8309
2C5E : BS    GROM@>2C65
2C60 : CALL  GROM@>2C7A       Skip space
2C63 : BS    GROM@>2C2A
2C65 : CALL  GROM@>2EF9       Convert in hex
2C68 : CZ    @>830C           All o.k.
2C6A : BS    GROM@>2C4D
2C6C : DST   @>831E,@>8344    Number as step or 2nd line number

```



```

2C6F : CZ      @>8300
2C71 : BS      GROM@>2C4D      Syntax error
2C73 : BR      GROM@>2F4F      Return with skip spaces

2C75 : CALL    GROM@>2CA6      Line end?
2C78 : BS      GROM@>2C80      Return condition bit set
2C7A : CEQ     @>8342,>20      Space?
2C7D : BS      GROM@>2C75      Yes, go on
2C7F : RTN

2C80 : CEQ     @>8300,@>8300
2C83 : RTNC                                RTN condition bit set

2C84 : DCH     @>8338,>03BE     Greater than end input buffer
2C88 : BS      GROM@>2C91      Error
2C8A : DINC    @>8338          Crunch pointer
2C8C : ST      VDP*>8338,@>8342 Write byte in crunch buffer
2C90 : RTN

2C91 : CLOG    @>8388,>20      Check flag byte
2C95 : BS      GROM@>2C4D      Syntax error
2C97 : DCLR    @>8344
2C99 : CALL    GROM@>2D99      Error
2C9C : DATA   >21            Line too long
2C9D : DATA   >3F
2C9E : BR      GROM@>2F4D      Reset cursor pointer

2CA0 : INC     VDP*>8302
2CA3 : CALL    GROM@>2C84      Write byte
2CA6 : DCH     @>8320,@>832A    Reached end
2CA9 : BS      GROM@>2C80      Return condition bit set
2CAB : ST      @>8342,VDP*>8320 Fetch byte
2CAF : CEQ     @>8342,>7F      Edge identification
2CB2 : BS      GROM@>2CBA      Yes, jump
2CB4 : SUB     @>8342,>60      Subtract offset
2CB7 : DINC    @>8320          Increase stack pointer
2CB9 : RTN
2CBA : DADD    @>8320,>0004     Skip edge
2CBE : BR      GROM@>2CA6

Crunch string:
2CC0 : DINC    @>8338
2CC2 : ST      VDP*>8338,>C8    String token
2CC6 : CALL    GROM@>2020      Pointer for length byte
2CC9 : DCH     @>8338,>03BE     End ?
2CCD : BR      GROM@>2CD9
2CCF : CALL    GROM@>2C7A      Skip spaces
2CD2 : BS      GROM@>2CEB      At line end jump
2CD4 : CEQ     @>8342,@>8313
2CD7 : BS      GROM@>2CEB
2CD9 : CZ      @>8382          Flag
2CDC : BS      GROM@>2CE1
2CDE : CALL    GROM@>3450      Permitted in symbol name?
2CE1 : CALL    GROM@>2CA0      Crunch byte
2CE4 : BS      GROM@>2CEB      At line end jump
2CE6 : CEQ     @>8342,@>8313
2CE9 : BR      GROM@>2CC9
2CEB : CEQ     VDP*>8338,>20    Space?
2CEF : BR      GROM@>2CFB      No, end
2CF1 : DDEC    @>8338          Eliminate spaces
2CF3 : DEC     VDP*>8302      Length byte minus 1
2CF6 : DCEQ    @>8338,@>8302    0?
2CF9 : BR      GROM@>2CEB
2CFB : RTN

2CFC : DCLR    @>834A

```

```

2CFE : XML    >1B           Fetch byte
2D00 : SUB    @>8342,>30     In Integer
2D03 : CHE    @>8342,>0A     Greater or equal 10?
2D06 : BS     GROM@>2D20     Yes, end
2D08 : DMUL   @>834A,>000A   *10
2D0C : DCZ    @>834A         To big?
2D0E : BR     GROM@>2C80
2D10 : ST     @>834B,@>8342   Plus actual byte
2D13 : DADD   @>834A,@>834C
2D16 : CARY                   Overflow?
2D17 : BS     GROM@>2C80     Yes, error
2D19 : CGE    @>834A,>00     Negative?
2D1C : BR     GROM@>2C80     Yes, error
2D1E : BR     GROM@>2CFE     Go on
2D20 : ADD    @>8342,>30     Repair
2D23 : RTN

```

Print warning:

```

2D24 : CALL   GROM@>4D00     Scroll
2D27 : MOVE   >000A TO VDP@>02E2 FROM GROM@>2022   Message * WARNING
2D2E : FETC   @>8376         Fetch pointer to text
2D30 : FETC   @>8377
2D32 : CALL   GROM@>4D00     Scroll
2D35 : CLR    @>8374
2D37 : CALL   GROM@>0036     Bad tone
2D3A : DST    @>8320,>02E4     Cursor position
2D3E : MOVE   >0001 TO @>835D FROM GROM@>0000(@>8376)  Fetch length message
2D45 : CLR    @>835C
2D47 : MOVE   @>835C TO VDP*>8320 FROM GROM@>0001(@>8376)  Text on screen
2D4E : DADD   @>8320,@>835C   New cursor position
2D51 : DCEQ   @>8376,>2113     I/O error?
2D55 : BR     GROM@>2D6C     No, jump
2D57 : DINC   @>8320
2D59 : DST    @>835F,VDP@>0004(@>8304)Fetch op code
2D5E : CLR    @>835E
2D60 : CALL   GROM@>2FAF     Convert into ASCII and print
2D63 : ST     @>835F,@>8360     Fetch error code
2D66 : SRL    @>835F,>05       In number-->
2D69 : CALL   GROM@>2FAF     In ASCII and print
2D6C : CLOG   @>8388,>20       Bit 2 set?
2D70 : BR     GROM@>2D96     Yes, jump
2D72 : DCZ    @>8344         RUN flag?
2D74 : BS     GROM@>2D96     No, jump
2D76 : DCH    @>8320,>02F5     New line?
2D7A : BR     GROM@>2D83
2D7C : CALL   GROM@>4D00     Scroll
2D7F : DST    @>8320,>02E4     Cursor again beginning of line
2D83 : DST    VDP@>0001(@>8320),>A9AE   Text "IN"
2D89 : DADD   @>8320,>0004
2D8D : ST     @>8376,@>8342     Save actual Basic byte
2D90 : CALL   GROM@>4D0A     Line number in ASCII and print
2D93 : ST     @>8342,@>8376     Basic byte back
2D96 : B      GROM@>4D00     Scroll with return

```

Error:

```

2D99 : FETC   @>8376         Fetch text pointer
2D9B : FETC   @>8377
2D9D : DCEQ   @>833E,>0320     Symbol table on >0320?
2DA1 : BR     GROM@>2DA7
2DA3 : DST    @>833E,VDP@>0322
2DA7 : CLOG   @>8388,>20       Bit 2 set?
2DAB : BR     GROM@>2DB0     Yes, jump
2DAD : ST     @>8373,>8A       New substack pointer
2DB0 : CALL   GROM@>4D00     Scroll
2DB3 : ST     VDP@>02E2,>8A     Text "*"
2DB7 : CALL   GROM@>2D35     Print message

```

```

2DBA : CLOG    @>8388,>20      Bit 2 set?
2DBE : BR      GROM@>2D23     Yes, jump
2DC0 : CALL    GROM@>201C     Reset character sets and color table
2DC3 : DCZ     @>8344        Run flag?
2DC5 : BS      GROM@>2DCE     No, jump
2DC7 : DCLR    @>8344        Clear RUN flag
2DC9 : CALL    GROM@>4012     Close files
2DCC : DDEC    @>8344        Set RUN flag again
2DCE : CZ      @>8389        GROM?
2DD1 : BR      GROM@>2C80     Yes, jump return condition bit set
2DD3 : DCH     @>836E,@>8324  Value stack
2DD6 : BR      GROM@>2DF1
2DD8 : CGT     VDP@>0002(@>836E),>65  Greater string tag
2DDD : BS      GROM@>2DE3
2DDF : XML     >18           VPOP
2DE1 : BR      GROM@>2DD3
2DE3 : CEQ     VDP@>0002(@>836E),>68  User defined function?
2DE8 : BR      GROM@>2DF1     No, end
2DEA : DCLR    @>834E
2DEC : CALL    GROM@>4D14     Clear entry
2DEF : BR      GROM@>2DD3     Go on
2DF1 : DCZ     @>8344        RUN flag?
2DF3 : BS      GROM@>2DF8     No, jump
2DF5 : DST     @>836E,@>8324  New top value stack
2DF8 : BR      GROM@>2012     Return Basic

2DFA : CALL    GROM@>401A     Write data block
2DFD : DST     @>835E,VDP*>8314  Fetch line number
2E01 : CALL    GROM@>2F9A     Convert number into ASCII and print
2E04 : DST     @>8320,@>8308   Pointer at beginning reserve output
2E07 : DINC    @>8320        Plus 1
2E09 : DST     @>832C,VDP@>0002(@>8314)  Begin text pointer
2E0E : AND     @>832C,>7F     Bit 0 reset
2E11 : DST     @>834C,>0020   Space
2E15 : CALL    GROM@>2F58     Fetch byte
2E18 : BS      GROM@>2F11     End at line end
2E1A : CZ      @>834D        Print byte on >834D
2E1C : BS      GROM@>2E27
2E1E : EX      @>834D,@>8342   Exchange byte
2E21 : CALL    GROM@>2FE3     Print byte
2E24 : EX      @>834D,@>8342   Byte back again
2E27 : CLR     @>834D        O.k. nothing more
2E29 : CHE     @>8342,>B3     Higher token ,?
2E2C : BR      GROM@>2E33     No, jump
2E2E : CHE     @>8342,>C8     Higher token string
2E31 : BR      GROM@>2E4B     No, jump
2E33 : ST      @>834D,>20     Space
2E36 : CZ      @>834C
2E38 : BS      GROM@>2E4B
2E3A : CEQ     @>8354,>20     Space
2E3D : BS      GROM@>2E4B
2E3F : ST      @>834C,@>8342   Byte on >834C
2E42 : ST      @>8342,>20     Space
2E45 : CALL    GROM@>2FE3     Print byte
2E48 : ST      @>8342,@>834C   And old value again
2E4B : EX      @>834C,@>834D
2E4E : CLOG    @>8342,>80     Token?
2E51 : BR      GROM@>2E5F     Yes, jump
2E53 : CALL    GROM@>2F55     Write data block
2E56 : BS      GROM@>2F11     End
2E58 : CLOG    @>8342,>80     Token?
2E5B : BS      GROM@>2E53     No, jump
2E5D : BR      GROM@>2E27     New start
2E5F : CEQ     @>8342,>C8     String?
2E62 : BS      GROM@>2E6C
2E64 : CEQ     @>8342,>C7     String in ""

```

```

2E67 : BR      GROM@>2E96
2E69 : CALL    GROM@>2FE0      Print quotation mark
2E6C : XML     >1B           Fetch byte
2E6E : ST      @>834A,@>8342  Save on FAC
2E71 : CZ      @>834A       0?
2E73 : BS      GROM@>2E8A    Yes, jump
2E75 : XML     >1B           Fetch byte
2E77 : CZ      @>834C
2E79 : BR      GROM@>2E83
2E7B : CEQ     @>8342,>22    Quotation mark, then twice
2E7E : BR      GROM@>2E83
2E80 : CALL    GROM@>2FE3    Print byte
2E83 : CALL    GROM@>2FE3    Print byte
2E86 : DEC     @>834A
2E88 : BR      GROM@>2E71    Till string end
2E8A : CZ      @>834C
2E8C : BR      GROM@>2EF5
2E8E : CALL    GROM@>2FE0    Print quotation mark
2E91 : ST      @>834C,>20    Space
2E94 : BR      GROM@>2EF5
2E96 : CEQ     @>8342,>C9    Token for line number ?
2E99 : BR      GROM@>2EAA
2E9B : XML     >1B           Fetch 1st byte
2E9D : ST      @>835E,@>8342
2EA0 : XML     >1B           Fetch 2nd byte
2EA2 : ST      @>835F,@>8342  Word complete
2EA5 : CALL    GROM@>2F9A    Convert integer into ASCII and print
2EA8 : BR      GROM@>2EF5
2EAA : DCLR    @>834A       1st table
2EAC : DST     @>834E,>0001   Length of token
2EB0 : MOVE    >0002 TO @>8352 FROM GROM@>285C(@>834A)  Fetch pointer to table
2EB7 : DADD    @>8352,@>834E  Beginning with token
2EBA : MOVE    >0002 TO @>8350 FROM GROM@>0000(@>8352)  Fetch first token
2EC1 : CEQ     @>8350,@>8342  Right?
2EC4 : BS      GROM@>2ED3
2EC6 : DINC    @>8352
2EC8 : CEQ     @>8351,>FF     End of list ?
2ECB : BR      GROM@>2EB7    Next token
2ECD : DINCT   @>834A       Next table
2ECF : DINC    @>834E       Length greater
2ED1 : BR      GROM@>2EB0
2ED3 : DSUB    @>8352,@>834E  Pointer minus length
2ED6 : ST      @>834A,@>8342
2ED9 : MOVE    >0001 TO @>8342 FROM GROM@>0000(@>8352)  Fetch byte
2EE0 : CALL    GROM@>2FE3    and write byte
2EE3 : DINC    @>8352       Increase pointer
2EE5 : DEC     @>834F       Length minus 1
2EE7 : BR      GROM@>2ED9    Till length 0
2EE9 : CHE     @>834A,>B3    Higher or equal token ,?
2EEC : BR      GROM@>2E11    No, start again
2EEE : CEQ     @>834A,>FD    Token #?
2EF1 : BR      GROM@>2EF5    No, jump
2EF3 : CLR     @>834C
2EF5 : CLR     @>834D
2EF7 : BR      GROM@>2E15    New start again

2EF9 : CLR     @>8300
2EFB : CLR     @>830C
2EFD : CHE     @>8342,>30    ASCII 0
2F00 : BR      GROM@>2F0A
2F02 : CGT     @>8342,>39    ASCII 9
2F05 : BS      GROM@>2F0A    No number
2F07 : CALL    GROM@>2F12    Convert in integer
2F0A : CEQ     @>8342,@>8308
2F0D : BS      GROM@>2F11
2F0F : INC     @>8300       Flag digit

```

```

2F11 : RTN

2F12 : DCLR  @>834A
2F14 : ST    @>8301,@>8342  Byte on >8301
2F17 : SUB   @>8301,>30     Integer
2F1A : CHE   @>8301,>0A     Greater or equal 10?
2F1D : BS    GROM@>2F3B    End
2F1F : DMUL  @>834A,>000A   *10
2F23 : CZ    @>834B        No overflow
2F25 : BR    GROM@>2F43    Error
2F27 : DST   @>834A,@>834C  New start again
2F2A : CLR   @>8300
2F2C : DADD  @>834A,@>8300
2F2F : INC   @>830C
2F31 : CGE   @>834A,>00     Negative?
2F34 : BR    GROM@>2F43    Error
2F36 : CALL  GROM@>2CA6    Next byte
2F39 : BR    GROM@>2F14    and new start again
2F3B : DST   @>8344,@>834A  Line number
2F3E : DCZ   @>834A        0?
2F40 : BS    GROM@>2F43    Error
2F42 : RTN

2F43 : CALL  GROM@>4D00     Scroll
2F46 : DCLR  @>8344        No line number
2F48 : CALL  GROM@>2D99     Error
2F4B : DATA >20          Bad line number
2F4C : DATA >D9

2F4D : BR    GROM@>2018     Reset cursor position
2F4F : CALL  GROM@>2C75     Line end, skip spaces
2F52 : BR    GROM@>2C4D     Syntax error
2F54 : RTN

2F55 : CALL  GROM@>2FE3     Write data block
2F58 : XML   >1B           Fetch byte
2F5A : CZ    @>8342        Line end?
2F5C : RTNC

Find line number in line table:
2F5D : DST   @>832E,@>8332  End line table on actual line pointer
2F60 : DSUB  @>832E,>0003   -3, on line number
2F64 : DCHE  VDP*>832E,@>8344 Line number greater?
2F68 : BR    GROM@>2F6F    No, go on
2F6A : DCEQ  VDP*>832E,@>8344 Line number equal?
2F6E : RTNC                                     Yes, return condition bit set

2F6F : DSUB  @>832E,>0004   Next line
2F73 : DCHE  @>832E,@>8330  Even higher than end of line table?
2F76 : BS    GROM@>2F64    Yes, go on searching
2F78 : DST   @>832E,@>8330
2F7B : RTN

Convert integer into ASCII:
2F7C : CLR   @>8361        Number of characters
2F7E : ST    @>8367,>67    Pointer to character
2F81 : DCLR  @>835C
2F83 : DEC   @>8367
2F85 : DDIV  @>835C,>000A   Decimal
2F89 : ADD   @>835F,>30     ASCII
2F8C : ST    *>8367,@>835F  On stack
2F90 : DST   @>835E,@>835C  Go on with rest
2F93 : INC   @>8361        Next digit
2F95 : DCZ   @>835E        Already 0?
2F97 : BR    GROM@>2F81    No, go on
2F99 : RTN

```

```

2F9A : CALL  GROM@>2F7C      Convert integer into ASCII
2F9D : ST      @>8342,*>8367  1st character
2FA1 : CALL  GROM@>2FE3      Print
2FA4 : INC    @>8367
2FA6 : DEC    @>8361          Till all characters
2FA8 : BR     GROM@>2F9D      Loop
2FAA : RTN

```

Convert integer into ASCII and print on screen:

```

2FAB : DST    @>835E,VDP*>8302  Fetch number from VDP
2FAF : CALL  GROM@>2F7C      Entry without VDP, convert into ASCII
2FB2 : ST     VDP*>8320,*>8367  On screen
2FB7 : ADD    VDP*>8320,>60     Add offset
2FBB : DINC   @>8320          Cursor +1
2FBD : INC    @>8367          Pointer +1
2FBF : DEC    @>8361          End of digits?
2FC1 : BR     GROM@>2FB2      No, go on
2FC3 : RTN

```

Fetch 1st token of next line :

```

2FC4 : DSUB   @>832E,>0004     Actual line pointer -4
2FC8 : CZ     @>8389          GROM flag
2FCB : BR     GROM@>2FD6      GROM jump
2FCD : DST    @>832C,VDP*>832E  Address line on text pointer
2FD1 : AND    @>832C,>7F      >0?
2FD4 : BR     GROM@>2FDD      Yes, jump
2FD6 : MOVE   >0002 TO @>832C  FROM GROM@>0000(@>832E) Set text pointer from GROM

2FDD : XML    >1B            Read 1 byte on >8342
2FDF : RTN

```

```

2FE0 : ST     @>8342,>22      ASCII for "
2FE3 : CH     @>8306,@>8307    Enough space?
2FE6 : BR     GROM@>2FEF      Yes, go on
2FE8 : CALL   GROM@>401A      No, write data block
2FEB : DSUB   @>8320,>0020     New start point
2FEF : ST     VDP*>8308,@>8317  Write offset
2FF3 : ADD    VDP*>8308,@>8342  Add or write byte
2FF7 : DINC   @>8308          Address plus 1
2FF9 : INC    @>8306          Actual position in data block
2FFB : ST     @>8354,@>8342    Save from >8354
2FFE : RTN

```

Prescan:

```

2FFF : DST    @>8306,>000A
3003 : AND    @>8388,>DF      Clear bit 3
3007 : DCZ    @>8344          Run flag?
3009 : BR     GROM@>301C      Yes, jump
300B : DST    @>832C,>0320     Text pointer
300F : XML    >1B            Fetch byte
3011 : CALL   GROM@>4D00      Scroll
3014 : ST     @>8316,>01      Set flag
3017 : CALL   GROM@>306D      Proper prescan
301A : BR     GROM@>3028      Execute
301C : CALL   GROM@>302B      Proper prescan
301F : AND    @>8388,>10      Reset all bits except bit 3
3023 : DST    @>83C0,>3567
3028 : B      GROM@>4D04      Execute Basic

302B : DCLR   @>8316
302D : DST    @>832E,@>8332    End line list
3030 : DADD   @>832E,>0003
3034 : DCLR   @>833E          Pointer to symbol table
3036 : DST    @>8340,@>8330    Start line list on free space symbol table
3039 : CZ     @>8389          In GROM?
303C : BS     GROM@>3041      No, jump

```

303E	:	DST	@>8340,@>8370	Top memory on free space for symbol table
3041	:	CLR	@>8343	Option base 0
3043	:	DDEC	@>8340	-1
3045	:	DST	@>8318,@>8340	Free space symbol table on start string space
3048	:	DDEC	@>8318	-1
304A	:	DST	@>831A,@>8318	End string space (low address)
304D	:	OR	@>8388,>80	Set bit 0
3051	:	DST	@>8312,@>8330	Start line list
3054	:	DINCT	@>8312	+2
3056	:	DCEQ	@>832E,@>8312	
3059	:	BR	GROM@>3067	Program, jump
305B	:	CZ	@>8317	Flag?
305D	:	BS	GROM@>3066	
305F	:	DCLR	@>8344	Clear run flag
3061	:	CALL	GROM@>2D99	Error
3064	:	DATA	>20	For-next error
3065	:	DATA	>F9	
3066	:	RTN		
3067	:	CALL	GROM@>2FC4	Fetch byte from first line
306A	:	AND	@>8316,>02	Clear all except bit 6
306D	:	CEQ	@>8342,>8A	Token DIM?
3070	:	BR	GROM@>3081	No, jump
3072	:	CALL	GROM@>31E5	
3075	:	CEQ	@>8342,>B3	Token ,?
3078	:	BS	GROM@>3072	Yes, once more
307A	:	BR	GROM@>30AF	
307C	:	CALL	GROM@>2D99	Error
307F	:	DATA	>20	Bad name
3080	:	DATA	>40	
3081	:	CEQ	@>8342,>9E	Token option?
3084	:	BR	GROM@>30C5	
3086	:	CALL	GROM@>31D1	Test
3089	:	CALL	GROM@>3481	Fetch byte
308C	:	CLOG	@>8316,>02	Bit 6 set ?
308F	:	BR	GROM@>31D6	No, jump
3091	:	CALL	GROM@>30B9	Token base?
3094	:	DATA	>F1	
3095	:	CALL	GROM@>30B9	Token string?
3098	:	DATA	>C8	
3099	:	CALL	GROM@>30B9	Length 1 ?
309C	:	DATA	>01	
309D	:	CLR	@>8343	
309F	:	SUB	@>8342,>30	Integer
30A2	:	BS	GROM@>30AA	Jump if 0
30A4	:	DEC	@>8342	-1
30A6	:	BR	GROM@>30C0	Error, if smaller 0
30A8	:	INC	@>8343	Therefore option base 1
30AA	:	OR	@>8316,>02	Set flag
30AD	:	XML	>1B	Fetch byte
30AF	:	CZ	@>8342	End of line?
30B1	:	BR	GROM@>30C0	No, error
30B3	:	CLOG	@>8316,>01	Direct mode?
30B6	:	BS	GROM@>3056	No, go on to program end
30B8	:	RTN		Return
30B9	:	FETC	@>834A	Data on FAC
30BB	:	CEQ	@>8342,@>834A	Token O.K.?
30BE	:	BS	GROM@>3481	Fetch next byte and return
30C0	:	CALL	GROM@>2D99	Error
30C3	:	DATA	>20	Incorrect statement
30C4	:	DATA	>2C	
30C5	:	CEQ	@>8342,>89	Token DEF?
30C8	:	BR	GROM@>3155	No, jump
30CA	:	CALL	GROM@>31D1	

