MECHATRONIC COMPUTER PRODUCTS

REFERENCE MANUAL FOR MECHATRONIC EXTENDED BASIC II PLUS

SUPPLEMENTARY VOLUME TO THE TI EXTENDED BASIC REFERENCE MANUAL

FOR TEXAS INSTRUMENTS TI-99/4A HOME COMPUTERS

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

INTRODUCTION

Mechatronic Extended BASIC II^{PLUS} exceeds the possibilities of TI Extended BASIC considerably with enhanced capability and flexibility. About 60 powerful commands and statements have been added.

Méchatronic Extended BASIC II^{PLUS} is consisting of three major elements:

1. Standard Extended BASIC

The Standard Extended BASIC is a copy of the original TI Extended BASIC (version 110) and is manufactured under license of Texas Instruments. An extensive description of this powerful programming language can be found in the reference manual for TI Extended BASIC.

2. Extended Statement Set

More than 20 new statements provide additional features, which are on the one hand - even with high programming effort - not achievable with the Standard Extended BASIC, on the other hand, efficient simplifications in program development and significant increasing of the processing speed.

Examples are the hardcopy routine, direct access to the VDP RAM, moving of memory blocks, saving and loading of blocks to and from cassette recorders and other external devices.
The extended statement set of Mechatronic Extended BASIC II^{PLUS} will be dicussed in part 2 of this manual. Interesting hints about the use of the VDP RAM are in the addendum (part 4) of this manual.

3. High Resolution Graphic

High resolution graphic allows to adress every single pixel of the screen using all 16 forground and background colors. 40 powerful graphic statements are available to draw lines, circles, rectangles, arcs, circle diagrams etc., to save and load the graphics on diskettes, or to make hard copies. The graphic statements are written in TMS 9900 assembly language and do only work with a connected 32K byte RAM expansion. The statement set of the high resolution graphic has been implemented under license of Apesoft, Micro Computer Software, Austria. These statements are discussed in part 3 of this manual.

Extended BASIC II^{PLUS} is fully compatible with TI Extended BASIC or Mecharonic (Standard) Extended BASIC. This means, every program written with Extended BASIC (version 110), will work with Mechatronic Extended BASIC II^{PLUS} too. Owners of Mechatronic (Standard) Extended BASIC can get their modules upgraded by their retailers. This is not possible with original TI Extended BASIC modules.

PRINCIPLE OF EXTENDED BASIC II^PLUS

PRINCIPLE OF MECHATRONIC EXTENDED BASIC II^PLUS

Extended BASIC II^PLUS has two screen output operating modes available, which are called STANDARD MODE and GRAPHIC MODE. Both operating modes may be changed during program execution.

The standard mode is available after plugging in the module and choosing MECHATRONIC EXTENDED BASIC. All statements, commands, and functions of (standard) Extended BASIC are available plus the extended statement set. The 32K byte RAM expansion is not required, but it can be used. In standard mode the statements of the high resolution graphic are not availble.

The graphic mode will be gotten by following actions:

1.) Initializing of RAM expansion and transferring of graphic statement set with the command

CALL APESOFT

before loading or typing in any program.

2.) Switching to the graphic mode with the statement

CALL LINK("GRAFIC",MODUS)

In graphic mode are all those statements of the standard mode prohibited affecting any screen output. The reason is the different kind of manipulation of screen outputs by the TI-99/4A in standard mode and in graphic mode. Included are all sprite statements and the sound statement. A detailed assingnment list of all statements to both operating modes can be found in the addendum (part 4).

The graphic mode can be quit on two ways

1.) Reset into standard mode with the statement:

CALL LINK("BYEBYE")

This is the only defined methode returning back into standard mode!

2.) Program interruption (in graphic mode) caused by

BREAK, STOP, END, FCTN 4 or program error

Breaks cause an undefined condition, but they are often unavoidable. Continuing the program uncontrolled will result unpleasant side effects (which may be explored by the user himself). Depending, if the program shall be continued in graphic mode or in standard mode, the first executed statement after the break must be a CALL LINK("GRAFIC",MODUS) or a CALL LINK("BYEBYE")

STANDARD MODE – GRAPHIC MODE

Correct application of the graphic mode may be compared with the call for a subroutine'