```

30CD : OR      @>8316,>84      Set flag for DEF
30D0 : CALL   GROM@>31E5      Entry in symbol table
30D3 : CLOG   VDP*>833E,>07   Dimensions?
30D7 : BS     GROM@>3177      No, jump
30D9 : OR     @>8316,>80      Set flags
30DC : OR     @>8388,>08
30E0 : CALL   GROM@>31E8      Entry in symbol table
30E3 : AND    @>8388,>F7      Reset
30E7 : CALL   GROM@>30B9      Token )?
30EA : DATA  >B6
30EB : CALL   GROM@>30B9      Token =?
30EE : DATA  >BE
30EF : MOVE   >002A TO VDP@>0320 FROM VDP*>833E  Fetch 42 bytes from symbol
                                     table
30F6 : DST    @>8300,VDP@>0324  Pointer to name
30FA : CZ     @>8389          In GROM?
30FD : BS     GROM@>310A      No, then jump
30FF : DSUB   @>8300,@>833E    Computer pointer for VDP
3102 : DADD   @>8300,>0320
3106 : DST    VDP@>0324,@>8300  and write in copy at >0320
310A : DST    @>8340,VDP@>0002(@>833E)  Pointer to free space symbol table
310F : DST    @>833E,>0320      Text pointer
3113 : DDEC   @>8340          -1
3115 : CZ     @>8342          End of line
3117 : BS     GROM@>314F      ?
3119 : CLOG   @>8342,>80      Token?
311C : BS     GROM@>312C      No, jump
311E : CEQ    @>8342,>C8      String?
3121 : BS     GROM@>3148      Yes, jump
3123 : CEQ    @>8342,>C7      String?
3126 : BS     GROM@>3148      Yes, jump
3128 : XML    >1B           Fetch byte
312A : BR     GROM@>3115      Go on
312C : OR     @>8316,>80      Set flag
312F : CALL   GROM@>31ED      Symbol entry
3132 : DCEQ   @>833E,>0320      Text pointer >0320?
3136 : BS     GROM@>3115      Yes, go on
3138 : DST    VDP@>0002(@>833E),VDP@>0322
313E : DST    VDP@>0322,@>833E  Save pointer symbol table
3142 : DST    @>833E,>0320      Pointer symbol table
3146 : BR     GROM@>3115
3148 : CALL   GROM@>3488      Fetch length byte string
314B : XML    >1B           Fetch byte
314D : BR     GROM@>3115      Go on
314F : DST    @>833E,VDP@>0322  New pointer to symbol table
3153 : BR     GROM@>30B3      End

3155 : CEQ    @>8342,>9A      Token REM
3158 : BS     GROM@>30B3      Yes, go on, line is of no interest
315A : CEQ    @>8342,>92      Token INPUT
315D : BS     GROM@>31CC
315F : CEQ    @>8342,>93      Token DATA
3162 : BR     GROM@>3169
3164 : CALL   GROM@>31D1      Test direct mode
3167 : BR     GROM@>3056      Go on
3169 : CEQ    @>8342,>87      Token GOSUB
316C : BS     GROM@>31CC      Forget
316E : CEQ    @>8342,>8C      Token FOR
3171 : BR     GROM@>31A2
3173 : INC    @>8317          Loop counter
3175 : BR     GROM@>31CC      Forget at 0
3177 : CALL   GROM@>30B9      Token =
317A : DATA  >BE
317B : CZ     @>8342          Loop end?
317D : BS     GROM@>30B3      Yes, from start i.e. return at direct mode
317F : CLOG   @>8342,>80      Token?

```


3182	:	BS	GROM@>3193	No, jump
3184	:	CEQ	@>8342,>C8	String?
3187	:	BS	GROM@>319B	Yes, jump
3189	:	CEQ	@>8342,>C7	String?
318C	:	BS	GROM@>319B	Yes, jump
318E	:	CALL	GROM@>31DB	
3191	:	BR	GROM@>317B	Go on
3193	:	OR	@>8316,>80	Set flag
3196	:	CALL	GROM@>31ED	Symbol entry
3199	:	BR	GROM@>317B	Go on
319B	:	CALL	GROM@>3488	Fetch length byte
319E	:	XML	>1B	Fetch byte
31A0	:	BR	GROM@>317B	Go on
31A2	:	CEQ	@>8342,>96	Token NEXT
31A5	:	BR	GROM@>31B3	
31A7	:	CALL	GROM@>31D1	Test on direct mode
31AA	:	DEC	@>8317	Loop counter
31AC	:	CGE	@>8317,>00	Negative?
31AF	:	BR	GROM@>3061	For-Next error
31B1	:	BR	GROM@>317B	Go on
31B3	:	CEQ	@>8342,>88	Token return
31B6	:	BS	GROM@>31CC	Forget
31B8	:	CEQ	@>8342,>9B	Token ON
31BB	:	BS	GROM@>31CC	Forget
31BD	:	CEQ	@>8342,>84	Token IF
31C0	:	BS	GROM@>31CC	
31C2	:	CEQ	@>8342,>85	Token GO
31C5	:	BS	GROM@>31CC	
31C7	:	CEQ	@>8342,>86	Token GOTO
31CA	:	BR	GROM@>317B	
31CC	:	CALL	GROM@>31D1	Flag byte
31CF	:	BR	GROM@>317B	
31D1	:	CLOG	@>8316,>01	Direct mode?
31D4	:	BS	GROM@>30B8	No, jump
31D6	:	CALL	GROM@>2D99	Error
31D9	:	DATA	>20	Can't do that
31DA	:	DATA	>BD	
31DB	:	CEQ	@>8342,>C9	Token line number
31DE	:	BR	GROM@>31E2	No, jump
31E0	:	DINCT	@>832C	Text pointer
31E2	:	XML	>1B	Fetch byte
31E4	:	RTN		
Symbol entry into table:				
31E5	:	CALL	GROM@>3481	Fetch byte
31E8	:	CLOG	@>8342,>80	Token?
31EB	:	BR	GROM@>30C0	Yes, jump error
31ED	:	ST	@>8359,>49	
31F0	:	DST	@>830C,@>832C	Text pointer
31F3	:	DDEC	@>830C	-1
31F5	:	CEQ	@>8359,>58	
31F8	:	BS	GROM@>307C	Fetch name on FAC
31FA	:	INC	@>8359	+1
31FC	:	ST	*>8359,@>8342	Byte on FAC
3200	:	XML	>1B	Fetch byte
3202	:	CGT	@>8342,>00	Token?
3205	:	BS	GROM@>31F5	No, fetch entire name
3207	:	DST	@>836C,@>832C	Text pointer
320A	:	DDEC	@>836C	Minus 1
320C	:	CEQ	*>8359,>24	\$ (String)
3210	:	BR	GROM@>3215	No string, jump
3212	:	OR	@>8316,>10	String flag
3215	:	SUB	@>8359,>4A	Length
3218	:	INC	@>8359	

```

321A : CEQ    @>8342,>B7      Token (?)
321D : BS     GROM@>326B     Yes, jump
321F : CLOG   @>8316,>80     Bit 0 set?
3222 : BR     GROM@>3229     Yes, jump
3224 : CLOG   @>8316,>04     Bit 5 set?
3227 : BS     GROM@>30C0     No, error
3229 : DDEC   @>832C        Text pointer minus 1
322B : DST    @>830E,@>832C
322E : CLR    @>83B8
3231 : ST     @>8310,>B8
3234 : CLOG   @>8388,>08     Flag byte bit 4 set?
3238 : BR     GROM@>325B     Yes, jump
323A : XML    >16           Does an entry already exist?
323C : BR     GROM@>325B
323E : DINC   @>832C        Text pointer
3240 : CLOG   @>8316,>80     Bit 0 set ?
3243 : BS     GROM@>32E6
3245 : ST     @>8300,VDP*>834A Fetch from symbol table byte
3249 : CLOG   @>8316,>04
324C : BR     GROM@>32E6     Error
324E : AND    @>8300,>07     Dimensions
3251 : CEQ    *>8310,@>8300 Equal ?
3255 : BR     GROM@>32E6     No, error
3257 : AND    @>8316,>03     Bit 6 and 7 only
325A : RTN
Normal variable:
325B : MOVE   >0010 TO @>835C FROM @>834A      Save name on ARG
3260 : DST    @>8314,>000E Length of entry for numerical variables
3264 : CLOG   @>8316,>14     String or DEF?
3267 : BS     GROM@>3384     No, jump
3269 : BR     GROM@>3380     Jump
Data field:
326B : DST    @>830E,@>832C Text pointer
326E : ST     @>8372,>B7     Pointer to data stack
3271 : MOVE   >0010 TO @>835C FROM @>834A
3276 : CLOG   @>8316,>84     DEF?
3279 : BR     GROM@>32A9     Yes, jump
327B : XML    >1B           Fetch byte
327D : CALL   GROM@>30B9     String?
3280 : DATA  >C8
3281 : CALL   GROM@>2CFC     Change digit into integer
3284 : BS     GROM@>328F     Error
3286 : CZ     @>834A        0?
3288 : BR     GROM@>3294     No, jump
328A : CHE    @>834B,@>8343 Greater option base?
328D : BS     GROM@>3294     Yes, jump
328F : CALL   GROM@>2D99     Error bad value
3292 : DATA  >20
3293 : DATA  >64
3294 : PUSH   @>834B        Number on data stack
3296 : PUSH   @>834A
3298 : CH     @>8372,>BD     Stack to high?
329B : BS     GROM@>30C0     Error
329D : CEQ    @>8342,>B3     Token ,?
32A0 : BS     GROM@>327B     Yes, go on
32A2 : CEQ    @>8342,>B6     Token )?
32A5 : BR     GROM@>30C0     No, error
32A7 : BR     GROM@>32F3     Make an entry
DEF:
32A9 : ST     @>8300,>01
32AC : CALL   GROM@>3481     Fetch byte
32AF : CLOG   @>8342,>80     Token?
32B2 : BS     GROM@>32AC     No, jump
32B4 : CEQ    @>8342,>B6     Token )?
32B7 : BS     GROM@>32EB     Yes, jump
32B9 : CLOG   @>8316,>04

```

32BC	:	BR	GROM@>30C0	Error
32BE	:	CEQ	@>8342,>C7	String?
32C1	:	BS	GROM@>32DD	Yes, jump
32C3	:	CEQ	@>8342,>B7	Token (?)
32C6	:	BS	GROM@>32E2	Yes, jump
32C8	:	CEQ	@>8342,>B3	Token ,?
32CB	:	BR	GROM@>32AC	No, jump
32CD	:	CGT	@>8300,>01	Greater 1
32D0	:	BS	GROM@>32AC	Yes, go on
32D2	:	PUSH	@>8307	On data stack
32D4	:	PUSH	@>8306	
32D6	:	CH	@>8372,>BD	Data stack to high?
32D9	:	BS	GROM@>30C0	Yes,error
32DB	:	BR	GROM@>32AC	Go on
32DD	:	CALL	GROM@>3488	Fetch length byte
32E0	:	BR	GROM@>32AC	Go on
32E2	:	INC	@>8300	+1
32E4	:	BR	GROM@>32AC	Go on
32E6	:	CALL	GROM@>2D99	Error
32E9	:	DATA	>20	Name conflict
32EA	:	DATA	>AF	
32EB	:	DEC	@>8300	-1
32ED	:	BR	GROM@>32AC	Go on, if not 0
32EF	:	PUSH	@>8307	On data stack
32F1	:	PUSH	@>8306	
32F3	:	ST	@>8300,@>8372	Data stack pointer
32F6	:	SUB	@>8300,>B7	Minus 183
32F9	:	SRL	@>8300,>01	Divided by 2
32FC	:	CGT	@>8300,>03	Greater than 3?
32FF	:	BS	GROM@>30C0	Yes, error
3301	:	PUSH	@>8300	On data stack
3303	:	ST	@>8310,@>8372	Old data stack pointer again
3306	:	MOVE	>0010 TO @>834A FROM @>835C	Save name on ARG
330B	:	XML	>16	Search variable name
330D	:	BR	GROM@>3314	Not found, jump
330F	:	DST	@>832C,@>830E	Text pointer
3312	:	BR	GROM@>3240	Finished
3314	:	CLOG	@>8316,>04	DEF?
3317	:	BR	GROM@>3380	Yes, jump
3319	:	ST	@>8310,@>8372	Save data stack pointer
331C	:	DEC	@>8372	-1
331E	:	ST	@>834A,*>837C	Fetch from data stack
3322	:	ST	@>834B,*>837C	
3326	:	DINC	@>834A	+1
3328	:	CLR	@>8308	
332A	:	ST	@>8309,@>8343	Option base
332D	:	DSUB	@>834A,@>8308	
3330	:	DST	@>8314,@>834A	Number in the dimension
3333	:	B	GROM@>3350	
3336	:	ST	@>834A,*>837C	Fetch from data stack
333A	:	ST	@>834B,*>837C	
333E	:	DINC	@>834A	
3340	:	DSUB	@>834A,@>8308	Option base
3343	:	DST	@>8302,@>8314	
3346	:	DMUL	@>8302,@>834A	Total number
3349	:	DCZ	@>8302	Overflow?
334B	:	BR	GROM@>344B	Error
334D	:	DST	@>8314,@>8304	Total number
3350	:	CEQ	@>8372,>B7	Lowest value reached?
3353	:	BR	GROM@>3336	No, go on
3355	:	CLOG	@>8316,>10	String?
3358	:	BS	GROM@>3365	No, jump
335A	:	CLOG	@>8314,>E0	More than 8191?
335D	:	BR	GROM@>344B	Then error
335F	:	DSSL	@>8314,>0001	*2
3363	:	BR	GROM@>336E	Go on

3365	:	CLOG	@>8314,>F0	More than 4095?
3368	:	BR	GROM@>344B	Error
336A	:	DSLL	@>8314,>0003	*8
336E	:	DADD	@>8314,>0006	Plus 6 for basic entry
3372	:	CLR	@>834A	
3374	:	ST	@>834B,*>8310	Number of dimensions from data stack
3378	:	SLL	@>834B,>01	*2 (Word)
337B	:	DADD	@>8314,@>834A	Total number of bytes
337E	:	BR	GROM@>3384	
3380	:	DST	@>8314,>0008	Length 8 for string
3384	:	CZ	@>8389	GROM flag?
3387	:	BR	GROM@>338E	Yes, jump
3389	:	CLOG	@>8316,>01	Direct mode?
338C	:	BS	GROM@>33AA	No, jump
338E	:	CZ	@>836B	0?
3390	:	BS	GROM@>33AA	Yes, jump
3392	:	CLR	@>8300	
3394	:	ST	@>8301,@>836B	Length
3397	:	DST	@>834A,@>8300	
339A	:	CALL	GROM@>3493	Fetch space for entry into symbol table
339D	:	DSUB	@>8340,@>8300	Length
33A0	:	DST	@>830C,@>8340	
33A3	:	DINC	@>830C	Plus 1
33A5	:	MOVE	@>8300 TO VDP*>830C FROM @>835C	Name in symbol table
33AA	:	DST	@>834A,@>8314	Length on FAC
33AD	:	CALL	GROM@>3493	Fetch space in symbol table
33B0	:	CLR	@>834A	
33B2	:	CLOG	@>8316,>10	String?
33B5	:	BS	GROM@>33BA	No, jump
33B7	:	OR	@>834A,>80	Set bit 0
33BA	:	CLOG	@>8316,>04	Bit 6 set ?
33BD	:	BS	GROM@>33C2	No, jump
33BF	:	OR	@>834A,>40	Set bit 1 (User defined function)
33C2	:	ST	@>8372,@>8310	Basis data stack pointer
33C5	:	ST	@>8350,*>837C	Fetch dimensions from data stack
33C9	:	CZ	@>8350	0?
33CB	:	BS	GROM@>33D8	Yes, jump
33CD	:	OR	@>834A,@>8350	Dimensions
33D0	:	CLOG	@>8316,>04	Bit 5 set ?
33D3	:	BR	GROM@>33D8	Yes, then jump
33D5	:	OR	@>8316,>02	Set bit 6
33D8	:	ST	@>834B,@>836B	Length of name
33DB	:	DST	@>834C,@>833E	Link to next entry
33DE	:	DST	@>834E,@>830C	Pointer to name
33E1	:	DSUB	@>8340,@>8314	Minus length of entry
33E4	:	DINC	@>8340	+1
33E6	:	MOVE	>0006 TO VDP*>8340 FROM @>834A	Write 3 words
33EC	:	DST	@>833E,@>8340	Pointer to first entry of symbol table
33EF	:	CLOG	@>8388,>08	Flag byte bit 4 set ?
33F3	:	BR	GROM@>33F9	Yes, jump
33F5	:	DST	VDP@>03E0,@>833E	Save pointer into VDP
33F9	:	DADD	@>8340,>0006	
33FD	:	CLOG	@>8316,>04	Bit 4 set ?
3400	:	BR	GROM@>3424	Yes, jump
3402	:	CH	@>8372,>B7	Data on stack?
3405	:	BR	GROM@>342A	No, then jump
3407	:	ST	@>8372,>B7	Data stack
340A	:	INC	@>8372	+1
340C	:	CHE	@>8372,@>8310	All bytes
340F	:	BS	GROM@>342A	Jump, if all
3411	:	ST	VDP@>0001(@>8340),*>8372	Write word from data stack
3417	:	INC	@>8372	
3419	:	ST	VDP*>8340,*>8372	
341E	:	DDECT	@>8314	Number -2
3420	:	DINCT	@>8340	Address +1
3422	:	BR	GROM@>340A	Loop

3424 : DST VDP*>8340,@>836C Pointer in symbol table
 3428 : BR GROM@>3438 Jump
 342A : DSUB @>8314,>0007 Minus 7
 342E : CLR VDP*>8340 Clear area for the value
 3431 : MOVE @>8314 TO VDP@>0001(@>8340) FROM VDP*>8340 Clear area
 3438 : DST @>8340,@>833E Pointer to first entry in symbol table
 343B : DDEC @>8340 Pointer on free space for symbol table
 343D : AND @>8316,>83 Only permit bits 0,6,7 as flags
 3440 : CLOG @>8316,>80 Bit 0 set ?
 3443 : BS GROM@>3448 No, jump
 3445 : DST @>832C,@>830E New text pointer
 3448 : XML >1B Fetch byte
 344A : RTN

344B : CALL GROM@>2D99 Error
 344E : DATA >20 Memory full
 344F : DATA >49

Check if variable name is permitted:

3450 : CHE @>8342,>30 Greater or equal 0?
 3453 : BR GROM@>345A No, jump
 3455 : CHE @>8342,>3A Greater or equal 9?
 3458 : BR GROM@>2C80 No, return condition bit set
 345A : CHE @>8342,>40 Greater or equal @?
 345D : BR GROM@>3464 No, jump
 345F : CHE @>8342,>5E Greater or equal ^?
 3462 : BR GROM@>2C80 No, return condition bit set
 3464 : CHE @>8342,>5F Greater or equal _?
 3467 : BR GROM@>3480 No, jump
 3469 : CHE @>8342,>61 Greater or equal a?
 346C : BR GROM@>3476 No, jump
 346E : CHE @>8342,>7B Greater or equal {?
 3471 : BS GROM@>3476 Yes, jump
 3473 : SUB @>8342,>20 Change into capital letter
 3476 : CEQ @>8342,>60 Equal `?
 3479 : BS GROM@>3480 Yes, jump
 347B : CHE @>8342,>7B Greater or equal {?
 347E : BR GROM@>2C80 No, return condition bit set
 3480 : RTN

3481 : XML >1B Fetch byte
 3483 : CZ @>8342 End of line ?
 3485 : BS GROM@>30C0 Yes, error
 3487 : RTN

3488 : XML >1B Fetch byte
 348A : DCLR @>8308
 348C : ST @>8309,@>8342 Byte as word in >8308
 348F : DADD @>832C,@>8308 Text pointer now behind the string
 3492 : RTN

Fetch space for PAB or entry in symbol list:

3493 : DST @>834E,@>834A Length
 3496 : DST @>834C,@>8340 Pointer free space symbol table
 3499 : DSUB @>834C,@>8318 Start of string space
 349C : DCGE @>834C,@>834A Enough space?
 349F : BS GROM@>3492 Yes, end
 34A1 : DSUB @>834A,@>834C Free space
 34A4 : DST @>834C,@>831A String space
 34A7 : DSUB @>834C,@>836E
 34AA : DSUB @>834C,>0040 +8*8 for value stack
 34AE : DCGE @>834C,@>834A Enough space?
 34B1 : BS GROM@>34C8 Yes, go on
 34B3 : CALL GROM@>4D18 Garbage collection
 34B6 : DST @>834C,@>831A Compute once more
 34B9 : DSUB @>834C,@>836E
 34BC : DSUB @>834C,>0040

```

34C0 : DST      @>834A,@>834E
34C3 : DCGE     @>834C,@>834A
34C6 : BR       GROM@>344B      Not enough space, memory full error
34C8 : DST      @>834C,@>8318    Start string space
34CB : DSUB     @>834C,@>831A    End of string space, therefore number
34CE : DST      @>834E,@>831A    End of string space
34D1 : DSUB     @>831A,@>834A    Length
34D4 : DCZ      @>834C
34D6 : BS       GROM@>34E0
34D8 : MOVE     @>834C TO VDP@>0001(@>831A) FROM VDP@>0001(@>834E) Shift strings
34E0 : DSUB     @>8318,@>834A    New start string space
34E3 : DST      @>834C,@>8318    On >834C
34E6 : CLR      @>834E
34E8 : ST       @>834F,VDP*>834C Length
34EC : DCHE     @>831A,@>834C    Below end of string space?
34EF : BS       GROM@>3492      No, end
34F1 : DSUB     @>834C,@>834E    Pointer to beginning of string
34F4 : DCZ      VDP@>FFFD(@>834C) Link pointer 0?
34F9 : BS       GROM@>3505      O.k. go on
34FB : DST      @>834A,VDP@>FFFD(@>834C) Fetch link pointer
3501 : DST      VDP*>834A,@>834C Set new pointer in variable list
3505 : DSUB     @>834C,>0004     Next string
3509 : BR       GROM@>34E8
350B : RTN

```

350C : DATA >0000
350E : DATA >0000

Error message :

```

3510 : B       GROM@>5671      Incorrect statement
3513 : B       GROM@>567D      Memory full
3516 : B       GROM@>4D7C      Bad value
3519 : B       GROM@>4D81      String number mismatch

```

CALL CLEAR:

```

351C : CALL    GROM@>37B4      Fetch byte
351F : ALL     >80             Clear screen
3521 : ST      @>837F,>03      XPT on 3rd line
3524 : CALL    GROM@>0012      End

```

Data for sound:

```

3527 : DATA   >42,>0B,>12
352A : DATA   >22,>00,>00,>00,>00
352F : DATA   >01,>FF,>01,>04,>9F,>BF,>DF,>FF,>00

```

CALL SOUND:

```

3538 : MOVE    >0009 TO VDP@>03E2 FROM GROM@>352F Write sound list
353F : CALL    GROM@>3767      Fetch first value
3542 : CGE     @>834A,>00      Negative?
3545 : BS      GROM@>354C      No, jump
3547 : DNEG    @>834A          Number already positive?
3549 : DCLR    @>83CE          Stop sound process
354C : DST     @>8310,>109A     Limit
3550 : CALL    GROM@>377D      CFI for duration
3553 : DMUL    @>834A,>0006     Compute duration, *6
3557 : DDIV    @>834A,>0064     divided by 100
355B : CZ      @>834B          0?
355D : BR      GROM@>3561
355F : INC     @>834B          At least 1
3561 : ST      VDP@>03E4,@>834B Duration in sound list
3565 : MOVE    >000C TO @>8300 FROM GROM@>579A Constant values on >8300
356B : CALL    GROM@>376F      Next value
356E : CALL    GROM@>4F79      Check if numeric
3571 : CGE     @>834A,>00      Negative?
3574 : BR      GROM@>35C3      Yes, jump to noise
3576 : MOVE    >0008 TO @>835C FROM GROM@>3527 Number on ARG

```

```

357C : XML    >09          FDIV
357E : DST    @>8310,>03FF  Limit
3582 : CALL   GROM@>3785   CFI for frequency
3585 : DCHE   @>834A,>0003  Greater 2
3589 : BR     GROM@>3516   Error
358B : DSRC   @>834A,>0004  Shift periodically(Word!)
358F : SRL    @>834A,>04    In LNybble
3592 : DOR    *>830A,@>834A DOR >8306
3596 : INCT   @>830A      +2
3598 : CALL   GROM@>5600   Fetch next value and CFI
359B : AND    *>830B,@>834B Volume
359F : INC    @>830B      +1
35A1 : CEQ    @>8342,>B6   Token )?
35A4 : BS     GROM@>35E2   Yes, end
35A6 : CEQ    @>8342,>B3   Token ,?
35A9 : BR     GROM@>3510   No, error
35AB : XML    >1B        Fetch byte
35AD : SRL    @>834A,>04
35B0 : CEQ    @>830C,>06
35B3 : CEQ    @>830A,>06   3rd sound processor already loaded?
35B6 : BR     GROM@>356B   New start without duration
35B8 : CALL   GROM@>376F   Fetch next value
35BB : CALL   GROM@>4F79   String tag?
35BE : CGE    @>834A,>00   Negative=Noise
35C1 : BS     GROM@>3510
35C3 : CEQ    @>8309,>FF   Noise
35C6 : BR     GROM@>3510
35C8 : DNEG   @>834A      Number positive again
35CA : DST    @>8310,>0008  Limit
35CE : CALL   GROM@>377D   CFI
35D1 : DEC    @>834B      Minus 1
35D3 : ST     @>8309,@>834B
35D6 : OR     @>8309,>E0
35D9 : CALL   GROM@>5600   Fetch volume
35DC : ST     VDP@>03E3,@>834B In Sound list
35E0 : BR     GROM@>35A1   Go on
35E2 : CLR    @>8310
35E4 : CZ     @>83CE      Sound byte
35E7 : BS     GROM@>35F9   If 0, go on
35E9 : SCAN   Keyboard scanning
35EA : BR     GROM@>35E4
35EC : CEQ    @>8375,>02   Clear?
35EF : BR     GROM@>35E4   No, go on waiting
35F1 : CZ     @>8389      GROM?
35F4 : BR     GROM@>35E4   Yes, go on
35F6 : B      GROM@>4E38   Program end
35F9 : ST     @>8400,*>8310 Sound bytes in sound chip
35FE : INC    @>8310      All
3600 : CEQ    @>8310,>0A
3603 : BR     GROM@>35F9
3605 : DST    @>834A,>03E2  Sound list
3609 : I/O    @>834A,>01   Print
360C : BR     GROM@>3620   End

```

```

CALL HCHAR:
360E : CALL   GROM@>37D6   Fetch all particulars
3611 : DCZ    @>834A      0?
3613 : BS     GROM@>361D   Yes, end
3615 : FMT    1 byte on screen
3616 : ... 01('@>8300')
3618 : ... END FMT
3619 : DDEC   @>834A      Till all bytes
361B : BR     GROM@>3615   Loop
361D : ST     @>837F,@>8302 Old screen line again

```

End of the subprograms with variables:

```

3620 : CEQ    @>8342,>B6    )?
3623 : BR      GROM@>3510    Incorrect statement
3625 : XML      >1B          Fetch byte
3627 : CALL    GROM@>0012    Subprogram end

CALL VCHAR:
362A : CALL    GROM@>37D6    Fetch all particular items and set screen pointer
362D : DCZ      @>834A        0?
362F : BS      GROM@>361D    Yes, end
3631 : FMT                                Print byte
3632 : ... 01('@>8300')
3634 : ... 1F<                    31 go on
3635 : ... END FMT
3636 : DDEC     @>834A        Till all
3638 : BS      GROM@>361D    All, end
363A : CZ      @>837E        Line 0?
363C : BR      GROM@>3631    No, print next byte
363E : INC     @>837F        Column plus 1
3640 : B       GROM@>3631    Go on

CALL CHAR:
3643 : CALL    GROM@>3767    Fetch first value
3646 : DST      @>8310,>009F  Limit
364A : CALL    GROM@>3785    CFI
364D : DCHE     @>834A,>0020  Greater or equal space?
3651 : BR      GROM@>3516    No, error
3653 : DSSL     @>834A,>0003  *8, therefore address pattern descriptor table
3657 : DADD     @>834A,>0300  + offset
365B : DST      @>8304,@>834A  Save address
365E : PARS     >B6          Go on till )
3660 : CEQ      @>834C,>65    String tag?
3663 : BR      GROM@>3519    No, error
3665 : DCGT     @>8304,@>8324  Higher basis of value stack?
3668 : BR      GROM@>36A3    No, execute
366A : DST      @>835E,@>8304  Address
366D : DSUB     @>835E,@>8324  Bytes, be missing
3670 : DST      @>8306,@>836E  Top value stack
3673 : DADD     @>8306,@>835E  Plus bytes
3676 : DADD     @>8306,>000F  + 15
367A : DCHE     @>831A,@>8306  Lower than string area?
367D : BS      GROM@>3687    Yes, jump
367F : CALL    GROM@>51A9    Garbage collection
3682 : DCHE     @>831A,@>8306  Enough space ?
3685 : BS      GROM@>3513    No, error
3687 : DSUB     @>8306,>0008  -8
368B : DST      @>8300,@>836E  Top value stack
368E : DADD     @>8300,>0007  +7
3692 : DST      @>835C,@>836E  Top value stack
3695 : DSUB     @>835C,@>8324  Number of bytes
3698 : BS      GROM@>369D    Jump at 0
369A : CALL    GROM@>201E    Shift stack
369D : DST      @>8324,@>8304  New basis of value stack
36A0 : DADD     @>836E,@>835E  New end of value stack
36A3 : DST      @>835E,@>834E  Address of string
36A6 : DST      @>8360,@>8350  Length
36A9 : DCH      @>8350,>0010  Longer than 16?
36AD : BR      GROM@>36B3    No
36AF : DST      @>8360,>0010  Length is 16
36B3 : ST       @>834A,>30    On FAC 16*>30
36B6 : MOVE     >000F TO @>834B FROM @>834A
36BB : DCZ      @>8360        0?
36BD : BS      GROM@>36C4    Yes, jump
36BF : MOVE     @>8360 TO @>834A FROM VDP*>835E Fetch string on FAC
36C4 : ST       @>8311,>4A    FAC
36C7 : ST       @>8310,>08    8
36CA : CLR      @>830C

```



```

36CC : SLL    @>830C,>04
36CF : ST     @>835C,*>8311    Fetch byte from FAC+
36D3 : CHE    @>835C,>30      Greater 0?
36D6 : BR     GROM@>3516     No, error
36D8 : CH     @>835C,>39      Greater 9?
36DB : BR     GROM@>36E7
36DD : CHE    @>835C,>41      Greater or equal A?
36E0 : BR     GROM@>3516     No, error
36E2 : CH     @>835C,>46      Greater F?
36E5 : BS     GROM@>3516     No, error
36E7 : SUB    @>835C,>30      -30
36EA : CH     @>835C,>0A      Greater 10?
36ED : BR     GROM@>36F2     No, jump
36EF : SUB    @>835C,>07      Minus 7
36F2 : OR     @>830C,@>835C  Transfer to >830C
36F5 : INC    @>8311         Next byte
36F7 : CLOG   @>8311,>01     Every 2nd time
36FA : BR     GROM@>36CC     Go on or
36FC : ST     VDP*>8304,@>830C Write byte in pattern descriptor table
3700 : DINC   @>8304         Increase address VDP
3702 : DEC    @>8310         Loop counter for 8 bytes
3704 : BR     GROM@>36CA     Not yet 8 bytes, then return
3706 : BR     GROM@>3620     End

```

CALL KEY:

```

3708 : CALL   GROM@>3767     Fetch first value
370B : DST    @>8310,>0005    Limit 5
370F : CALL   GROM@>3785     CFI for keyboard mode
3712 : CALL   GROM@>5770     Fetch numerical variable and scan keyboard
3715 : BS     GROM@>3722     New key, then jump
3717 : CEQ    @>8375,>FF     No key
371A : BR     GROM@>3720
371C : DCLR   @>834A         Status=0
371E : BR     GROM@>3722
3720 : DNEG   @>834A         Status=-1
3722 : XML    >15           Transfer status to variable
3724 : DST    @>834A,>4001    Repair the 1
3728 : CEQ    @>8375,>FF     No key?
372B : BS     GROM@>3742
372D : CHE    @>8375,>64     Key value greater 100?
3730 : BR     GROM@>373D
3732 : INC    @>834A         Exponent plus 1
3734 : SUB    @>8375,>64     Minus 100
3737 : ST     @>834C,@>8375  Key value one digit back
373A : B      GROM@>3740
373D : ST     @>834B,@>8375  Key value in number
3740 : BR     GROM@>3744
3742 : DNEG   @>834A         -1
3744 : XML    >15           Transfer key to variable
3746 : BR     GROM@>3620     End

```

CALL JOYST:

```

3748 : CALL   GROM@>3767     Fetch value
374B : DST    @>8310,>0004    Limit
374F : CALL   GROM@>377D     CFI for mode
3752 : CALL   GROM@>5770     Scan keyboard and fetch variables
3755 : ST     @>8300,@>8376  Y value
3758 : CALL   GROM@>5755     Transfer to variable
375B : DST    @>834A,>4001    Repair 1
375F : ST     @>8300,@>8377  X value
3762 : CALL   GROM@>5755     Transfer to variable
3765 : BR     GROM@>3620     End

```

```

3767 : CALL   GROM@>37B4     Fetch byte and text pointer
376A : CALL   GROM@>57A6     Check token (
376D : XML    >1B           Fetch byte

```

```

376F : PARS >B6 Go on till )
3771 : CEQ @>8342,>B3 Token ,?
3774 : BR GROM@>3510 No, error
3776 : XML >1B Fetch byte
3778 : RTN

3779 : DST @>8310,>0010 Set limit
377D : CALL GROM@>3785 CFI
3780 : DCZ @>834A 0?
3782 : BS GROM@>3516 Error
3784 : RTN

3785 : CALL GROM@>5740 CFI
3788 : DCH @>834A,@>8310 Greater limit?
378B : BS GROM@>3516 Error
378D : RTN

```

Set column and line for subprogram:

```

378E : CALL GROM@>3767 Next argument
3791 : DST @>8310,>0018 Limit
3795 : CALL GROM@>377D CFI
3798 : ST @>8302,@>837F Column screen
379B : DEC @>834B
379D : ST @>837E,@>834B Line screen
37A0 : CALL GROM@>376F Next argument
37A3 : DST @>8310,>0020 Limit
37A7 : CALL GROM@>377D CFI
37AA : DEC @>834B
37AC : ST @>837F,@>834B Column screen
37AF : RTN
37B0 : DATA >2020,>2020

```

Fetch Basicbyte

```

37B4 : CZ @>8389 GROM?
37B7 : BR GROM@>37BC
37B9 : DST @>832C,@>8356 Text pointer
37BC : XML >1B Fetch byte
37BE : RTN

```

CALL SCREEN:

```

37BF : CALL GROM@>37B4 Fetch byte
37C2 : CALL GROM@>57A6 Check (
37C5 : XML >1B Fetch byte
37C7 : PARS >B6 Fetch value
37C9 : CALL GROM@>3779 CFI
37CC : DEC @>834B -1
37CE : MOVE >0001 TO REG>07 FROM @>834B Load register with background color
37D3 : B GROM@>3620 End

37D6 : CALL GROM@>378E Fetch and set screen pointer
37D9 : PARS >B6 Go on till )
37DB : CALL GROM@>5740 CFI
37DE : ADD @>834B,>60 Add offset
37E1 : ST @>8300,@>834B Character on >8300
37E4 : DST @>834A,>0001 Repetition 1
37E8 : CEQ @>8342,>B6 Token )?
37EB : BS GROM@>37F9 Yes, end
37ED : CEQ @>8342,>B3 Token ,?
37F0 : BR GROM@>3510 No, error
37F2 : XML >1B Fetch byte
37F4 : PARS >B6 Go on till )
37F6 : CALL GROM@>5740 CFI
37F9 : RTN

37FA : DATA >0000,>0000,>C945

```

4000	: BR	GROM@>426C	Display
4002	: BR	GROM@>4160	Delete
4004	: BR	GROM@>4227	Print
4006	: BR	GROM@>4344	Input
4008	: BR	GROM@>401E	Open
400A	: BR	GROM@>4174	Close
400C	: BR	GROM@>41D7	Restore
400E	: BR	GROM@>45E3	Read
4010	: BR	GROM@>4956	Fetch data from GROM or VDP
4012	: BR	GROM@>41CF	Close all open files
4014	: BR	GROM@>46FC	Save
4016	: BR	GROM@>4641	Load
4018	: BR	GROM@>474C	List
401A	: BR	GROM@>4BFC	Output record
401C	: BR	GROM@>482B	EOF
Basic OPEN:			
401E	: CALL	GROM@>4993	Fetch number
4021	: BS	GROM@>57DE	Error
4023	: CALL	GROM@>49B1	Search PAB
4026	: BS	GROM@>57DE	Nothing found, error
4028	: CEQ	@>8342,>B5	Token :
402B	: BR	GROM@>40F5	Error
402D	: XML	>1B	Fetch byte
402F	: CALL	GROM@>4BA1	Build PAB with name
4032	: DDEC	@>832C	Text pointer in Basic line minus 1
4034	: XML	>1B	Fetch byte
4036	: CEQ	@>8342,>B3	Token ,?
4039	: BR	GROM@>40FD	No, error value
403B	: XML	>1B	
403D	: CEQ	@>8342,>A2	Display
4040	: BS	GROM@>40D6	
4042	: CEQ	@>8342,>92	Input
4045	: BS	GROM@>40E0	
4047	: SUB	@>8342,>F3	->F3 (Token variable)
404A	: CHE	@>8342,>09	Greater or equal 9
404D	: BS	GROM@>40F2	Error
404F	: CASE	@>8342	
4051	: BR	GROM@>40AB	Variable
4053	: BR	GROM@>406B	Relative
4055	: BR	GROM@>40D1	Internal
4057	: BR	GROM@>4070	Sequential
4059	: BR	GROM@>4095	Output
405B	: BR	GROM@>409A	Update
405D	: BR	GROM@>40A4	Append
405F	: BR	GROM@>40B0	Fixed
Permanent:			
4061	: CLOG	@>8317,>04	Inefficient, since not considered
4064	: BR	GROM@>40F2	
4066	: OR	@>8317,>04	
4069	: BR	GROM@>4034	
Relative:			
406B	: OR	VDP@>0005(@>8304),>01	Set relative flag bit
Sequential:			
4070	: CLOG	@>8317,>08	Control if something is already set
4073	: BR	GROM@>40F2	Error
4075	: OR	@>8317,>08	Set flag bit
4078	: XML	>1B	Fetch byte
407A	: CEQ	@>8342,>B3	Token ,?
407D	: BS	GROM@>403B	Go on
407F	: CZ	@>8342	End of line
4081	: BS	GROM@>40FD	End
4083	: CALL	GROM@>408D	Fetch number
4086	: DST	VDP@>000A(@>8304),@>834A	Number on record number
408B	: BR	GROM@>4036	

```

408D : PARS >B3
408F : CALL GROM@>499C Compute number into integer
4092 : BS GROM@>40F2 0=Error
4094 : RTN
Output:
4095 : OR VDP@>0005(@>8304),>02 Set output flag bit
Update:
409A : CLOG @>8317,>01 Control if something is set already
409D : BR GROM@>40F2 Error
409F : OR @>8317,>01 Set flag
40A2 : BR GROM@>4034 Go on
Append:
40A4 : OR VDP@>0005(@>8304),>06 Set flag bit
40A9 : BR GROM@>409A Go on
Variable:
40AB : OR VDP@>0005(@>8304),>10 Set flag bit
Fixed:
40B0 : XML >1B Fetch byte
40B2 : CEQ @>8342,>B3 Token ,?
40B5 : BS GROM@>40C7
40B7 : CZ @>8342 End of line
40B9 : BS GROM@>40C7
40BB : CALL GROM@>408D Fetch record length
40BE : CZ @>834A 0?
40C0 : BR GROM@>40F2 Error
40C2 : ST VDP@>0008(@>8304),>834B Set record length
40C7 : CLOG @>8317,>10 Set already?
40CA : BR GROM@>40F2
40CC : OR @>8317,>10
40CF : BR GROM@>4036 Go on
Internal:
40D1 : OR VDP@>0005(@>8304),>08 Set flag bit
Display:
40D6 : CLOG @>8317,>02 Set already?
40D9 : BR GROM@>40F2
40DB : OR @>8317,>02
40DE : BR GROM@>4034 Go on
Input:
40E0 : OR VDP@>0005(@>8304),>04 Set flag bit
40E5 : BR GROM@>409A Go on

40E7 : CLR @>8302 Length
40E9 : ST @>8303,VDP@>0003(@>8304) Internal offset
40EE : DADD @>8340,@>8302 New pointer free space symbol table
40F1 : RTN

40F2 : CALL GROM@>40E7 Reset PAB
40F5 : CALL GROM@>284E Error
40F8 : DATA >20 Incorrect statement
40F9 : DATA >2C
40FA : B GROM@>2012 Return Basic

40FD : CLOG VDP@>0005(@>8304),>01 Is relative set?
4102 : BS GROM@>410D
4104 : CLOG VDP@>0005(@>8304),>10 No, set variable
4109 : BR GROM@>40F2
410B : BR GROM@>4117
410D : CLOG @>8317,>10 Fixed or variable set?
4110 : BR GROM@>4117
4112 : OR VDP@>0005(@>8304),>10 No, set variable
4117 : CALL GROM@>4CC6 DSRLINK
411A : BR GROM@>57C0 Not found, error
411C : DCLR VDP@>000A(@>8304) Clear record number
4120 : CZ VDP@>0008(@>8304) Length of record 0?
4124 : BS GROM@>40F2 Error
4126 : ST @>8303,VDP@>0008(@>8304) Right record length

```

```

412B : CLR    @>8302
412D : CLR    VDP@>0003(@>8304) Clear internal offset
4131 : DST    @>834A,@>8302    Record length on FAC
4134 : DCZ    @>833C          Exist PAB pointer ?
4136 : BR     GROM@>413D
4138 : DST    @>833C,@>8304    New PAB pointer
413B : BR     GROM@>414F
413D : DST    @>830A,@>833C
4140 : DCZ    VDP*>830A      Next PAB
4143 : BS     GROM@>414B
4145 : DST    @>830A,VDP*>830A
4149 : BR     GROM@>4140      Till end
414B : DST    VDP*>830A,@>8304 Set PAB pointer into old PAB pointer
414F : DST    VDP@>0006(@>8304),@>8304 Buffer pointer
4154 : CALL   GROM@>2844      Fetch space for buffer
4157 : DSUB   @>8340,@>8302    New pointer symbol table
415A : DSUB   VDP@>0006(@>8304),@>8302 Correct pointer to buffer
415F : CONT

```

Basic DELETE:

```

4160 : CLR    @>8317
4162 : CALL   GROM@>4BA1      Built PAB
4165 : CLR    @>8302
4167 : ST     @>8303,VDP@>0003(@>8304) Internal offset
416C : DADD   @>8340,@>8302    Old pointer again
416F : CALL   GROM@>4CB9      DSR access
4172 : DATA  >07           Op code
4173 : CONT

```

Basic CLOSE:

```

4174 : CALL   GROM@>4993      Fetch file number
4177 : BS     GROM@>57DE      Error
4179 : CALL   GROM@>49B1      Search PAB
417C : BR     GROM@>57DE      Not found, error
417E : CALL   GROM@>49D0      Data block to be written, then write
4181 : ST     VDP@>0004(@>8304),>01 Close op code
4186 : CEQ    @>8342,>B5      Token :?
4189 : BR     GROM@>4199      No, jump
418B : XML    >1B
418D : CEQ    @>8342,>99      Token delete?
4190 : BR     GROM@>40F5      No, error
4192 : ST     VDP@>0004(@>8304),>07 Op code delete
4197 : XML    >1B           Fetch byte
4199 : CALL   GROM@>4CC6      Call DSR
419C : BR     GROM@>41A2      If error, jump
419E : CALL   GROM@>49E6      Clear PAB
41A1 : CONT

```

```

41A2 : DST    @>835C,VDP@>0004(@>8304) Op code on >835C
41A7 : CALL   GROM@>49E6      Clear PAB
41AA : DST    @>8304,@>8340    New pointer free space symbol table
41AD : DSUB   @>8304,>0006     -6
41B1 : DST    VDP@>0004(@>8304),@>835C Repair op code
41B6 : BR     GROM@>57D6      I/O error

```

```

41B8 : DST    @>8304,VDP*>8304 Fetch pointer to next PAB
41BC : DCZ    VDP*>8304      More PAB?
41BF : BR     GROM@>41B8      No, fetch next
41C1 : CALL   GROM@>49D0      If necessary write data block
41C4 : ST     VDP@>0004(@>8304),>01 Op code close
41C9 : CALL   GROM@>4CC6      Call DSR
41CC : CALL   GROM@>49E6      Eliminate PAB

```

Close all open files :

```

41CF : DST    @>8304,@>833C    Pointer on PAB's
41D2 : DCZ    @>833C          No PAB?

```

41D4 : BR GROM@>41BC No, jump
 41D6 : RTN

Basic RESTORE:

41D7 : DCLR @>834A
 41D9 : CEQ @>8342,>FD Token #?
 41DC : BR GROM@>41F9 No, jump
 41DE : CALL GROM@>4993 Fetch file number
 41E1 : DCZ @>834A 0?
 41E3 : BS GROM@>4202
 41E5 : CALL GROM@>49B1 Search PAB
 41E8 : BR GROM@>57DE Not found, error
 41EA : CALL GROM@>49D0 If necessary write data block
 41ED : DCLR VDP@>000A(@>8304) Restore
 41F1 : CALL GROM@>4B03 If necessary write special record number
 41F4 : CALL GROM@>4CB9 Call DSR
 41F7 : DATA >04 Op code
 41F8 : CONT
 41F9 : CZ @>8342 Line end?
 41FB : BS GROM@>4202 Yes, jump
 41FD : CLR @>8311
 41FF : CALL GROM@>4D06 Fetch line number
 4202 : DCEQ @>8330,@>8332 Does line list exist?
 4205 : BS GROM@>4D88 No, error
 4207 : DST @>8336,@>8332 Begin line list on pointer to data line
 420A : DSUB @>8336,>0003 First line
 420E : DCH @>834A,VDP*>8336 Line number smaller than searched one?
 4212 : BR GROM@>421F No, jump
 4214 : DCEQ @>8336,@>8330 Reached end of line list ?
 4217 : BS GROM@>57E8 Yes, error
 4219 : DSUB @>8336,>0004 Next line
 421D : BR GROM@>420E Go on
 421F : DADD @>8336,>0003 +3
 4223 : CALL GROM@>4D08 Set data pointer
 4226 : CONT

Basic PRINT:

4227 : CEQ @>8342,>FD Token # for file ?
 422A : BR GROM@>426C No, go on with display
 422C : CALL GROM@>4C9B Set cursor positions
 422F : CALL GROM@>4993 Fetch file number
 4232 : DCZ @>834A 0 (screen)?
 4234 : BS GROM@>425F
 4236 : CALL GROM@>49B1 Search PAB
 4239 : BR GROM@>57DE Not found, error
 423B : CLOG VDP@>0005(@>8304),>04 Input?
 4240 : BS GROM@>4249
 4242 : CLOG VDP@>0005(@>8304),>02 Still Append?
 4247 : BS GROM@>57DE No, error
 4249 : CEQ VDP@>0004(@>8304),>02 Read op code?
 424E : BR GROM@>4254 No, jump
 4250 : CLR VDP@>0003(@>8304) Clear internal offset
 4254 : ST VDP@>0004(@>8304),>03 Write op code
 4259 : CALL GROM@>4C2A Set pointer in PAB
 425C : CALL GROM@>4B03 Write record number in PAB if necessary
 425F : CZ @>8342 Line end?
 4261 : BS GROM@>4325
 4263 : CEQ @>8342,>B5 Token :?
 4266 : BR GROM@>40F5 Error
 4268 : XML >1B Fetch byte
 426A : BR GROM@>426F Go on

Basic DISPLAY:

426C : CALL GROM@>4C9B Set cursor position
 426F : CALL GROM@>4B6F Check table token
 4272 : CEQ @>8342,>FC Token TAB

```

4275 : BS      GROM@>42D3
4277 : PARS    >B5          Go on till :
4279 : CALL    GROM@>433A      Check display
427C : BS      GROM@>42A8      Yes, jump
427E : CEQ     @>834C,>65      String?
4281 : BS      GROM@>4291
4283 : ST      @>8356,>08      Length
4286 : MOVE   >0008 TO @>835C FROM @>834A On ARG
428B : ST      @>8355,>5C      Pointer to "String", i.e. number
428E : CALL    GROM@>4B53      Change into string, with string entry on FAC
4291 : ST      @>835C,@>8307    Whole length
4294 : SUB    @>835C,@>8306    Minus present length
4297 : INC    @>835C          +1 for length byte
4299 : CHE    @>8351,@>835C    Length
429C : BS      GROM@>57DE      Error
429E : ST      VDP*>8308,@>8351 Length
42A2 : DINC   @>8308
42A4 : INC    @>8306
42A6 : BR     GROM@>42AD
42A8 : CEQ    @>834C,>65      Output display
42AB : BR     GROM@>42B2      No, jump
42AD : CALL    GROM@>4C6E      Write string
42B0 : BR     GROM@>42CE
42B2 : CLR    @>8355
42B4 : CALL    GROM@>0014      Change number into string
42B7 : CALL    GROM@>4B53      Write string in VDP
42BA : CALL    GROM@>4C6E      And write string in file
42BD : CHE    @>8307,@>8306    Enough space?
42C0 : BR     GROM@>42CE
42C2 : ST      VDP*>8308,>20  One space?
42C6 : ADD    VDP*>8308,@>8317 Offset
42CA : DINC   @>8308
42CC : INC    @>8306
42CE : CALL    GROM@>4B6F      Check more tokens
42D1 : BR     GROM@>40F5
TAB:
42D3 : CALL    GROM@>433A      Check display
42D6 : BR     GROM@>57DE      No, error
42D8 : XML    >1B          Fetch byte
42DA : CEQ    @>8342,>B7      Token (?)
42DD : BR     GROM@>40F5      Error
42DF : PARS    >B6          Go on till )
42E1 : CALL    GROM@>4AF9      Change into integer
42E4 : ST      @>834C,@>8307
42E7 : CALL    GROM@>4B62      Compute tab at the beginning
42EA : CH     @>8306,@>834B    Enough space?
42ED : BR     GROM@>42F2
42EF : CALL    GROM@>4BFC      Write data
42F2 : CEQ    @>8306,@>834B
42F5 : BS      GROM@>42CE      End, go on
42F7 : ST      @>8303,@>834B
42FA : CALL    GROM@>4C43      Fill with space
42FD : BR     GROM@>42CE

42FF : ST      @>8303,@>8306    Length
4302 : DEC    @>8303
4304 : CLR    @>8302
4306 : DIV    @>8302,>0E      /14
4309 : INC    @>8302          +1
430B : MUL    @>8302,>0E      *14 = 28 = one line
430E : CH     @>8307,@>8303    End of data block?
4311 : BR     GROM@>431A
4313 : INC    @>8303          +1
4315 : CALL    GROM@>4C43      Set pointer
4318 : BR     GROM@>431D
431A : CALL    GROM@>4BFC      Write data block

```

```

Token":":
431D : XML    >1B                Fetch byte
431F : CZ     @>8342                End of line?
4321 : BR     GROM@>426F           New start with "scroll"
4323 : BR     GROM@>4328           End
End of line (rcond):
4325 : CALL   GROM@>4BFC           End with writing data block
4328 : CZ     @>8317                Screen flag
432A : BR     GROM@>4334           Yes, end
432C : DEC    @>8306                New internal offset
432E : ST     VDP@>0003(@>8304),@>8306
4333 : CONT
4334 : ST     @>837F,@>8306        New line pointer
4337 : INCT   @>837F
4339 : CONT
433A : CZ     @>8317                Flag file
433C : BR     GROM@>49CC           Return condition bit set
433E : CLOG   VDP@>0005(@>8304),>08 Internal
4343 : RTNC   Return condition bit set at display

Basic INPUT:
4344 : CALL   GROM@>4C9B           Prepare pointer
4347 : CEQ    @>8342,>FD           Token #?
434A : BR     GROM@>44B3           No, jump
434C : CALL   GROM@>4993           Fetch file number
434F : DCZ    @>834A                0?
4351 : BS     GROM@>44DC           Yes, then jump screen
4353 : CALL   GROM@>49B1           Search PAB
4356 : BR     GROM@>57DE           Not found, error
4358 : CLOG   VDP@>0005(@>8304),>02 Update or input?
435D : BR     GROM@>57DE           No, error
435F : CALL   GROM@>49D0           Write data block if necessary
4362 : ST     VDP@>0004(@>8304),>02 Input
4367 : CALL   GROM@>4B03           Fetch and write data set number if necessary
436A : CEQ    @>8342,>B5           Token :?
436D : BR     GROM@>40F5           No, error
436F : XML    >1B                Fetch byte
4371 : CLR    @>8317                No screen offset
4373 : CLOG   VDP@>0005(@>8304),>08 Display?
4378 : BS     GROM@>4410           Yes, jump
437A : CZ     VDP@>0003(@>8304) Internal offset?
437E : BR     GROM@>4383           Yes, still there
4380 : CALL   GROM@>4CC0           Call DSR
4383 : ST     @>832B,VDP@>0003(@>8304) Internal offset on screen output end
4388 : CLR    @>832A
438A : DST    @>8366,VDP@>0006(@>8304) Fetch buffer pointer
438F : DADD   @>8366,@>832A        Plus offset
4392 : CALL   GROM@>4CEA           Fetch variable name and build stack entry
4395 : XML    >17                VPUSHG
4397 : DCLR   @>830C
4399 : CHE    @>832B,VDP@>0009(@>8304) Actual data set short
439E : BS     GROM@>43A8
43A0 : ST     @>830D,VDP*>8366 Length on >830D
43A4 : DINC   @>8366
43A6 : INC    @>832B
43A8 : CH     @>834C,>63            Numeric ?
43AB : BR     GROM@>43B5           Yes, jump
43AD : DST    @>8350,@>830C        Length
43B0 : CALL   GROM@>492D           Fetch string from PAB buffer
43B3 : BR     GROM@>43E3
43B5 : CEQ    @>830D,>08            Length
43B8 : BR     GROM@>57DE           No, error
43BA : MOVE   @>830C TO @>834A FROM VDP*>8366 Fetch value on FAC
43BF : DCZ    @>834A                0?
43C1 : BS     GROM@>43E1
43C3 : ST     @>835C,>51

```



```

43C6 : CH      *>835C,>63      Is a value wrong?
43CA : BS      GROM@>57DE      Then error
43CC : DEC      @>835C
43CE : CEQ      @>835C,>4B      All 6 bytes
43D1 : BR      GROM@>43C6
43D3 : DST      @>835C,@>834A    Check independently of sign (+ -)
43D6 : DABS     @>835C
43D8 : DEC      @>835D
43DA : CH      @>835D,>62
43DD : BS      GROM@>57DE      Error
43DF : BR      GROM@>43E3
43E1 : DCLR     @>834C
43E3 : DADD     @>8366,@>830C    New buffer pointer
43E6 : ADD      @>832B,@>830D    New value screen input end
43E9 : XML      >15          Transfer value
43EB : CLR      VDP@>0003(@>8304) Internal offset 0
43EF : CEQ      @>8342,>B3      Token ,?
43F2 : BR      GROM@>440F      No, end
43F4 : XML      >1B          Fetch byte
43F6 : CZ      @>8342          Line end ?
43F8 : BS      GROM@>4403
43FA : CHE      @>832B,VDP@>0009(@>8304) Screen input pointer smaller length
43FF : BS      GROM@>4380      No, call new DSR                of data block.
4401 : BR      GROM@>4392      Otherwise without DSR
4403 : CHE      @>832B,VDP@>0009(@>8304) Screen input pointer smaller length
4408 : BS      GROM@>440F      No, end                of data block.
440A : ST      VDP@>0003(@>8304),@>832B Store internal offset in PAB
440F : CONT
4410 : CALL     GROM@>48CC      Input display
4413 : DST      @>8338,>0320    Seize crunch pointer
4417 : CLR      @>8307
4419 : ST      VDP@>0004(@>8304),>02 Read op code
441E : CZ      VDP@>0003(@>8304) Internal offset?
4422 : BR      GROM@>4449
4424 : DINC     @>8338
4426 : DDEC     @>8338
4428 : ST      VDP*>8338,>B3    Token ,
442C : CALL     GROM@>4CC0      Call DSR, fetch data set
442F : CLR      VDP@>0003(@>8304) Internal offset 0
4433 : CALL     GROM@>45C6      Buffer pointer on start "Screen" input
4436 : ST      @>832A,VDP@>0009(@>8304) Length of received data set
443B : CZ      @>832A          0?
443D : BS      GROM@>4449
443F : ADD      VDP*>8320,>60    Add offset
4443 : DINC     @>8320
4445 : DEC      @>832A
4447 : BR      GROM@>443B      Loop till end
4449 : CALL     GROM@>45C6      Buffer pointer on start "Screen"
444C : ST      @>832B,VDP@>0009(@>8304) Length of data block
4451 : CLR      @>832A
4453 : DADD     @>832A,VDP@>0006(@>8304) Plus pointer to buffer
4458 : DDEC     @>832A          Result: End of input buffer
445A : CALL     GROM@>2014      Crunch string
445D : BS      GROM@>57E3      No jump, input error
445F : INC      @>8311
4461 : ADD      @>8307,@>8311
4464 : CHE      @>8307,@>8310
4467 : BR      GROM@>4426
4469 : DDECT    @>832C          Pointer to text
446B : XML      >1B          Fetch byte
446D : CALL     GROM@>45C6      Set pointer
4470 : CLR      VDP@>0003(@>8304) Internal offset 0
4474 : CEQ      @>8342,>B3      Token ,?
4477 : BR      GROM@>44AE      No, jump
4479 : CEQ      @>8307,@>8310
447C : BS      GROM@>44AE

```

```

447E : SUB    @>8307,@>8310
4481 : SUB    @>8311,@>8307    Scan " and ,
4484 : CEQ    VDP*>8320,>82    "?"
4488 : BR     GROM@>4496    No, go on
448A : DINC   @>8320    Forget
448C : CEQ    VDP*>8320,>82    "?"
4490 : BR     GROM@>448A    Till end of string
4492 : DINC   @>8320
4494 : BR     GROM@>4484
4496 : DINC   @>8320
4498 : CEQ    VDP@>FFFF(@>8320),>8C ,?
449E : BR     GROM@>4496    No, jump
44A0 : DEC    @>8311
44A2 : BR     GROM@>4484    New start
44A4 : DSUB   @>8320,VDP@>0006(@>8304) Pointer to start minus buffer address
44A9 : ST     VDP@>0003(@>8304),@>8321 Add internal offset
44AE : ST     @>8311,@>8310
44B1 : BR     GROM@>452D
44B3 : CALL   GROM@>4C9B    Screen input, prepare pointer ten
44B6 : DST    @>830A,@>832C    Text pointer
44B9 : DDEC   @>830A
44BB : CALL   GROM@>4B2F    Text pointer to start of string
44BE : BS     GROM@>44D6    End
44C0 : CEQ    @>8342,>B5    Token :?
44C3 : BR     GROM@>44BB    No, then till end
44C5 : DST    @>832C,@>830A    New text pointer
44C8 : XML    >1B    Fetch byte
44CA : PARS   >B5    Go on
44CC : CEQ    @>834C,>65    String tag?
44CF : BR     GROM@>40F5    No, error
44D1 : CALL   GROM@>4C6E    Write string (Input dialogue)
44D4 : BR     GROM@>44E5    Jump
44D6 : DST    @>832C,@>830A    New text pointer
44D9 : ST     @>8342,>B5    Token :
44DC : CALL   GROM@>45D3    Screen address
44DF : ST     VDP*>8308,>9F    Write > ?
44E3 : DINCT  @>8308    +2
44E5 : CEQ    @>8342,>B5    Token :?
44E8 : BR     GROM@>40F5    No, error
44EA : XML    >1B    Fetch byte
44EC : CALL   GROM@>48CC    Fetch variable
44EF : CALL   GROM@>45D3    Scroll if necessary
44F2 : DST    @>8320,@>8308    Start screen input
44F5 : ST     VDP*>8308,>80    Space
44F9 : DINC   @>8308
44FB : DCHE   @>8308,>02FE    Clear line
44FF : BR     GROM@>44F5
4501 : DST    VDP@>02FE,>7F7F Mark end of line
4506 : CZ     @>83CE    Sound byte
4509 : BR     GROM@>450E
450B : CALL   GROM@>0034    Accept tone output
450E : DEX    @>836E,@>830E    Again old value stack pointer
4511 : CALL   GROM@>2832    Keyboard input
4514 : DEX    @>836E,@>830E    Again new value stack pointer
4517 : DST    @>8338,>0320    Crunch buffer pointer
451B : CALL   GROM@>2014    Crunch input line
451E : BS     GROM@>455F    Warning
4520 : CALL   GROM@>4D00    Scroll
4523 : ST     @>837F,>03    Line 3
4526 : INC    @>8311
4528 : CEQ    @>8310,@>8311
452B : BR     GROM@>455F    Warning
452D : DST    @>831E,@>8334    Data pointer
4530 : DST    @>8334,>0321
4534 : DST    @>8302,@>830E
4537 : DADD   @>8302,>0008

```

```

453B : DST      @>8306,VDP*>8302
453F : CALL     GROM@>495A      Fetch byte
4542 : CLOG     VDP*>8306,>80    Token?
4546 : BR       GROM@>4571      Yes, jump
4548 : CALL     GROM@>493B      Convert number if necessary
454B : BR       GROM@>4558
454D : CZ       @>8354
454F : BS       GROM@>4576
4551 : DST      @>8334,@>831E    Again old data pointer
4554 : CZ       @>8317          Screen flag
4556 : BR       GROM@>4564      No, jump
4558 : CZ       @>8317
455A : BS       GROM@>57E3
455C : DST      @>8334,@>831E
455F : CALL     GROM@>284C      Warning
4562 : DATA    >21            Input error
4563 : DATA    >28
4564 : MOVE     >000B TO VDP@>02E2 FROM GROM@>2108
456B : DST      @>8308,>02ED
456F : BR       GROM@>44EF      Once again
4571 : CALL     GROM@>496C      Fetch length byte
4574 : BS       GROM@>4558      Error
4576 : CALL     GROM@>495A      Fetch byte on >8301
4579 : CEQ     @>8301,>B3      Token ,?
457C : BS       GROM@>4588
457E : DEC     @>8311          More variables ?
4580 : BR       GROM@>4558      Error
4582 : CZ       @>8301
4584 : BR       GROM@>4558      Error
4586 : BR       GROM@>458C
4588 : DEC     @>8311          Number of variables
458A : BR       GROM@>4537
458C : DST      @>8334,>0321    New data pointer
4590 : DST      @>832C,@>830A
4593 : DDEC     @>832C
4595 : DST      @>836E,@>830E    Old stack pointer
4598 : XML     >1B            Fetch byte
459A : CZ       @>8342          End of line?
459C : BS       GROM@>45C2
459E : CALL     GROM@>4CEA      Build stack entry
45A1 : XML     >17            VPUSHG
45A3 : CALL     GROM@>495A      Length byte on >8301
45A6 : CEQ     @>834C,>65      String tag
45A9 : BS       GROM@>45B0
45AB : CALL     GROM@>493B      Convert in number if necessary
45AE : BS       GROM@>45B9
45B0 : CALL     GROM@>496C      Fetch length byte
45B3 : DST      @>830C,@>8350    Length of string
45B6 : CALL     GROM@>492D      String to new place
45B9 : XML     >15            Submit value to variable
45BB : CALL     GROM@>495A
45BE : CZ       @>8342
45C0 : BR       GROM@>4598
45C2 : DST      @>8334,@>831E    Pointer data
45C5 : CONT

45C6 : ST       @>8321,VDP@>0003(@>8304) Internal offset
45CB : CLR      @>8320
45CD : DADD     @>8320,VDP@>0006(@>8304) Plus buffer pointer on start screen
45D2 : RTN

45D3 : DCH     @>8308,>02FD      Screen line is full
45D7 : BR       GROM@>45E0      No, go on
45D9 : CALL     GROM@>4D00      Scroll
45DC : DST      @>8308,>02E2      New start
45E0 : RTN

```

```

45E1 : XML    >1B           Fetch byte

Basic READ:
45E3 : CALL   GROM@>4CEA     Build stack entry of variable name
45E6 : XML    >17           VPUSHG
45E8 : CEQ    @>8334,>FF     Data pointer >FF?
45EB : BS     GROM@>57E8     Yes, data error
45ED : CALL   GROM@>4952     Fetch byte from VDP on >8301
45F0 : CEQ    @>834C,>65     String tag?
45F3 : BS     GROM@>4614
45F5 : CEQ    @>8301,>C8     String without " " ?
45F8 : BR     GROM@>57E8     No, error
45FA : CALL   GROM@>4978     Prepare pointer
45FD : DINC   @>8350
45FF : CALL   GROM@>4D02     Move string from program to string part
4602 : DST    @>8356,@>831C
4605 : DADD   @>831C,@>8350   Length
4608 : DDEC   @>831C
460A : CALL   GROM@>4D16     Convert string in number
460D : DCEQ   @>8356,@>831C
4610 : BR     GROM@>57E8     Error
4612 : BR     GROM@>461C
4614 : CALL   GROM@>496C     Fetch string length
4617 : BS     GROM@>57E8     Not found, error
4619 : CALL   GROM@>4D02     Move string from program part to string area
461C : XML    >15           Submit value to variable
461E : CALL   GROM@>4952     Fetch byte on >8301
4621 : CZ     @>8301         Line end ?
4623 : BR     GROM@>4636
4625 : DDECT  @>8336         New pointer to data line
4627 : ST     @>8334,>FF     Pointer to data element
462A : DCEQ   @>8336,@>8330   End of line list ?
462D : BS     GROM@>4634
462F : DDEC   @>8336         Prepare for subprogram
4631 : CALL   GROM@>4D08     Set new data pointer
4634 : BR     GROM@>463B
4636 : CEQ    @>8301,>B3     Is a comma in data line?
4639 : BR     GROM@>57E8     No, error
463B : CEQ    @>8342,>B3     Is a comma in program line?
463E : BS     GROM@>45E1     Yes, the same once again
4640 : CONT

```

```

Basic LOAD:
4641 : CALL   GROM@>4888     Build PAB, reset all Basic pointers
4644 : DST    @>830A,@>8304   Save PAB address
4647 : DADD   @>830A,VDP@>000C(@>8304) Plus name length
464C : DADD   @>830A,>000A     Plus 10 for PAB length
4650 : DST    VDP@>000A(@>8304),@>8370 Top memory
4655 : DSUB   VDP@>000A(@>8304),@>830A Minus seized one up to now, equal length
465A : DINC   VDP@>000A(@>8304) +1
465E : DST    VDP@>0006(@>8304),@>830A PAB buffer
4663 : ST     VDP@>0004(@>8304),>05 Load op code
4668 : CALL   GROM@>4CC6     DSRLNK
466B : BR     GROM@>46DA     Error
466D : DST    @>8302,VDP@>0002(@>830A) Check sum
4672 : DXOR   @>8302,VDP@>0004(@>830A)
4677 : DCEQ   VDP*>830A,@>8302
467B : BR     GROM@>46DA     Wrong, then error
467D : DST    @>8332,VDP@>0002(@>830A) End of line list (high address)
4682 : DST    @>8330,VDP@>0004(@>830A) Start of line list (low address)
4687 : DST    @>8302,VDP@>0006(@>830A) Old top memory
468C : DADD   @>830A,>0008     Begin program
4690 : DSUB   @>8332,@>8302     Line pointer minus old top memory
4693 : DSUB   @>8330,@>8302
4696 : DSUB   @>830A,@>8330

```

```

4699 : DCLR @>8302
469B : DSUB @>8302,@>8330 Calc. number
469E : DINC @>8302
46A0 : DST @>8304,@>8370 New top memory
46A3 : DADD @>8332,@>8370 Line pointer plus new top memory
46A6 : DADD @>8330,@>8370
46A9 : ST VDP*>8304,VDP*>830A
46AE : DDEC @>830A Load address
46B0 : DDEC @>8304 Goal address
46B2 : DDEC @>8302 Number
46B4 : BR GROM@>46A9 Loop till all shifted
46B6 : CALL GROM@>215A Reset Basic pointer
46B9 : DDEC @>830A Convert line list
46BB : DST @>8302,VDP*>830A Fetch pointer to line
46BF : DSUB @>8302,@>8370 Minus new top memory
46C2 : BS GROM@>4745 0, then end
46C4 : DST @>830A,@>8330 Start line list
46C7 : DCHE @>8332,@>830A Still under end of line list?
46CA : BR GROM@>4745 No, end
46CC : DINCT @>830A +2
46CE : DSUB VDP*>830A,@>8302 Minus difference
46D2 : AND VDP*>830A,>7F Bit 0 reset if necessary
46D6 : DINCT @>830A Next line
46D8 : BR GROM@>46C7 Go on till end
46DA : CALL GROM@>284C Warning
46DD : DATA >46 Text
46DE : DATA >E4
46DF : CALL GROM@>215A Reset Basic pointer
46E2 : BR GROM@>57C3 I/O Error

```

```

46E4 : DATA >17,>A3,>A8,>A5,>A3,>AB,>80,>B0,>B2,>AF,>A7,>B2,>A1,>AD,>80,>A9
>AE,>80,AD,>A5,>AD,>AF,>B2,>B9 Text: Check Program in Memory

```

Basic SAVE:

```

46FC : CALL GROM@>4888 Build PAB
46FF : DST @>830A,@>8330 Start line table
4702 : DINCT @>830A +2
4704 : AND VDP*>830A,>7F Reset bit 0
4708 : DADD @>830A,>0004 Next line
470C : DCH @>830A,@>8332 End reached?
470F : BR GROM@>4704 No, loop
4711 : DST @>830A,@>8340 Pointer free space symbol table
4714 : DDEC @>830A -1
4716 : DST VDP*>830A,@>8370 Top memory
471A : DDECT @>830A
471C : DST VDP*>830A,@>8330 Start line list
4720 : DDECT @>830A
4722 : DST VDP*>830A,@>8332 End line list
4726 : DDECT @>830A
4728 : DST VDP*>830A,@>8330 Build check sum
472C : DXOR VDP*>830A,@>8332
4730 : DST VDP@>0006(@>8304),@>830A Set to start
4735 : DDEC @>830A
4737 : DST VDP@>000A(@>8304),@>8370 Calc. number of bytes
473C : DSUB VDP@>000A(@>8304),@>830A
4741 : CALL GROM@>4CB9 DSR access
4744 : DATA >06 Save op code
4735 : ST @>8388,>20 Set edit mode
4749 : B GROM@>2012 Go on in Basic

```

Basic LIST:

```

474C : DCLR @>8314
474E : DCLR @>831E
4750 : ST @>8308,>2D Decimal 45
4753 : CALL GROM@>2834 Line number
4756 : DCZ @>8314 Line number 0?

```

```

4758 : BR      GROM@>4768
475A : DST      @>8314,VDP@>FFFD(@>8332)  Error value:1st line from line list
4760 : DCZ      @>831E      Step 0, in this case end
4762 : BR      GROM@>4768
4764 : DST      @>831E,VDP*>8330 Error value: Last line
4768 : DCZ      @>831E      0?
476A : BR      GROM@>477A
476C : DDEC     @>8320      Start on screen
476E : CEQ      VDP*>8320,>80   Space?
4772 : BS      GROM@>476C
4774 : CEQ      VDP*>8320,>8D   Hyphen?
4778 : BS      GROM@>4764      Yes, last line is error value
477A : DCHE     @>831E,@>8314  End smaller than start
477D : BS      GROM@>4782
477F : DST      @>831E,@>8314  Otherwise end=start
4782 : DST      @>8344,@>8314  Line number on >8344
4785 : CALL     GROM@>283E      Find line number
4788 : DST      @>8314,@>832E  Pointer to line number
478B : DST      @>8344,@>831E  End
478E : CALL     GROM@>283E      Find line number
4791 : DCH      VDP*>832E,@>831E Does this line exist?
4795 : BR      GROM@>479B
4797 : DADD     @>832E,>0004    Pointer again on last line
479B : DST      @>831E,@>832E  Pointer on last line
479E : DDEC     @>832C      Pointer in line minus 1
47A0 : XML      >1B          Fetch byte
47A2 : CZ      @>8342
47A4 : BS      GROM@>47F3      Go on
47A6 : CALL     GROM@>41CF      Close all open files
47A9 : DST      @>836E,>06F8    Reset value stack
47AD : DST      @>8324,@>836E
47B0 : XML      >1B          Fetch byte
47B2 : DST      @>8304,>0708    PAB pointer
47B6 : CLR      @>8317
47B8 : MOVE     >000D TO VDP*>8304 FROM GROM@>481E  Write PAB
47BF : DST      @>8308,>0715
47C3 : ST       @>834C,@>8342    Length byte
47C6 : INC      @>834C
47C8 : ST       VDP*>8308,@>8342 Write name into PAB
47CC : XML      >1B          Fetch byte
47CE : DINC     @>8308
47D0 : DEC      @>834C      Length 0?
47D2 : BR      GROM@>47C8      Go on
47D4 : CALL     GROM@>4CC0      Open file
47D7 : CLR      @>834A
47D9 : ST       @>8307,VDP@>0008(@>8304) Record length
47DE : ST       @>834B,@>8307
47E1 : DADD     @>834A,@>8308
47E4 : DST      VDP@>0006(@>8304),@>8308 Buffer address
47E9 : DCH      @>834A,@>8330    Enough space?
47EC : BS      GROM@>57D6      No, error
47EE : ST       @>8306,>01      Last length 1
47F1 : BR      GROM@>47F9
47F3 : ST       @>837F,>1F      Last row XPT
47F6 : CALL     GROM@>4C9B      Screen pointer for output
47F9 : CALL     GROM@>282E      List line
47FC : SCAN     Keyboard scanning
47FD : BR      GROM@>4804
47FF : CEQ      @>8375,>02      Clear key?
4802 : BS      GROM@>480D      Yes, end
4804 : DSUB     @>8314,>0004      Next line
4808 : DCH      @>831E,@>8314    Till above the last?
480B : BR      GROM@>47F9      Yes, go on
480D : CZ      @>8317      Screen flag
480F : BR      GROM@>481B      Return Basic by clearing
4811 : CALL     GROM@>4BFC      Write data set

```

```

4814 : CALL   GROM@>4CB9      Call DSR
4817 : DATA  >01            Close op code
4818 : B       GROM@>201A
481B : B       GROM@>2012
481E : DATA  >0000
4820 : DATA  >0000
4822 : DATA  >0012
4824 : DATA  >0000
4826 : DATA  >0000
4828 : DATA  >0000
482A : DATA  >60

Basic EOF:
482B : CEQ   @>8342,>B7      Token (?)
482E : BR    GROM@>40F5      No, error
4830 : PARS  >FF            Go on till end
4832 : CALL  GROM@>499C      Convert into integer
4835 : BS    GROM@>4D7C      Bad value error
4837 : CZ    @>834A          0?
4839 : BR    GROM@>4D7C      Error
483B : DST   @>835C,@>833C  Pointer on first PAB
483E : CZ    @>835C          0?
4840 : BS    GROM@>57DE      Yes, error
4842 : CEQ   VDP@>0002(@>835C),@>834B Right number ?
4847 : BS    GROM@>484F      Yes, found
4849 : DST   @>835C,VDP*>835C Pointer next PAB
484D : BR    GROM@>483E      Go on searching
484F : DEX   @>8304,@>835C  PAB pointer on >8304 for subprogram
4852 : ST    @>835E,>09      Op code for status
4855 : EX    VDP@>0004(@>8304),@>835E Op code in PAB
485A : CALL  GROM@>4CC0      Call DSR
485D : DEX   @>8304,@>835C  Old PAB pointer again
4860 : ST    VDP@>0004(@>835C),@>835E Old op codes again
4865 : ST    @>835E,VDP@>000C(@>835C) Fetch status
486A : MOVE  >0008 TO @>834A FROM GROM@>4880 1 on FAC
4870 : CLOG  @>835E,>03      End of file?
4873 : BS    GROM@>487D      No end, then 0
4875 : CLOG  @>835E,>02      Physical end of file?
4878 : BS    GROM@>487C      No, end
487A : DNEG  @>834A          Yes, -1
487C : CONT
487D : DCLR  @>834A          0
487F : CONT

4880 : DATA  >4001
4882 : DATA  >0000
4884 : DATA  >0000
4886 : DATA  >0000

4888 : CLR    @>8388          Clear flag byte
488B : CEQ   @>8342,>C7      String in ""?
488E : BS    GROM@>4895      Yes, jump
4890 : CEQ   @>8342,>C8      String without ""?
4893 : BR    GROM@>40F5      No, then error
4895 : CALL  GROM@>41CF      Close all open files
4898 : DST   @>836E,>06F8    New value stack pointer
489C : DST   @>8324,@>836E    New start value stack
489F : CALL  GROM@>215A      Reset all Basic pointer
48A2 : DST   @>8304,>0700    PAB pointer
48A6 : CLR   VDP*>8304      Clear RAM for PAB
48A9 : MOVE  >0009 TO VDP@>0001(@>8304) FROM VDP*>8304
48B1 : XML   >1B            Fetch byte
48B3 : DSUB  @>8304,>0004    +4 (that subprogram for Basic are OK)
48B7 : ST    VDP@>000D(@>8304),@>8342 Length byte
48BC : DST   @>830A,VDP@>000C(@>8304) Whole length
48C1 : MOVE  @>830A TO VDP@>000E(@>8304) FROM VDP*>832C Name in VDP

```

```

48C8 : ST      @>8334,>FF      Flag for DATA
48CB : RTN

48CC : DST      @>830A,@>832C    Text pointer
48CF : CLR      @>8310
48D1 : DST      @>830E,@>836E    Save value stack pointer
48D4 : CHE      @>8342,>80      Token?
48D7 : BS      GROM@>40F5      Yes, error
48D9 : XML      >13           Fetch variable
48DB : CLR      @>8311
48DD : CEQ      @>8342,>B7      Token (
48E0 : BR      GROM@>48E4
48E2 : INC      @>8311          Counter for paranthesis
48E4 : CZ      @>8311          No paranthesis
48E6 : BS      GROM@>48F7      Then jump
48E8 : CZ      @>8342          End of line?
48EA : BS      GROM@>40F5      Error
48EC : CEQ      @>8342,>B6      Token )
48EF : BR      GROM@>48F3
48F1 : DEC      @>8311          Counter minus 1
48F3 : XML      >1B           Fetch byte
48F5 : BR      GROM@>48DD      Go on till of paranthesis
48F7 : XML      >17           VPUSHG
48F9 : INC      @>8310          Counter for start
48FB : CZ      @>8342          End of line?
48FD : BS      GROM@>490E      Yes, end
48FF : CEQ      @>8342,>B3      Token ,?
4902 : BR      GROM@>40F5      Error
4904 : XML      >1B           Fetch byte
4906 : CZ      @>8342          End of line
4908 : BR      GROM@>48D4      From start
490A : CZ      @>8317          Flag for file i.e. screen offset
490C : BR      GROM@>40F5      Error
490E : RTN

String stack entry:
490F : DST      @>834C,>6500     As of now treat also number as string
4913 : DST      @>8350,@>830C     Length
4916 : MOVE     >001A TO VDP@>03C0 FROM @>8352   Save ARG
491C : CALL     GROM@>4D12       Fetch space for string
491F : MOVE     >001A TO @>8352 FROM VDP@>03C0   Repair ARG
4925 : DST      @>834E,@>831C     Address string
4928 : DST      @>834A,>001C     Printing
492C : RTN

492D : CALL     GROM@>490F       Build string entry
4930 : CZ      @>8351            0 string?
4932 : BS      GROM@>493A       Yes, end
4934 : MOVE     @>830C TO VDP*>831C FROM VDP*>8366   Shift string to new space
493A : RTN

493B : CEQ      @>8301,>C8       String in ""?
493E : BR      GROM@>4951       No, end
4940 : CALL     GROM@>495A       Fetch byte on >8301
4943 : DST      @>8356,@>8334     DATA pointer
4946 : CLR      @>8300
4948 : DADD     @>8334,@>8300     New DATA pointer
494B : CALL     GROM@>4D16       Convert string into number
494E : DCEQ     @>8356,@>8334
4951 : RTNC

4952 : ST      @>834D,@>8389     GROM flag
4956 : CZ      @>834D
4958 : BR      GROM@>4962       Jump, if in GROM
495A : ST      @>8301,VDP*>8334   Byte nach DATA-Pointer aus VDP holen
495E : CLR      @>834D

```



```

4960 : BR      GROM@>4969
4962 : MOVE    >0001 TO @>8301 From GROM@>0000(@>8334)
4969 : DINC    @>8334      Increase pointer
496B : RTN

496C : DCLR    @>8350
496E : CEQ    @>8301,>C7      String in ""?
4971 : BS     GROM@>4978
4973 : CEQ    @>8301,>C8      String without ""?
4976 : BR     GROM@>4987      No, jump
4978 : CALL   GROM@>4956      Fetch byte on >8301
497B : CLR    @>8350      Length
497D : ST     @>8351,@>8301    Length is now word
4980 : DST    @>8366,@>8334    Data pointer
4983 : DADD   @>8334,@>8350    Address end of strings
4986 : RTN
4987 : CEQ    @>8301,>B3      Token ,?
498A : BS     GROM@>4990      Yes, jump
498C : CZ     @>8301      End of line?
498E : BR     GROM@>49CC      No, jump to end with condition bit set
4990 : DDEC   @>8334      Data pointer return again
4992 : RTN

4993 : CEQ    @>8342,>FD      Toke #?
4996 : BR     GROM@>40F5      No, end
4998 : XML    >1B      Fetch byte
499A : PARS   >B5      Go on till :
499C : CEQ    @>834C,>65      String?
499F : BS     GROM@>4D81      Yes, error
49A1 : CLR    @>8354
49A3 : XML    >12      Convert floating point into integer
49A5 : CZ     @>8354      Error?
49A7 : BR     GROM@>4D7C      Yes, jump
49A9 : CLOG   @>834A,>80      Bit 0 set
49AC : BR     GROM@>49CC      Yes, end with condition bit set
49AE : DCZ    @>834A      0, then condition bit set
49B0 : RTNC

49B1 : ST     @>8317,@>834B    Number on >8317
49B4 : CZ     @>834A      Smaller than 256?
49B6 : BR     GROM@>4D7C      No, error
49B8 : DST    @>8304,@>833C    PAB pointer
49BB : DCZ    @>8304      No PAB?
49BD : BS     GROM@>49CF      Then return condition bit reset
49BF : CEQ    VDP@>0002(@>8304),@>8317    Number OK?
49C4 : BS     GROM@>49CC      Yes, then return with condition bit set
49C6 : DST    @>8304,VDP*>8304    Pointer to next PAB
49CA : BR     GROM@>49BB      From start
49CC : CEQ    @>8300,@>8300    Set condition bit
49CF : RTNC

49D0 : CLR    @>8317
49D2 : CEQ    VDP@>0004(@>8304),>03    Write op code?
49D7 : BR     GROM@>49E5      No, end
49D9 : CZ     VDP@>0003(@>8304)    Length of data block 0?
49DD : BS     GROM@>49E5      Yes, end
49DF : CALL   GROM@>4C2A      Fetch PAB pointer
49E2 : CALL   GROM@>4BFC      DSR access
49E5 : RTN

Clear PAB ( or entry in symbol table ) :
49E6 : DST    @>830A,VDP@>0006(@>8304)    PAB buffer pointer
49EB : DDEC   @>830A      -1
49ED : CLR    @>8308
49EF : ST     @>8309,VDP@>000D(@>8304)    Length of name
49F4 : ADD    @>8309,>0D      Whole PAB

```

```

49F7 : DADD @>8308,@>8304 Pointer to end >8308
49FA : DCEQ @>833C,@>8304 Pointer equal PAB pointer
49FD : BS GROM@>4A26
49FF : DST @>8302,@>833C Save PAB pointer
4A02 : DCEQ VDP*>8302,@>8304 Clear 2nd PAB ?
4A06 : BS GROM@>4A0E Yes, go on
4A08 : DST @>8302,VDP*>8302 No, next
4A0C : BR GROM@>4A02
4A0E : DST VDP*>8302,VDP*>8304 Change pointer
4A13 : DCZ VDP*>8302 Pointer now 0?
4A16 : BS GROM@>4A20
4A18 : DADD VDP*>8302,@>8308 Put as high as necessary
4A1C : DSUB VDP*>8302,@>830A
4A20 : DST @>8304,VDP*>8302
4A24 : BR GROM@>4A37
4A26 : DST @>833C,VDP*>8304 New PAB pointer
4A2A : DCZ @>833C Now 0?
4A2C : BS GROM@>4A34
4A2E : DADD @>833C,@>8308 Increase pointer for area of cleared PAB
4A31 : DSUB @>833C,@>830A
4A34 : DST @>8304,@>833C Now 1st PAB
4A37 : DST @>8302,@>830A Address of PAB to be cleared
4A3A : DSUB @>8302,@>8340 Minus old pointer on free space
4A3D : DST @>8306,@>8308 End of PAB to be cleared
4A40 : DCZ @>8302 0?
4A42 : BS GROM@>4A51
4A44 : ST VDP*>8308,VDP*>830A Shift RAM
4A49 : DDEC @>830A
4A4B : DDEC @>8308
4A4D : DDEC @>8302
4A4F : BR GROM@>4A40 Loop till end
4A51 : DSUB @>8308,@>830A Difference
4A54 : DCZ @>8304 0?
4A56 : BS GROM@>4A71
4A58 : DCZ VDP*>8304 Pointer next PAB 0?
4A5B : BS GROM@>4A6C
4A5D : DADD VDP*>8304,@>8308 Increase pointer by difference
4A61 : DADD VDP@>0006(@>8304),@>8308 Pointer on buffer too
4A66 : DST @>8304,VDP*>8304 Do this for next PAB
4A6A : BR GROM@>4A58 Till all
4A6C : DADD VDP@>0006(@>8304),@>8308 Increase pointer buffer for first PAB
4A71 : DCZ @>833E Pointer to symbol table 0?
4A73 : BS GROM@>4AF5 Yes, end
4A75 : DCGE @>833E,@>8306 Higher than cleared PAB?
4A78 : BS GROM@>4AF5 Yes, end
4A7A : DADD @>833E,@>8308 Plus difference
4A7D : DST @>8304,@>833E
4A80 : CZ @>8389 In GROM?
4A83 : BR GROM@>4A8C
4A85 : DCGE VDP@>0004(@>8304),@>8330 Start line list with pointer to value
4A8A : BS GROM@>4A91
4A8C : DADD VDP@>0004(@>8304),@>8308 Increase pointer to value
4A91 : CGE VDP*>8304,>00 String?
4A95 : BS GROM@>4ADC No, then jump
4A97 : ST @>834A,>07 Check on data field
4A9A : AND @>834A,VDP*>8304
4A9E : DST @>834C,@>8304 PAB pointer
4AA1 : DADD @>834C,>0006 Plus 6
4AA5 : DST @>8350,>0001 1
4AA9 : CLR @>834E
4AAB : CZ @>834A No data field?
4AAD : BS GROM@>4AC3 Then jump
4AAF : ST @>834F,>01
4AB2 : SUB @>834F,@>8343 Option base
4AB5 : DADD @>834E,VDP*>834C Dimensions
4AB9 : DMUL @>834E,@>8350 Multiply

```

```

4ABC : DEC      @>834A          Dimensions minus 1
4ABE : DINCT   @>834C
4AC0 : B        GROM@>4AAB      Till all
4AC3 : DCZ     @>8350
4AC5 : BS      GROM@>4ADC
4AC7 : DST     @>834A,VDP*>834C Fetch link address
4ACB : DCZ     @>834A          0?
4ACD : BS      GROM@>4AD5      Then jump
4ACF : DST     VDP@>FFFF(@>834A),@>834C Write new link address
4AD5 : DDEC    @>8350          -1
4AD7 : DINCT   @>834C          +2
4AD9 : B        GROM@>4AC3      Go on
4ADC : DCZ     VDP@>0002(@>8304) More entrys into symbol table?
4AE0 : BS      GROM@>4AF5      No, end
4AE2 : DCGE    VDP@>0002(@>8304),@>8306 Higher?
4AE7 : BS      GROM@>4AF5      Yes, then end
4AE9 : DADD    VDP@>0002(@>8304),@>8308 Increase link pointer
4AEE : DST     @>8304,VDP@>0002(@>8304) New pointer
4AF3 : BR      GROM@>4A80      The same once again
4AF5 : DADD    @>8340,@>8308    New pointer free space for symbol table
4AF8 : RTN

4AF9 : CALL    GROM@>499C      Convert floating point into integer
4AFC : BR      GROM@>4B02
4AFE : DST     @>834A,>0001     If error 1
4B02 : RTN

4B03 : CEQ     @>8342,>B3       Token ,?
4B06 : BR      GROM@>4B2E      No, go on
4B08 : XML     >1B             Fetch byte
4B0A : CEQ     @>8342,>DE       Token REC?
4B0D : BR      GROM@>40F5      No, error
4B0F : CLOG    VDP@>0005(@>8304),>01 Relative file?
4B14 : BS      GROM@>57DE      No, error
4B16 : XML     >1B             Fetch byte
4B18 : CALL    GROM@>49D0      Write data set PAB if necessary
4B1B : CLR     VDP@>0003(@>8304) Internal offset
4B1F : PARS    >B5             Go on till :
4B21 : CALL    GROM@>499C      CFI
4B24 : CLOG    @>834A,>80       Negative
4B27 : BR      GROM@>4D7C      Error
4B29 : DST     VDP@>000A(@>8304),@>834A Write data block number in PAB
4B2E : RTN

4B2F : CZ      @>8342          End of line?
4B31 : BS      GROM@>49CC      Return with condition bit set
4B33 : CEQ     @>8342,>C7       String in ""?
4B36 : BS      GROM@>4B3D
4B38 : CEQ     @>8342,>C8       String without ""?
4B3B : BR      GROM@>4B49
4B3D : XML     >1B             Fetch length byte
4B3F : ST      @>834B,@>8342
4B42 : CLR     @>834A          Length as word on FAC
4B44 : DADD    @>832C,@>834A    Plus text pointer
4B47 : BR      GROM@>4B50      End
4B49 : CEQ     @>8342,>C9       Token line number ?
4B4C : BR      GROM@>4B50      No, end
4B4E : DINCT   @>832C          Skip
4B50 : XML     >1B             Fetch byte
4B52 : RTN

4B53 : ST      @>830D,@>8356    Length
4B56 : CLR     @>830C
4B58 : CALL    GROM@>490F      Tread as string
4B5B : MOVE    @>830C TO VDP*>831C FROM *>8355 In VDP
4B61 : RTN

```

```

4B62 : DIV    @>834A,@>834C
4B65 : CZ     @>834B           Rest 0
4B67 : BR     GROM@>4B6C
4B69 : ST     @>834B,@>834C   Rest on >834C
4B6C : CLR    @>834A           FAC 0
4B6E : RTN

4B6F : CZ     @>8342           End of line?
4B71 : BR     GROM@>4B78       No, jump
4B73 : DST    *>8373,>4325    Trick return routine address
4B78 : CHE    @>8342,>B3       Greater or equal token ,?
4B7B : BR     GROM@>4BA0       End
4B7D : CH     @>8342,>B5       Greater token :
4B80 : BS     GROM@>4BA0       End
4B82 : DST    *>8373,>431D    Trick return routine address
4B87 : CALL   GROM@>433A       Check internal
4B8A : BR     GROM@>4BA0
4B8C : CEQ    @>8342,>B3       Token ,?
4B8F : BR     GROM@>4B96
4B91 : DST    *>8373,>42FF    Trick return routine address
4B96 : CEQ    @>8342,>B5       Token :?
4B99 : BR     GROM@>4BA0
4B9B : DST    *>8373,>431A    Trick return routine address
4BA0 : RTN    Return

```

Build PAB (rough) :

```

4BA1 : PARS  >B3           Fetch name
4BA3 : CEQ   @>834C,>65     String?
4BA6 : BR    GROM@>4D81     No, error
4BA8 : DST   @>8302,@>8350   Length on >8302
4BAB : ADD   @>8303,>0E     Complete PAB length
4BAE : XML   >17           VPUSHG
4BB0 : DST   @>834A,@>8302   Length on FAC
4BB3 : CALL  GROM@>2844     Fetch space for PAB
4BB6 : XML   >18           VPOP
4BB8 : DSUB  @>8340,@>8302   Free space minus length
4BBB : DST   @>8304,@>8340   Save on >8304 (New pointer to PAB)
4BBE : DINC  @>8304         +1
4BC0 : CLR   VDP*>8304     Clear PAB
4BC3 : MOVE  >000D TO VDP@>0001(@>8304) FROM VDP*>8304
4BCB : ST    VDP@>0003(@>8304),@>8303 Internal offset
4BD0 : ST    @>8302,@>8351   Length
4BD3 : ST    VDP@>000D(@>8304),@>8351 Length of name
4BD8 : ST    VDP@>0002(@>8304),@>8317 Number
4BDD : DST   @>8308,@>8304
4BE0 : DADD  @>8308,>000E    Pointer to name
4BE4 : CLR   @>8317
4BE6 : CZ    @>8302           Length of name 0?
4BE8 : BS    GROM@>4BFB     Return no error, since also used for screen
4BEA : ST    VDP*>8308,@>8317 Clear area for name
4BEE : ADD   VDP*>8308,VDP*>834E Write name
4BF3 : DINC  @>834E         +1
4BF5 : DINC  @>8308         +1
4BF7 : DEC   @>8302         -1
4BF9 : BR    GROM@>4BE6     Name written?
4BFB : RTN

4BFC : CZ    @>8317           Screen flag (offset)?
4BFE : BS    GROM@>4C08     No, jump
4C00 : CALL  GROM@>4D00     Scroll
4C03 : ST    @>8306,>01     Start
4C06 : BR    GROM@>4CAF     Go on
4C08 : CLOG  VDP@>0005(@>8304),>10 variables
4C0D : BR    GROM@>4C17     Yes, jump

```

```

4C0F : ST      @>8303,@>8307   Length of data set
4C12 : INC      @>8303           +1
4C14 : CALL    GROM@>4C43      Fill with space if necessary
4C17 : DEC      @>8306
4C19 : ST      VDP@>0009(@>8304),@>8306   Length record
4C1E : CLR      VDP@>0003(@>8304)   Internal offset 0
4C22 : CALL    GROM@>4CB9        Write data block
4C25 : DATA    >03
4C26 : CLR      @>8309
4C28 : BR      GROM@>4C36        New pointer and end

4C2A : CLR      @>8317
4C2C : ST      @>8307,VDP@>0008(@>8304)   Record length
4C31 : ST      @>8309,VDP@>0003(@>8304)   Internal offset
4C36 : ST      @>8306,@>8309        New internal offset
4C39 : INC      @>8306           +1
4C3B : CLR      @>8308
4C3D : DADD    @>8308,VDP@>0006(@>8304)   Buffer address in >8308
4C42 : RTN

4C43 : CZ      @>8303
4C45 : BR      GROM@>4C4D
4C47 : CZ      @>8306           Internal offset 0?
4C49 : BR      GROM@>4C52        Yes, set new pointer
4C4B : BR      GROM@>4C6D        No, end
4C4D : CH      @>8303,@>8306
4C50 : BR      GROM@>4C6D
4C52 : SUB     @>8303,@>8306        Minus offset
4C55 : ADD     @>8306,@>8303        +
4C58 : ST      @>8302,@>8317        Screen offset
4C5B : CALL    GROM@>433A        Check display?
4C5E : BR      GROM@>4C63        No, jump
4C60 : ADD     @>8302,>20
4C63 : ST      VDP*>8308,@>8302   Fill space
4C67 : DINC    @>8308
4C69 : DEC     @>8303           Number
4C6B : BR      GROM@>4C63
4C6D : RTN

4C6E : ST      @>830C,@>8351        Length byte
4C71 : CZ      @>830C           0?
4C73 : BS      GROM@>4C9A        Yes, end
4C75 : ST      @>8302,@>8307        Length of data block
4C78 : SUB     @>8302,@>8306        Minus seized number
4C7B : INC     @>8302           +1
4C7D : CHE     @>8302,@>830C        String longer?
4C80 : BS      GROM@>4C8C        No, jump
4C82 : CEQ     @>8306,>01          No data until now
4C85 : BS      GROM@>4C8F        Divide
4C87 : CALL    GROM@>4BFC        Print data block
4C8A : BR      GROM@>4C75        The same once again
4C8C : ST      @>8302,@>830C        Length of string
4C8F : SUB     @>830C,@>8302
4C92 : ADD     @>8306,@>8302        New pointer in data set (internal offset)
4C95 : CALL    GROM@>4BE6        Write data
4C98 : BR      GROM@>4C71        From start, if not whole sentence
4C9A : RTN

4C9B : CLR      @>8304
4C9D : ST      @>8317,>60          Screen offset
4CA0 : ST      @>8306,>01          Start data set
4CA3 : CZ      @>837F           Line 0
4CA5 : BS      GROM@>4CAC
4CA7 : ST      @>8306,@>837F        Line pointer
4CAA : DECT    @>8306           -1
4CAC : ST      @>8307,>1C          Length>1C

```

```

4CAF : ST      @>8309,@>8306
4CB2 : CLR      @>8308
4CB4 : DADD     @>8308,>02E1      Pointer to screen address
4CB8 : RTN

4CB9 : FETC     @>8356           Fetch op code
4CBB : ST      VDP@>0004(@>8304),@>8356   Write op code
4CC0 : CALL     GROM@>4CC6       Call DSR
4CC3 : BR      GROM@>57C3       I/O error
4CC5 : RTN

4CC6 : ST      VDP@>000C(@>8304),>60   Screen offset
4CCB : MOVE     >001E TO VDP@>03C0 FROM @>834A   Save FAC
4CD1 : DST      @>8356,@>8304       Pointer for DSR
4CD4 : DADD     @>8356,>000D       Point to name
4CD8 : CALL     GROM@>0010       DSRLNK
4CDB : DATA    >08
4CDC : MOVE     >001E TO @>834A FROM VDP@>03C0   Repair FAC
4CE2 : BS      GROM@>4CE9       Error
4CE4 : CLOG     VDP@>0005(@>8304),>E0   Check error
4CE9 : RTNC                    Return, condition bit set if no error

4CEA : CHE      @>8342,>80
4CED : BS      GROM@>40F5
4CEF : XML      >13
4CF1 : XML      >14
4CF3 : RTN

4CF4 : DATA    >0000
4CF6 : DATA    >0000
4CF8 : DATA    >0000
4CFA : DATA    >0000
4CFC : DATA    >0000
4CFE : DATA    >0000

4D00 : BR      GROM@>56CD       Scroll one line
4D02 : BR      GROM@>5120       Move string from program part into string part
4D04 : BR      GROM@>4DB0       2nd entry point for Basic execution
4D06 : BR      GROM@>56BB       Fetch line number
4D08 : BR      GROM@>5613       Set subprogram data pointer
4D0A : BR      GROM@>5645       Convert integer into ASCII
4D0C : BR      GROM@>4DBF       CONT
4D0E : BR      GROM@>4E38       Break program
4D10 : BR      GROM@>4D8A       RUN
4D12 : BR      GROM@>515C       Fetch memory space for string
4D14 : BR      GROM@>55BB       Clear actual string
4D16 : BR      GROM@>56E1       CSN
4D18 : BR      GROM@>51A9       Garbage collection

```

Subprogram list:

```

4D1A : DATA    >4D24       Next entry g
4D1C : DATA    >3538       Routine address
4D1E : DATA    >05        Length of name
4D1F : TEXT     ':SOUND:'   name

4D24 : DATA    >4D2E
4D26 : DATA    >351C
4D28 : DATA    >05
4D29 : TEXT     ':CLEAR:'

4D2E : DATA    >4D38
4D30 : DATA    >5713
4D32 : DATA    >05
4D33 : TEXT     ':COLOR:'

4D38 : DATA    >4D42

```

```

4D3A : DATA >56EF
4D3C : DATA >05
4D3D : TEXT ':GCHAR:'

4D42 : DATA >4D4C
4D44 : DATA >360E
4D46 : DATA >05
4D47 : TEXT ':HCHAR:'

4D4C : DATA >4D56
4D4E : DATA >362A
4D50 : DATA >05
4D51 : TEXT ':VCHAR:'

4D56 : DATA >4D5F
4D58 : DATA >3643
4D5A : DATA >04
4D5B : TEXT ':CHAR:'

4D5F : DATA >4D67
4D61 : DATA >3708
4D63 : DATA >03
4D64 : TEXT ':KEY:'

4D67 : DATA >4D71
4D69 : DATA >3748
4D6B : DATA >05
4D6C : TEXT ':JOYST:'

4D71 : DATA >0000
4D73 : DATA >37BF
4D75 : DATA >06
4D76 : TEXT ':SCREEN:'

```

```

-----
4D7C : CALL GROM@>284E      Error
4D7F : DATA >20           Bad value
4D80 : DATA >64
4D81 : CALL GROM@>284E      Error
4D84 : DATA >20           String number mismatch
4D85 : DATA >7D
4D86 : BR GROM@>56D4
4D88 : BR GROM@>566C

```

Execution Basic:

```

4D8A : DDEC @>8320
4D8C : CALL GROM@>282C      Skip space
4D8F : BS GROM@>4DA3       No line number, jump
4D91 : CALL GROM@>283C      Fetch line number
4D94 : DCHE @>8320,@>832A  End of line?
4D97 : BR GROM@>5671       No, error
4D99 : CALL GROM@>283E      Search line number in line list
4D9C : BR GROM@>5682       Not found, error
4D9E : DST @>8334,@>832E  Pointer to line on data pointer
4DA1 : BR GROM@>4DAA
4DA3 : DST @>8334,@>8332  End of line list
4DA6 : DSUB @>8334,>0003
4DAA : DST @>8344,VDP*>8334 Fetch pointer to first line
4DAE : BR GROM@>56CD       Return with scroll of a line

```

Execution of Basic after prescan:

```

4DB0 : DCZ @>8344          Run flag?
4DB2 : BS GROM@>4DCD       No, direct mode
4DB4 : DST @>832E,@>8334  Data pointer on line pointer
4DB7 : DINCT @>832E       Plus 2
4DB9 : DST @>8336,@>8332  End line list on line pointer data
4DBC : CALL GROM@>5613     Set data pointer

```

```

Continue:
4DBF : AND    @>8388,>7F      Clear bit 0
4DC3 : ST     @>837F,>03      3rd column
4DC6 : DCH    @>832E,@>8330  Pointer to actual line lower than start of list?
4DC9 : BR     GROM@>4E5B     Yes, end
4DCB : BR     GROM@>4DD1
4DCD : DST    @>832C,>0320   Text pointer at direct mode
4DD1 : DST    @>8326,>4DEB   Return address for GPL and assembler
4DD5 : DST    @>8328,>4E84   Address of jump table Basic
4DD9 : DCZ    @>8344        Run flag?
4ddb : BS     GROM@>4DEA     Jump at direkt mode
4DDD : BACK   >03          Background color
4DDF : ST     VDP@>030F,>10  New color table
4DE3 : MOVE   >0010 TO VDP@>0310 FROM VDP@>030F
4DEA : EXEC   GROM@>4DEA     Execute Basic
4DEB : CASE   @>8323        Return, according to value execution of following
4DED : BR     GROM@>4E5B     Code 0, end of program          routines
4DEF : BR     GROM@>4E2E     1 for Break
4DF1 : BR     GROM@>4E01     2 for Trace
4DF3 : BR     GROM@>565C     3 for error codes
4DF5 : BR     GROM@>5689     4 Check memory full
4DF7 : BR     GROM@>5696     5 Involution routine
4DF9 : BR     GROM@>54CF     6 User defined function
4DFB : BR     GROM@>5156     7 Fetch space for string
4DFD : BR     GROM@>51F2     8 Add strings
4DFF : BR     GROM@>51A3     9 Garbage collection
Trace:
4E01 : CLR    @>8320
4E03 : ST     @>8321,@>837F  Pointer screen output
4E06 : DADD   @>8320,>02DF
4E0A : DCH    @>8320,>02F8
4E0E : BR     GROM@>4E17
4E10 : CALL   GROM@>56CD     Scroll
4E13 : DST    @>8320,>02E2
4E17 : ST     VDP*>8320,>9C  "<"
4E1B : DINC   @>8320
4E1D : CALL   GROM@>5645     Write line number
4E20 : ST     VDP*>8320,>9E  ">"
4E24 : DSUB   @>8320,>02DE
4E28 : ST     @>837F,@>8321
4E2B : DCLR   @>8322        Nothing else
4E2D : RTNB
Break:
4E2E : CZ     @>8389        GROM flag?
4E31 : BR     GROM@>4DEA     Yes, then go on in Basic, since not permitted
4E33 : AND    VDP*>832E,>7F  Remove break point
4E37 : SCAN   GROM@>4DEA     Keyboard scanning
4E38 : DST    VDP@>03EC,@>832E Save pointer for continue
4E3C : CZ     @>8389        GROM flag?
4E3F : BR     GROM@>4DC6     Yes, then continue
4E41 : CALL   GROM@>56CD     Scroll
4E44 : CALL   GROM@>0036     Bad tone
4E47 : MOVE   >0010 TO VDP@>02E2 FROM GROM@>20E9  Text "Breakpoint in"
4E4E : DST    @>8320,>02F2   Pointer for output of the line number
4E52 : CALL   GROM@>5645     Write line number
4E55 : CALL   GROM@>201C     Load VDP
4E58 : B      GROM@>2012     Basic return
End:
4E5B : DCZ    @>8344        Run flag?
4E5D : BS     GROM@>4E55     No, return to Basic
4E5F : CALL   GROM@>56CD     Scroll
4E62 : FMT
4E63 : ... XPT=>02
4E65 : ... YPT=>17
4E67 : ... '>8A,>8A,>80,>A4,>AF,>AE,>A5,>80,>8A,>8A'  Text "DONE"
4E72 : ... END FMT

```


4E73 : DST @>836E,@>8324 Clear value stack
 4E76 : DCLR @>8344 Clear program execution
 4E78 : CALL GROM@>4012 Close PABs
 4E7B : CALL GROM@>56CD Scroll
 4E7E : CZ @>8389 GROM?
 4E81 : BS GROM@>4E55 No, jump
 4E83 : RTN

4E84 : BR GROM@>4FB6 FOR
 4E86 : BR GROM@>5463 BREAK
 4E88 : BR GROM@>5479 UNBREAK
 4E8A : BR GROM@>5459 TRACE
 4E8C : BR GROM@>545E UNTRACE
 4E8E : BR GROM@>400E READ
 4E90 : BR GROM@>4004 PRINT
 4E92 : BR GROM@>50DB CALL
 4E94 : BR GROM@>5111 String in "
 4E96 : BR GROM@>400C RESTORE
 4E98 : BR GROM@>50C8 RANDOMIZE
 4E9A : BR GROM@>4006 INPUT
 4E9C : BR GROM@>4008 OPEN
 4E9E : BR GROM@>400A CLOSE
 4EA0 : BR GROM@>4F99 (
 4EA2 : BR GROM@>4FB2 +
 4EA4 : BR GROM@>4FA8 -
 4EA6 : BR GROM@>4ED1 ABS
 4EA8 : BR GROM@>4EDC ATN
 4EAA : BR GROM@>4EE2 COS
 4EAC : BR GROM@>4EE8 EXP
 4EAE : BR GROM@>4EEE INT
 4EB0 : BR GROM@>4EFA LOG
 4EB2 : BR GROM@>4F26 SGN
 4EB4 : BR GROM@>4F40 SIN
 4EB6 : BR GROM@>4F46 SQRT
 4EB8 : BR GROM@>4F4C TAN
 4EBA : BR GROM@>52BE LEN
 4EBC : BR GROM@>52EA CHR\$
 4EBE : BR GROM@>4F00 RND
 4EC0 : BR GROM@>4000 DISPLAY
 4EC2 : BR GROM@>4002 DELETE
 4EC4 : BR GROM@>524A SEG\$
 4EC6 : BR GROM@>531A STR\$
 4EC8 : BR GROM@>5349 VAL
 4ECA : BR GROM@>53A9 POS
 4ECC : BR GROM@>5306 ASC
 4ECE : B GROM@>401C EOF

Basic ABS:

4ED1 : CALL GROM@>57A6 Check token (
 4ED4 : PARS >CB Go on till ABS
 4ED6 : CALL GROM@>4F79 Check string tag
 4ED9 : DABS @>834A Set absolute value
 4EDB : CONT

Basic ATN:

4EDC : DST @>835C,>0032 Routine address ATN
 4EE0 : BR GROM@>4F50

Basic COS:

4EE2 : DST @>835C,>002C Routine address COS
 4EE6 : BR GROM@>4F50

Basic EXP:

4EE8 : DST @>835C,>0028 Routine address EXP
 4EEC : BR GROM@>4F50

```

Basic INT:
4EEE : CALL GROM@>57A6      Check token (
4EF1 : PARS >CF             Fetch quotation
4EF3 : CALL GROM@>4F79      String tag?
4EF6 : CALL GROM@>0022      Call INT routine
4EF9 : CONT

Basic LOG:
4EFA : DST @>835C,>002A     Routine address LOG
4EFE : BR GROM@>4F50

Basic RND:
4F00 : ST @>834A,>3F        Exponent
4F03 : ST @>8310,>4B        Loop counter
4F06 : RAND >63            till 100
4F08 : CZ @>8378           0?
4F0A : BR GROM@>4F16       No, go on
4F0C : DEC @>834A          -1
4F0E : CZ @>834A          0?
4F10 : BS GROM@>4F23       End with 0
4F12 : BR GROM@>4F06       Go on
4F14 : RAND >63            till 100
4F16 : ST *>8310,@>8378    All digits
4F1A : CEQ @>8310,>51      Till >8351
4F1D : BS GROM@>4F25
4F1F : INC @>8310          Increase loop counter
4F21 : BR GROM@>4F14
4F23 : CLR @>834B         Set 0
4F25 : CONT

Basic SGN
4F26 : CALL GROM@>57A6      Check token (
4F29 : PARS >D1           Fetch quotation
4F2B : CALL GROM@>4F79      String tag?
4F2E : DCZ @>834A         0?
4F30 : BS GROM@>4F3F       End
4F32 : CLOG @>834A,>80     Sign (+ -)?
4F35 : MOVE >0008 TO @>834A FROM GROM@>50C0  Fetch the 1 on FAC
4F3B : BS GROM@>4F3F       Positive end
4F3D : DNEG @>834A        -1
4F3F : CONT

Basic SIN:
4F40 : DST @>835C,>002E     Routine address SIN
4F44 : BR GROM@>4F50

Basic SQR:
4F46 : DST @>835C,>0026     Routine address SQR
4F4A : BR GROM@>4F50

Basic TAN:
4F4C : DST @>835C,>0030     Routine address TAN
4F50 : CALL GROM@>4F7F      Check memory
4F53 : CALL GROM@>57A6      Check token (
4F56 : INCT @>8373         Return on substack
4F58 : DST *>8373,>4F6B
4F5D : INCT @>8373         Routine address on substack
4F5F : DST *>8373,@>835C
4F63 : PARS >FF           Fetch quotation
4F65 : CALL GROM@>4F79      String tag?
4F68 : CLR @>8354
4F6A : RTN                Execute routine
4F6B : CZ @>8354           Error?
4F6D : BS GROM@>4FA7       No, CONT
4F6F : CH @>8354,>01
4F72 : BR GROM@>56B5       Number too big

```

4F74 : CALL GROM@>284E
 4F77 : DATA >20 Bad argument
 4F78 : DATA >94

Check for string tag:

4F79 : CEQ @>834C,>65 String tag?
 4F7C : BS GROM@>4D81 Yes, error
 4F7E : RTN

Memory check:

4F7F : CHE @>8373,>B2 Substack not too high?
 4F82 : BS GROM@>567D Yes, error memory full
 4F84 : DST @>835E,@>836E Value stack pointer on >835E
 4F87 : DADD @>835E,>0028 + >28
 4F8B : DCHE @>835E,@>831A Enough space?
 4F8E : BR GROM@>4F98 Yes, end
 4F90 : CALL GROM@>51A9 Garbage collection
 4F93 : DCHE @>835E,@>831A Enough space now?
 4F96 : BS GROM@>567D No, error
 4F98 : RTN

Basic (:

4F99 : CEQ @>8342,>B6 Token)?
 4F9C : BS GROM@>5671 Error
 4F9E : PARS >B7 Go on till (
 4FA0 : CEQ @>8342,>B6 Token)
 4FA3 : BR GROM@>5671 No, error
 4FA5 : XML >1B Fetch Basic byte
 4FA7 : CONT

Basic -:

4FA8 : PARS >C2 Go on till -
 4FAA : DNEG @>834A Change number into negative number
 4FAC : CH @>834C,>63 Numeric?
 4FAF : BS GROM@>5671 No, error
 4FB1 : CONT

Basic +:

4FB2 : PARS >C1 Go on till +
 4FB4 : BR GROM@>4FAC

Basic FOR:

4FB6 : CGT @>8342,>00
 4FB9 : BR GROM@>5671 Incorrect statement
 4FBB : XML >13 Fetch variable name
 4FBD : CZ VDP*>834A Not present ?
 4FC0 : BR GROM@>5671 Incorrect statement
 4FC2 : CEQ @>8342,>BE =?
 4FC5 : BR GROM@>5671 Incorrect statement
 4FC7 : XML >14 Build stack entry
 4FC9 : XML >1B Fetch byte
 4FCB : ST @>834C,>67 FOR tag
 4FCE : DST @>8350,@>832E Actual line number
 4FD1 : DST @>835C,@>836E Stack pointer
 4FD4 : DCGT @>835C,@>8324 Compare with start value stack
 4FD7 : BR GROM@>5007
 4FD9 : CEQ VDP@>0002(@>835C),>67 Another loop ?
 4FDE : BR GROM@>4FEC No, jump
 4FE0 : DCEQ VDP*>835C,@>834A Same loop ?
 4FE4 : BS GROM@>4FF2
 4FE6 : DSUB @>835C,>0018 3*8
 4FEA : BR GROM@>4FD4
 4FEC : DSUB @>835C,>0008 -8
 4FF0 : BR GROM@>4FD4
 4FF2 : DST @>835E,@>836E Value stack pointer
 4FF5 : DSUB @>835E,@>835C Minus old value stack pointer

```

4FF8 : BS      GROM@>5003
4FFA : MOVE    @>835E TO VDP@>FFF0(@>835C) FROM VDP@>0008(@>835C) 24 bytes
5003 : DSUB    @>836E,>0018      -24
5007 : XML     >17              VPUSHG
5009 : XML     >17
500B : XML     >17
500D : PARS   >B1              Go on till TO
500F : XML     >17              VPUSHG
5011 : CEQ    @>8342,>B1        TO?
5014 : BR     GROM@>5671        Incorrect statement
5016 : XML     >1B              Fetch byte
5018 : PARS   >B2              Go on till STEP
501A : CH     @>834C,>63        Numeric ?
501D : BS     GROM@>4D81        String number mismatch
501F : DSUB    @>836E,>0020      Minus 4*8
5023 : XML     >17              VPUSHG
5025 : CZ     @>8342            End of line ?
5027 : BS     GROM@>5045
5029 : CEQ    @>8342,>B2        STEP?
502C : BR     GROM@>5671
502E : DADD   @>836E,>0018
5032 : XML     >1B              Fetch byte
5034 : PARS   >00              Go on till end of line
5036 : DSUB    @>836E,>0018
503A : DCZ    @>834A            0?
503C : BS     GROM@>4D7C        Bad value
503E : CH     @>834C,>63        String tag?
5041 : BS     GROM@>4D81
5043 : BR     GROM@>504B
5045 : MOVE    >0008 TO @>834A FROM GROM@>50C0 Error value 1 on stack
504B : XML     >17              VPUSHG
504D : DADD   @>836E,>0010
5051 : XML     >18              VPOP
5053 : XML     >15              Appoint value to variable
5055 : DADD   @>836E,>0008
5059 : DST    @>8300,VDP@>0004(@>836E) Fetch pointer to value of variable
505E : MOVE    >0008 TO @>835C FROM VDP*>8300 Value on ARG
5064 : MOVE    >0008 TO @>834A FROM VDP@>FFF0(@>836E) Limit on FAC
506C : XML     >0A              FCOMP
506E : BS     GROM@>507B        Equal, end
5070 : CLOG   VDP@>FFF8(@>836E),>80 Negative?
5076 : BR     GROM@>507C
5078 : GT     Test if greater
5079 : BS     GROM@>507F
507B : CONT
507C : GT     Test if greater
507D : BS     GROM@>507B
507F : ST     @>8300,>01
5082 : DST    @>8302,@>832E      Pointer to line table
5085 : DST    @>8304,@>8330      Start line table
5088 : DINCT  @>8304            +2
508A : DCEQ   @>832E,@>8304      Line equal ?
508D : BR     GROM@>5094        No, jump
508F : DST    @>832E,@>8302
5092 : BR     GROM@>566C        Can't do that error
5094 : CALL   GROM@>2836        Find line address and 1st token
5097 : CEQ    @>8342,>8C        FOR?
509A : BR     GROM@>50A0        No, go on
509C : INC    @>8300            Counter FOR-NEXT loops
509E : BR     GROM@>50A9
50A0 : CEQ    @>8342,>96        NEXT?
50A3 : BR     GROM@>50A9
50A5 : DEC    @>8300            -1
50A7 : BS     GROM@>50AB        If zero, end
50A9 : BR     GROM@>508A        Next line
50AB : XML     >1B              Fetch byte

```

```

50AD : CZ      @>8342          0?
50AF : BS      GROM@>5671      Incorrect statement
50B1 : XML     >13             Fetch name
50B3 : DCEQ   @>834A,VDP*>836E Equal stack entry?
50B7 : BR      GROM@>566C      Incorrect statement
50B9 : DSUB   @>836E,>0018
50BD : CLR    @>8342          Set end of line
50BF : CONT   Go on in Basic

50C0 : DATA  >40,>01,>00,>00,>00,>00,>00,>00

Basic randomize:
50C8 : CZ      @>8342          End of line ?
50CA : BR      GROM@>50D1      No, jump
50CC : ST      @>83C1,@>8379   Set RAND on random number seed
50D0 : CONT
50D1 : PARS   >00             Fetch quotation
50D3 : CALL   GROM@>4F79      String tag?
50D6 : DST    @>83C0,@>834A   Quotation on random number seed
50DA : CONT

Basic CALL:
50DB : CEQ    @>8342,>C8       Unquoted string?
50DE : BR      GROM@>5671      No, error
50E0 : CLR    @>830C
50E2 : DST    @>8356,@>832C   Pointer Basic rollout area on DSR pointer
50E5 : CZ     @>8389          Program in GROM?
50E8 : BS     GROM@>5101      No, jump
50EA : MOVE   >0001 TO @>830D FROM GROM@>0000(@>832C) Fetch length byte
50F1 : INC    @>830D
50F3 : MOVE   @>830C TO VDP@>0320 FROM GROM@>0000(@>832C) Fetch name
50FA : DST    @>8356,>0320    Set DSR pointer
50FE : DADD   @>832C,@>830C   Text pointer plus length
5101 : CALL   GROM@>0010      Call subprogram
5104 : DATA  >0A
5105 : BS     GROM@>510C      Not found, then error
5107 : CZ     @>8342          End of line?
5109 : BR     GROM@>5671      No, error
510B : CONT
510C : CALL   GROM@>284E      Error
510F : DATA  >20            Bad name
5110 : DATA  >40

Basic string in "":
5111 : DST    @>8366,@>832C   Text pointer on >8366
5114 : CALL   GROM@>511D      Fetch text
5117 : DADD   @>832C,@>8350   New text pointer
511A : XML    >1B            Fetch byte
511C : CONT

Fetch quoted string :
511D : ST     @>8351,@>8342   Length on >8351
5120 : CLR    @>8350
5122 : DST    @>830C,@>8350   Length(word) on >830C
5125 : CALL   GROM@>515C      Fetch space for string
5128 : ST     @>8352,@>8389   GROM flag
512C : DST    @>834A,>001C    Tag for quotation
5130 : DST    @>834E,@>831C   Address
5133 : DST    @>834C,>6500    String tag
5137 : DCZ    @>830C          0?
5139 : BS     GROM@>514E      End
513B : CZ     @>8352          GROM?
513D : BR     GROM@>5147      Yes, jump
513F : MOVE   @>830C TO VDP*>831C FROM VDP*>8366 Fetch string from VDP
5145 : BR     GROM@>514E
5147 : MOVE   @>830C TO VDP*>831C FROM GROM@>0000(@>8366) Fetch string from GROM

```

```

514E : RTN

514F : CALL GROM@>515C      Fetch space for string
5152 : CLR   @>8352
5154 : BR    GROM@>512C      Build string stack entry

5156 : CALL GROM@>515C      Fetch space for string
5159 : CLR   @>8323          Clear error code
515B : RTNB                               Go on in Basic

Fetch space for string:
515C : DADD @>830C,>0004      Length+4 (address and twice length byte)
5160 : DST  @>8356,@>831A      Start string space
5163 : DSUB @>8356,@>830C      Minus length
5166 : DST  @>8358,@>836E      Value stack pointer
5169 : DADD @>8358,>0040      + 8*8
516D : DCH  @>8356,@>8358      Enough space?
5170 : BS   GROM@>5187          Yes, jump
5172 : CALL GROM@>51A9          Garbage collection
5175 : DST  @>8358,@>836E      Compute space once more
5178 : DADD @>8358,>0040
517C : DST  @>8356,@>831A
517F : DSUB @>8356,@>830C
5182 : DCH  @>8356,@>8358
5185 : BR    GROM@>567D          No space, memory full
5187 : DSUB @>830C,>0004      Length -4
518B : ST   VDP*>831A,@>830D Write length byte in VDP
518F : DSUB @>831A,@>830C      Start string space minus length
5192 : DST  @>831C,@>831A      Temporary pointer on string space
5195 : DSUB @>831A,>0004      -4
5199 : DCLR VDP@>0001(@>831A) Free
519D : ST   VDP@>0003(@>831A),@>830D 2nd length byte in VDP
51A2 : RTN

51A3 : CLR   @>8323          Clear error code
51A5 : CALL GROM@>51A9          Garbage collection
51A8 : RTNB

Garbage collection:
51A9 : DST  @>8354,@>8340      Pointer free space under symbol table
51AC : DST  @>8352,@>8318      End string space (high address)
51AF : DST  @>8318,@>8340      New pointer end string space
51B2 : DINC @>8352
51B4 : DDEC @>8352
51B6 : CLR  @>8356
51B8 : ST   @>8357,VDP*>8352 Length first string
51BC : DCHE @>831A,@>8352      Unprotected string?
51BF : BR    GROM@>51C5          No, shift if necessary
51C1 : DST  @>831A,@>8354      New pointer start string space, low address
51C4 : RTN

51C5 : DSUB @>8352,@>8356      String space minus length of 1st string
51C8 : DSUB @>8352,>0003      Minus 3, points to address
51CC : DCZ  VDP*>8352          0?
51CF : BS   GROM@>51B4          String cleared, jump
51D1 : CALL GROM@>5423          Shift string
51D4 : DADD @>8354,>0004      +4
51D8 : DST  @>8358,VDP@>FFFD(@>8354) CDP address symbol table
51DE : DST  VDP*>8358,@>8354 New address
51E2 : DSUB @>8354,>0004      -4
51E6 : BR    GROM@>51B4          Go on

51E8 : CEQ  @>834C,>65          String tag?
51EB : BR    GROM@>4D81          Error string number mismatch
51ED : XML  >17                 VPUSHG
51EF : XML  >1B                 Fetch byte

```

```

51F1 : RTN

51F2 : CLR      @>8323          Clear vector
51F4 : CALL    GROM@>51E8      Check string tag
51F7 : PARS    >B8            Fetch 2nd string
51F9 : CEQ     @>834C,>65      String?
51FC : BR      GROM@>4D81      No, error
51FE : DST     @>830C,@>8350    Length
5201 : DADD    @>830C,VDP@>0006(@>836E) Plus length of 1st string from value stac
k5206 : DCH     @>830C,>00FF    Smaller than 255?
520A : BR      GROM@>5210      Yes, jump
520C : DST     @>830C,>00FF    That's all
5210 : DST     @>8368,@>830C    Save on >8368
5213 : XML     >17            V PUSHG
5215 : CALL    GROM@>515C      Fetch space for new string
5218 : XML     >18            V POP
521A : MOVE    >0008 TO @>835C FROM @>834A Stack entry on ARG
521F : XML     >18            V POP
5221 : DST     @>8366,@>834E    Address 1st string
5224 : DST     @>830C,@>8350    Length
5227 : CLR     @>8352
5229 : CALL    GROM@>512C      String into new space
522C : DCZ     @>8362          2nd string with length 0?
522E : BS      GROM@>5246      Yes, end
5230 : DST     @>8364,@>831C    Address
5233 : DADD    @>8364,@>8350    Plus length of 1st strings
5236 : DSUB    @>8368,@>8350    Total length minus length of first string
5239 : DCZ     @>8368          Number 0?
523B : BS      GROM@>5246      Yes, jump
523D : MOVE    @>8368 TO VDP*>8364 FROM VDP*>8360 Add new string
5243 : DADD    @>8350,@>8368    New length
5246 : DST     @>8366,@>8360    Address of 2nd string
5249 : CONT

```

Basic SEG\$:

```

524A : CALL    GROM@>57A6      Token (?)
524D : XML     >1B            Fetch byte
524F : PARS    >B3            Go on till ,
5251 : CEQ     @>8342,>B3      Token ,?
5254 : BR      GROM@>5671      No, error
5256 : CALL    GROM@>51E8      Controll string and V PUSHG
5259 : PARS    >B3            Till ,
525B : CEQ     @>8342,>B3      Token ,?
525E : BR      GROM@>5671      No, error
5260 : CALL    GROM@>5740      Convert into integer
5263 : DCZ     @>834A          0?
5265 : BS      GROM@>4D7C      Error
5267 : XML     >17            V PUSHG
5269 : XML     >1B            Fetch byte
526B : PARS    >B6            Go on till )
526D : CEQ     @>8342,>B6      End ?
5270 : BR      GROM@>5671      No, end
5272 : CALL    GROM@>5740      Convert into integer
5275 : DST     @>835C,@>834A    Save on >835C
5278 : XML     >18            V POP, Fetch position
527A : DST     @>835E,@>834A    On >835E
527D : XML     >18            V POP
527F : DST     @>8356,@>835E
5282 : DCH     @>8356,@>8350    Longer?
5285 : BS      GROM@>52BA
5287 : DADD    @>8356,@>835C    Plus length
528A : DSUB    @>8356,@>8350
528D : DDEC    @>8356
528F : DCGE    @>8356,>0000    Still greater than 0?
5293 : BR      GROM@>529D
5295 : DST     @>835C,@>8350    New length

```

5298 : DSUB @>835C,@>835E Minus position
529B : DINC @>835C
529D : DST @>830C,@>835C
52A0 : XML >17 VPUSHG
52A2 : CALL GROM@>515C Fetch space for string
52A5 : XML >18 VPOP
52A7 : DST @>8366,@>834E Address
52AA : DADD @>8366,@>835E Plus length
52AD : DDEC @>8366 Minus 1
52AF : DST @>8350,@>830C Length
52B2 : CLR @>8352
52B4 : CALL GROM@>512C Fetch string
52B7 : XML >1B Appoint value
52B9 : CONT
52BA : DCLR @>835C Length 0
52BC : BR GROM@>529D Go on

Basic LEN:

52BE : CALL GROM@>57A6 Check token (
52C1 : PARS >FF Go on
52C3 : CEQ @>834C,>65 String tag?
52C6 : BR GROM@>4D81 No, error
52C8 : ST @>835C,@>8351 Length
52CB : CLR @>835D
52CD : MOVE >0008 TO @>834A FROM GROM@>50C0 1 on FAC
52D3 : CH @>835C,>63 Greater than 100?
52D6 : BR GROM@>52E0 No, end
52D8 : EX @>835D,@>835C In lbyte
52DB : DIV @>835C,>64 /100
52DE : INC @>834A Plus 1
52E0 : DST @>834B,@>835C Back
52E3 : CZ @>834B 0?
52E5 : BR GROM@>52E9
52E7 : CLR @>834A Well then 0
52E9 : CONT

Basic CHR\$:

52EA : CALL GROM@>57A6 Token (?
52ED : PARS >FF Go on
52EF : CALL GROM@>5740 Convert into integer
52F2 : DST @>830C,>0001 Length 1
52F6 : ST VDP@>03DD,@>834B Save in VDP
52FA : DST @>8366,>03DD
52FE : CALL GROM@>514F Space for string, and transfer string
5301 : DST @>8350,>0001 Length 1
5305 : CONT

Basic ASC:

5306 : CALL GROM@>57A6 Check token (
5309 : PARS >FF Go on
530B : CEQ @>834C,>65 String tag?
530E : BR GROM@>4D81 Error
5310 : CZ @>8351 Length 0?
5312 : BS GROM@>4F74 Error
5314 : ST @>835C,VDP*>834E Fetch 1st byte from string
5318 : BR GROM@>52CB Go on by converting into floating point

Basic STR\$:

531A : CALL GROM@>57A6 Check token (?
531D : PARS >FF Go on
531F : CHE @>834C,>64 Numeric ?
5322 : BS GROM@>4D81 No, error
5324 : CLR @>8355
5326 : CALL GROM@>0014 Convert number to string
5329 : CEQ *>8355,>20 Is 1st byte of string space?
532D : BR GROM@>5333

532F : INC @>8355 Address +1
5331 : DEC @>8356 Length -1
5333 : CLR @>830C Length
5335 : ST @>830D,@>8356 Complete length
5338 : MOVE @>830C TO VDP@>03C0 FROM *>8355 String in VDP
533E : DST @>8366,>03C0
5342 : CALL GROM@>514F Fetch space for string, build string entry
5345 : DST @>8350,@>830C Length on stack entry
5348 : CONT

Basic VAL:

5349 : CALL GROM@>57A6 Check token (?
534C : PARS >FF Go on
534E : CEQ @>834C,>65 String tag?
5351 : BR GROM@>4D81 No, error
5353 : CZ @>8351 Length 0?
5355 : BS GROM@>4F74 Bad argument
5357 : DST @>8366,@>834E Pointer to string
535A : DADD @>8366,@>8350 Plus length
535D : DDEC @>8366 Minus 1
535F : DST @>830C,@>8350 Save length
5362 : CEQ VDP*>8366,>20 Space?
5366 : BR GROM@>5371 No, execute
5368 : DDEC @>830C Length -1
536A : BS GROM@>4F74 Now 0? Then error
536C : DDEC @>8366 Address -1
536E : B GROM@>5362 New start

5371 : DINC @>830C Length -1
5373 : XML >17 VPUSHG
5375 : CALL GROM@>515C Fetch space for string
5378 : XML >18 VPOP
537A : DST @>8366,@>834E Pointer to string
537D : CLR @>8352
537F : CALL GROM@>5137 Fetch string
5382 : DADD @>830C,@>831C Address to length
5385 : DDEC @>830C -1
5387 : ST VDP*>830C,>20 Space
538B : DST @>8356,@>831C Pointer to string
538E : CEQ VDP*>8356,>20 Space?
5392 : BR GROM@>5399 No, convert
5394 : DINC @>8356 Pointer +1
5396 : B GROM@>538E Once again
5399 : CLR @>834C
539B : CLR @>8354
539D : XML >10 Convert string into number
539F : DCEQ @>8356,@>830C
53A2 : BR GROM@>4F74 Error bad argument
53A4 : CZ @>8354 Error?
53A6 : BR GROM@>56B5 Yes, number too big
53A8 : CONT

Basic POS:

53A9 : CALL GROM@>57A6 Check token (
53AC : XML >1B Fetch byte
53AE : PARS >B3 Go on till ,
53B0 : CEQ @>8342,>B3 Token ,?
53B3 : BR GROM@>5671 No, error
53B5 : CALL GROM@>51E8 String? And on stack
53B8 : PARS >B3 Go on till ,
53BA : CEQ @>8342,>B3 Token ,?
53BD : BR GROM@>5671 No, error
53BF : CALL GROM@>51E8 String? And on stack
53C2 : PARS >B6 Go on till)
53C4 : CEQ @>8342,>B6 Token)?
53C7 : BR GROM@>5671 No, error

```

53C9 : CALL   GROM@>5740      Numeric, convert into integer
53CC : DCZ     @>834A          Not too big?
53CE : BS     GROM@>4D7C      Yes, error
53D0 : DST    @>830C,@>834A   Save position
53D3 : DDEC   @>830C          -1
53D5 : XML    >18            VPOP
53D7 : MOVE   >0008 TO @>835C FROM @>834A   Stack entry on ARG
53DC : XML    >18            VPOP
53DE : CZ     @>8351          0-String?
53E0 : BS     GROM@>541F      Yes, jump
53E2 : CH     @>8351,@>830D   Position behind length of string?
53E5 : BR     GROM@>541F      Yes, jump
53E7 : CZ     @>8363          Search string with length 0?
53E9 : BS     GROM@>540E      Yes, jump
53EB : DADD   @>834E,@>830C   Compute address from search point
53EE : SUB    @>8351,@>830D
53F1 : CHE    @>8351,@>8363   Does it still fit?
53F4 : BR     GROM@>541F      No, jump
53F6 : DST    @>834A,@>834E   Compute addresses in VDP for searching
53F9 : DST    @>835C,@>8360
53FC : ST     @>8364,@>8363   Length
53FF : CEQ    VDP*>835C,VDP*>834A   Equal ?
5404 : BR     GROM@>5417      No, jump
5406 : DINC   @>834A          Increase addresses
5408 : DINC   @>835C
540A : DEC    @>8364          Decrease length
540C : BR     GROM@>53FF      And go on
540E : INC    @>830D          Found, correct position
5410 : ST     @>835C,@>830D
5413 : XML    >1B            Fetch byte
5415 : BR     GROM@>52CB      Convert data on >835C in floating point, end
5417 : INC    @>830D          +1
5419 : DEC    @>8351          -1
541B : DINC   @>834E          +1
541D : BR     GROM@>53F1      Once again
541F : CLR    @>830D          0
5421 : BR     GROM@>5410      End with 0

```

Shift string :

```

5423 : DADD   @>8356,>0004     Length +4
5427 : DST    @>8358,@>8354   Pointer to free space
542A : DSUB   @>8358,@>8352   Minus end string space
542D : DSUB   @>8358,@>8356   Minus length
5430 : DINC   @>8358          0?
5432 : BS     GROM@>5455      Yes, jump
5434 : DCHE   @>8358,@>8356   Greater than length?
5437 : BR     GROM@>543C
5439 : DST    @>8358,@>8356   Length
543C : DST    @>835A,@>8352   End string space (low address)
543F : DADD   @>835A,@>8356   Plus length
5442 : DSUB   @>835A,@>8358   Minus old pointer to free space
5445 : DSUB   @>8354,@>8358   Minus length
5448 : MOVE   @>8358 TO VDP@>0001(@>8354) FROM VDP*>835A Shift string
544F : DSUB   @>8356,@>8358   0?
5452 : BR     GROM@>5427      No, again
5454 : RTN

```

```

5455 : DSUB   @>8354,@>8356   New pointer free space
5458 : RTN

```

Basic TRACE:

```

5459 : OR     @>8388,>10      Set trace flag
545D : CONT

```

Basic UNTRACE:

```

545E : AND    @>8388,>EF      Clear trace flag

```

5462 : CONT

Basic BREAK:

5463 : ST @>8300,>FF Break flag
5466 : CZ @>8342 End of line?
5468 : BR GROM@>547F No, go on
546A : DCZ @>8344 Run flag?
546C : BS GROM@>566C No, can't do that
546E : DST VDP@>03EC,@>832E Save actual line pointer into VDP
5472 : DSUB VDP@>03EC,>0004 -4 for next line
5477 : BR GROM@>4E3C End of program

Basic UNBREAK:

5479 : CLR @>8300 Unbreak flag
547B : CZ @>8342 End of line?
547D : BS GROM@>54BC
547F : CALL GROM@>56BB Fetch line number
5482 : DST @>8312,@>8332 Pointer to line table
5485 : DSUB @>8312,>0003 1st line number
5489 : DCHE @>8312,@>8330 Still greater than lower end of line list ?
548C : BR GROM@>54B5 No, jump
548E : DCEQ VDP*>8312,@>834A Righth line ?
5492 : BS GROM@>549A Yes, jump
5494 : DSUB @>8312,>0004 No, next line
5498 : BR GROM@>5489 And once more
549A : DINCT @>8312 On pointer to line
549C : AND VDP*>8312,>7F 0 > bit reset
54A0 : CZ @>8300 Break flag?
54A2 : BS GROM@>54A8 No, jump
54A4 : OR VDP*>8312,>80 0 > Set bit as break flag
54A8 : CZ @>8342 End of line
54AA : BS GROM@>54CE
54AC : CEQ @>8342,>B3 Token ,?
54AF : BR GROM@>5671 No, error
54B1 : XML >1B Fetch byte
54B3 : BR GROM@>547F Once again
54B5 : CALL GROM@>284C Error
54B8 : DATA >20 Bad line number
54B9 : DATA >D9
54BA : BR GROM@>54A8
54BC : DST @>8312,@>8330 Start line table
54BF : DINCT @>8312
54C1 : AND VDP*>8312,>7F Reset bit 0
54C5 : DADD @>8312,>0004 Next line
54C9 : DCH @>8312,@>8332 Till upper end of line list
54CC : BR GROM@>54C1 Loop
54CE : CONT

54CF : CLR @>8351 No argument
54D1 : DCLR @>8322
54D3 : CLR @>835E
54D5 : CLR @>834C
54D7 : CEQ @>8342,>B7 Yet argument?
54DA : BR GROM@>54ED No, jump
54DC : XML >17 VPUSHG
54DE : XML >1B Fetch byte
54E0 : PARS >B6 Fetch argument
54E2 : XML >1B Fetch byte
54E4 : MOVE >0008 TO @>835C FROM @>834A Stack entry argument on ARG
54E9 : XML >18 VPOP
54EB : INC @>8351 Count argument
54ED : ST @>8366,@>8351 Save
54F0 : DST @>8364,@>834A Save pointer on entry symbol table
54F3 : XML >17 VPUSHG
54F5 : MOVE >0008 TO @>834A FROM @>835C Again on FAC
54FA : XML >17 VPUSHG

```

54FC : ST      @>834C,VDP*>8364  Fetch entry symbol table
5500 : ST      @>834D,@>834C
5503 : AND     @>834C,>07
5506 : CEQ    @>834C,@>8366   Equal type?
5509 : BR     GROM@>5671     No, error incorrect statement
550B : DST    @>834A,@>832C   Program text pointer
550E : CLR    @>834C
5510 : AND    @>834D,>80     String or not?
5513 : DSUB  @>836E,>0010   Value stack pointer
5517 : DST    @>834E,@>833E   Old pointer to symbol table
551A : DST    @>8350,@>8340   Old pointer free space symbol table
551D : XML    >17          VPUSHG (Stack entry DEF)
551F : DADD  @>836E,>0008   Stack pointer
5523 : DST    @>832C,VDP@>0006(@>8364)  Pointer to definition is new text
                    pointer
5528 : XML    >1B          Fetch byte
552A : CH    @>8373,>AC     Substack to high?
552D : BS    GROM@>567D     Error memory full
552F : MOVE  >0018 TO VDP@>03C0 FROM @>8300  Save several pointers
5535 : OR    @>8388,>08     Set flag
5539 : ST    @>8316,>80     Length
553C : CEQ  @>8342,>BE     Token =?
553F : BR    GROM@>554E     No, jump
5541 : CLR  @>8359
5543 : CALL  GROM@>284A     Dummy entry into symbol table
5546 : DDECT @>832C       Text pointer minus 2
5548 : CLR  VDP@>0002(@>836E)  Clear on stack
554C : BR    GROM@>5551
554E : CALL  GROM@>2848     Entry symbol table
5551 : XML  >1B          Fetch byte
5553 : AND  @>8388,>F7     Clear flag again
5557 : MOVE >0018 TO @>8300 FROM VDP@>03C0  Old value again
555D : ST    VDP@>FFFA(@>836E),>68  DEF key value
5563 : DST  VDP@>0002(@>833E),VDP@>03E0  Pointer next entry symbol table
5569 : DST  @>834A,@>833E   Pointer symbol table
556C : XML  >14          Build stack entry
556E : MOVE >0008 TO @>8352 FROM @>834A     Save on >8352
5573 : XML  >18          VPOP
5575 : MOVE >0008 TO @>835C FROM @>834A     Save on >835C
557A : MOVE >0008 TO @>834A FROM @>8352     On FAC again
557F : XML  >17          VPUSHG
5581 : MOVE >0008 TO @>834A FROM @>835C     On FAC again
5586 : CEQ  @>834C,>65     String?
5589 : BR   GROM@>5595     No, jump
558B : DCEQ @>834A,>001C   Quotation?
558F : BS   GROM@>5595     No, jump
5591 : DST  @>834E,VDP*>834A  Address string pointer
5595 : XML  >1B          Fetch byte
5597 : XML  >15          Appoint value to variable
5599 : PARS >00          Execute DEF
559B : CEQ  @>834C,>65     String?
559E : BR   GROM@>55A8     No, jump
55A0 : CZ   VDP@>0003(@>836E)  Flag for string (>80)?
55A4 : BS   GROM@>4D81     Yes, error
55A6 : BR   GROM@>55AE     Jump
55A8 : CZ   VDP@>0003(@>836E)  Flag for numeric (>00)?
55AC : BR   GROM@>4D81     No, error
55AE : CALL GROM@>55BB     Protect string
55B1 : DST  @>832C,VDP@>0008(@>836E)
55B6 : DDEC @>832C       Text pointer
55B8 : XML  >1B          Fetch byte
55BA : CONT  Go on in Basic

55BB : DST    @>8366,@>833E   Save pointer to symbol table
55BE : MOVE  >0004 TO @>833E FROM VDP@>0004(@>836E)  Fetch 4 bytes from stack

```

```

55C5 : CGE      VDP*>8366,>00      String?
55C9 : BS      GROM@>55FB      No, jump
55CB : DST      @>8366,VDP@>0006(@>8366) Fetch pointer to value
55D0 : DCZ      @>8366      0-string?
55D2 : BS      GROM@>55E4      Yes, jump
55D4 : DST      @>8356,VDP@>FFFD(@>8366) New pointer before string
55DA : DCHE     @>8356,@>833E    String lies above symbol table
55DD : BS      GROM@>55E4      Yes, jump
55DF : DCLR     VDP@>FFFD(@>8366) Clear string
55E4 : CEQ      @>834C,>65      String tag?
55E7 : BR      GROM@>55FB      No, end
55E9 : DCHE     @>834A,@>833E
55EC : BS      GROM@>55FB
55EE : DCZ      @>834E
55F0 : BS      GROM@>55F7
55F2 : DCLR     VDP@>FFFD(@>834E) No value
55F7 : DST      @>834A,>001C    Quotation
55FB : DSUB     @>836E,>0008    Stack -8
55FF : RTN

5600 : PARS     >B6          Go on till )
5602 : CALL     GROM@>4F79    Check string tag
5605 : DST      @>8310,>001E    Limit 30
5609 : CALL     GROM@>3785    CFI
560C : SRL      @>834B,>01     Divide by 2
560F : OR       @>834B,>F0     Set the first 4 bits
5612 : RTN

5613 : DDEC     @>8336          On pointer to line
5615 : CZ       @>8389          In GROM?
5618 : BR      GROM@>5623      Yes, jump
561A : DST      @>8334,VDP*>8336 Fetch pointer to line
561E : AND      @>8334,>7F     Bit 0 reset
5621 : BR      GROM@>562A
5623 : MOVE     >0002 TO @>8334 FROM GROM@>0000(@>8336) Fetch pointer to line
562A : ST       @>834D,@>8389
562E : CALL     GROM@>4010      Fetch DATA
5631 : CEQ      @>8301,>93     Token DATA?
5634 : BS      GROM@>5644      Yes, end
5636 : DDECT    @>8336          Next line
5638 : DCEQ     @>8336,@>8330  End of line list?
563B : BS      GROM@>5641      Yes, end with flag
563D : DDEC     @>8336          -1
563F : BR      GROM@>5613      The same once again
5641 : ST       @>8334,>FF     No more data for READ
5644 : RTN

5645 : CZ       @>8389          In GROM?
5648 : BR      GROM@>5652
564A : DST      @>835E,VDP@>FFFE(@>832E) Fetch line number from VDP
5650 : BR      GROM@>5659
5652 : MOVE     >0002 TO @>835E FROM GROM@>FFFE(@>832E) Fetch line number from GRO
M5659 : B       GROM@>2842      Stop writing on screen

```

Jump table for error codes on >8322:

```

565C : CASE     @>8322
565E : BR      GROM@>5671
5660 : BR      GROM@>567D
5662 : BR      GROM@>4D7C
5664 : BR      GROM@>5682
5666 : BR      GROM@>566C
5668 : BR      GROM@>5678
566A : BR      GROM@>4D81

```

Call error :

```

566C : CALL     GROM@>284E

```

```

566F : DATA >20          Can't do that
5670 : DATA >BD
5671 : CALL GROM@>284E
5674 : DATA >20          Incorrect statement
5675 : DATA >2C
5676 : BR GROM@>4E58      Basic return
5678 : CALL GROM@>284E
567B : DATA >20          Bad subscript
567C : DATA >A1
567D : CALL GROM@>284E
5680 : DATA >20          Memory full
5681 : DATA >49
5682 : CALL GROM@>284E
5685 : DATA >20          Bad line number
5686 : DATA >D9
5687 : BR GROM@>4E58      Basic return
5689 : CH @>8373,>B0      Substack to high?
568C : BS GROM@>567D      Yes, memory full
568E : CALL GROM@>284C    Print warning
5691 : DATA >20          Number too big
5692 : DATA >6E
5693 : DCLR @>8322        Reset error code
5695 : RTNB

5696 : DCLR @>8322        Error code 0
5698 : CALL GROM@>4F7F    Check substack with garbage collection
569B : CH @>834C,>63      Numeric ?
569E : BS GROM@>4D81      String number mismatch
56A0 : CH VDP@>0002(@>836E),>63 Numeric ?
56A5 : BS GROM@>4D81      String number mismatch
56A7 : CLR @>8354
56A9 : CALL GROM@>0024    Involution routine
56AC : CZ @>8354          Error?
56AE : BS GROM@>56BA      No, end
56B0 : CEQ @>8354,>01     Same error?
56B3 : BR GROM@>4D7C      Bad value
56B5 : CALL GROM@>284C
56B8 : DATA >20          Number too big
56B9 : DATA >6E
56BA : CONT

Fetch line number on FAC:
56BB : CEQ @>8342,>C9     Token for line number
56BE : BR GROM@>5671      Incorrect statement
56C0 : XML >1B           Fetch 1st byte
56C2 : ST @>834A,@>8342
56C5 : XML >1B           Fetch 2nd byte
56C7 : ST @>834B,@>8342   Everything on FAC
56CA : XML >1B           Fetch byte
56CC : RTN

Scroll one line :
56CD : MOVE >02E0 TO VDP@>0000 FROM VDP@>0020 All one line up
56D4 : FMT               Print empty line
56D5 : ... XPT=>00       Column 0
56D7 : ... YPT=>17       Line 23 (last line, sinc counted from 0)
56D9 : ... 02'>7F'      2 x >7F
56DB : ... 1C'>80'      28 x >80 (Space with offset)
56DD : ... 02'>7F'      2 x >7F
56DF : ... END FMT
56E0 : RTN

Convert string into number:
56E1 : CLR @>8354        Error clear
56E3 : XML >10           CSN
56E5 : CZ @>8354        Error?

```

```

56E7 : BS      GROM@>56EE
56E9 : CALL    GROM@>284C
56EC : DATA   >20          Number too big
56ED : DATA   >6E
56EE : RTN

CALL GCHAR:
56EF : CALL    GROM@>378E      Fetch and set column and row on screen
56F2 : CALL    GROM@>578D      Variable name from program
56F5 : MOVE    >0008 TO @>834A FROM GROM@>50C0 1 on FAC
56FB : ST      @>834B,@>837D   Byte from screen
56FE : SUB     @>834B,>60       Minus offset
5701 : CHE     @>834B,>64       Greater than decimal 100?
5704 : BR      GROM@>570E      No, jump
5706 : EX      @>834C,@>834B   Convert
5709 : DIV     @>834B,>64
570C : INC     @>834A
570E : XML     >15             Transfer value to variable
5710 : B       GROM@>361D     End with reset XPT

CALL COLOR:
5713 : CALL    GROM@>3767      Fetch byte and first argument
5716 : CALL    GROM@>3779      CFI with limit 16
5719 : DADD    @>834A,>030F    Add offset color table
571D : XML     >17             VPU SHG
571F : CALL    GROM@>376F      Next value
5722 : CALL    GROM@>3779      CFI with limit
5725 : DEC     @>834B          -1
5727 : ST      @>830E,@>834B   Save
572A : SLL     @>830E,>04      1st nybble
572D : PARS    >B6            Next value
572F : CALL    GROM@>3779      CFI
5732 : DEC     @>834B          -
5734 : OR      @>830E,@>834B   Complete value for color table
5737 : XML     >18             VPOP (fetch gruppe)
5739 : ST      VDP*>834A,@>830E Set color table
573D : B       GROM@>3620     End of subprograms

Convert floating point into line number:
5740 : CH      @>834C,>63      Numeric ?
5743 : BS      GROM@>4D81      String number mismatch
5745 : CLR     @>8354
5747 : DCLR    @>836C
5749 : XML     >12             CFI
574B : CZ      @>8354          Error?
574D : BR      GROM@>4D7C      Bad value
574F : CLOG    @>834A,>80      Negative?
5752 : BR      GROM@>4D7C      Bad value
5754 : RTN

5755 : CLOG    @>8300,>80      Negative?
5758 : BR      GROM@>5767      Yes, jump
575A : CZ      @>8300          0?
575C : BS      GROM@>5763      Yes, jump
575E : ST      @>834B,@>8300   Value in number
5761 : BR      GROM@>576D
5763 : DCLR    @>834A          Number = 0
5765 : BR      GROM@>576D
5767 : ST      @>834A,>BF      Negative
576A : ST      @>834B,@>8300   Value into number
576D : XML     >15             Transfer number into variable
576F : RTN

Keyboard scanning for CALL KEY usw:
5770 : ST      @>8300,@>834B   Save mode
5773 : CALL    GROM@>578D      Fetch 1st variable

```

```

5776 : CEQ    @>8342,>B3      ,?
5779 : BR      GROM@>5671     No, error
577B : XML     >1B           Fetch byte
577D : CALL   GROM@>578D     Fetch 2nd variable
5780 : ST      @>8374,@>8300  Set mode
5783 : MOVE   >0008 TO @>834A FROM GROM@>50C0 1 on FAC
5789 : SCAN   Keyboard scanning
578A : CLR    @>8374       Mode again 0
578C : RTNC   Set condition bit = key is pressed

578D : XML     >13           Fetch name from program
578F : CLOG   VDP*>834A,>F8  Test if numeric
5793 : BR      GROM@>5671     Incorrect statement
5795 : XML     >14           Build stack entry
5797 : XML     >17           VPUSHG
5799 : RTN

579A : DATA  >8000,>A000,>C000,>9FBF,>DFFF,>0006

57A6 : CEQ    @>8342,>B7      Token (?)
57A8 : BR      GROM@>5671     No, error
57AC : RTN

57AC : DATA  >0000
bis
57BE : DATA  >0000

57C0 : CALL   GROM@>40E7     New free space for symbol table
57C3 : SUB    @>8373,>04
57C6 : DCEQ   *>8373,>41C4  Called by >41C4 (Close all files)?
57CB : BR      GROM@>57D3     No, go on
57CD : CALL   GROM@>284C     Warning
57D0 : DATA  >21           I/O error
57D1 : DATA  >13
57D2 : RTN

57D3 : ADD    @>8373,>04     Old substack pointer
57D6 : CALL   GROM@>284E     Error
57D9 : DATA  >21           I/O error
57DA : DATA  >13
57DB : B      GROM@>201A     Return Basic
57DE : CALL   GROM@>284E
57E1 : DATA  >21           File error
57E2 : DATA  >1D
57E3 : CALL   GROM@>284E
57E6 : DATA  >21           Input error
57E7 : DATA  >28
57E8 : CALL   GROM@>284E
57EB : DATA  >21           Data error
57EC : DATA  >34
57ED : RTN

57EE : DATA  >0000
to
57FC : DATA  >0000
57FE : DATA  >5F2E

```


REFERENCES TO EXTENDED BASIC

The function of the Extended Basic for use of TI99/4A is very similar to that of the Basic interpreter. For once the Basic program runs also under Extended Basic, and the Extended Basic uses several routines from ROM.

Of course Extended Basic cannot directly uses the Basic interpreter. The Extended Basic interpreter is newly programed in the ROM of the module. But its form and functions are only little different from the Basic interpreter. GPL commands EXEC, PARSE and CONT of course are not usable anymore. They are replaced by XML commands. XML >74 is PARSE, XML >75 is CONT and XML >76 replaces EXEC.

The structure of the list in the GROM for the GPL-subprogram activated by Basic command CALL, is also different; otherwise there would be problems with the compatibility to TI Basic when the Extended Basic module is inserted. Therefore the starting address of the routines in Extended Basic is put after the name of the Subprograms. In Extended Basic, while the program runs in VDP RAM, there is a subprogram list (Pointer on >833A) which is completed during prescan. The list has the same structure as the symbol list; there is a symbol list for each self-defined subprogram containing variables.

Since Extended Basic can also use memory extension, two additional pointers are necessary:

>8384 points to the highest usable address in the memory extension.

>8386 points to the end of the free space in the memory extension. The GPL-substack is therefore placed higher.

There is only one important exception in connection with the other pointers in Scratch-Pad-RAM as compared to TI-Basic: The Flag-Byte is located at >8345.

Due to a restrained information politic of the manufacturer, there is not much known about the inside of the very successful Home Computer TI99/4A. This book therefore shall help the interested user to look into the secret of the operating system of the TI99/4A.

In individual chapters the commented listings of the ROM'S and the 3 GROM'S contained in the TI99/4A are explained i.e. the complete operating system including Basic. The contents of this book is completed by a listing of all commands, used in the Graphic Programing Language and by short explanations to the commented listings.