CALL APESOFT in direct mode before loading of any program
immediately after selection of Extended BASIC

```
 *************************             ****************************
 *                       *             *                          *
 *     STANDARD MODE     *             *       GRAPHIC MODE       *
 *                       *             *                          *
 *************************             ****************************


  xxxx REM  start of program
  xxxx REM  in standard mode
                 '
                 '
                 '
  xxxx REM  switching to
  xxxx REM  graphic mode
  xxxx REM
  xxxx CALL LINK("GRAFIC",0)
                 '
                 --------------------------------
                                                '
                                       xxxx REM  program segment in
                                       xxxx REM  graphic mode
                                                '
                                                '
                                                '
                                       xxxx REM  quit
                                       xxxx REM  graphic mode
                                       xxxx REM
                                       xxxx CALL LINK("BYEBYE")
                                                '
                 --------------------------------
                 '
  xxxx REM segment in
  xxxx REM standard mode
                 '
                 '
                 '
  xxxx REM end of program
```

In graphic mode are maximum 2 opended filed at once permitted. Therfore CALL FILES should not be applied.

CALL CLRAPE quits the graphic mode and causes a reset to the initial condition, just after selection of MECHATRONIC EXTENDED BASIC. However, all stored programs and data will be erased.

EXTENDED STATEMENT SET

EXTENDED STATEMENT SET

Mechatronic Extended BASIC II PLUS extends the TI Extended BASIC. More than 20 new statements provide additional features, which are on the one hand - even with high programming effort - not achievable with the Standard Extended BASIC, on the other hand, efficient simplifications in program development and significant increasing of the processing speed.

The screen copy feature with a line printer is implemented as well as the choice the save or load screen displays or other segments of memory from and to cassette or diskette.

All extended statements are available in standard mode. They can be also applied in graphic mode if they do not affect screen outputs. A detailed assignment list of all statements to both operating modes can be found in the addendum (part 4).

Following system configuration is required if the standard mode shall be applied:

TI-99/4A + Extended BASIC II PLUS

Useful, but not essentially:

+ 32k byte RAM expansion
+ Disk system
+ Printer (with graphic capabilities)

CAUTION

The power supply of the 32k byte RAM expansion may be n o t provided by the TI-99/4A - Overheating!

BHCOPY

FORMAT

CALL BHCOPY("FILE NAME";"ESC-SEQUENCE")

DESCRIPTION

This subroutine generates a hardcopy of the screen on line printers operating in the BIT IMAGE MODE. Sprites will not be considered.

All valid file names of connected peripheral devices may be used, e.g. "RS232.BA=9600.DA=8.CR" or "PIO.CR". The extension .CR is important for correct working of the subroutine. Serial interfaces (RS232) need to use 8 data bits and the appropriate extension ".DA=8". Of course the printer has to be set internal to 8 data bits, too.

Concerning the ESC sequence the reference manual of the printer should be taken into consideration. BHCOPY is sending at the beginning of every line ESC (=CHR$(27)), followed by the inserted string sequence (up to 10 characters), than the characters CHR$(0), CHR$(1) (for 256 following bytes), than 256 bytes, correspondeing to the hardcopy of one line, and finally "CR" (=CHR$(13)). Many line printers will have good results if the ESC sequence "L" or "K" is applied.

EXAMPLES

```
CALL BHCOPY("RS232.DA=8.CR","L")
CALL BHCOPY("PIO.CR","K")
```

Program examples can be found in the description of BHCOPY and in the section of HARDCOPY DEMONSTRATION in Part 3.

NOTE,

LINEFEED MUST BE SET BEFORE BHCOPY IS CALLED

SEE PAGE 79 PROGRAM LINES 260 - 290

VPEEK

FORMAT

```
CALL VPEEK(ADDRESS, NUMERIC VARIABLE LIST)
```

DESCRIPTION

The VPEEK subroutine allows to read directly the contents of the memory adresses in the VDP RAM. To the first NUMERIC VARIABLE of the variable list will be assigned the content of that memory location, which has been called via ADDRESS. All subsequent variables of the variable list get assigned the bytes of the appropriate subsequent addresses. The value set of one byte is 0 through 255.

It has to be considered, that the address range of the VDP RAM is 0 through 16383. Entering higher values can cause malfunctions of the computer.

EXAMPLES

Character patterns are stored in VDP area >03F0 through >077F (decimal: 1008 through 1919). The definition of one character requires 8 bytes. Beginning with the address 1008 and the subsequent 7 addresses is stored the character pattern identifier of the cursor (ASCII code 30). It may be read by:

```
100 CALL VPEEK(1008,A,B,C,D,E,F,G,H)
110 PRINT A;B;C;D;E;F;G;H
```

Resulting appear in screen 8 decimal numbers:

```
0   124   124   124   124   124   124   124
```

The appropriate values in hexadecimal notation are:

```
00    7C    7C    7C    7C    7C    7C    7C
```

Used in a CALL CHAR statement (with an ASCII code > 31) will result the shape of the cursor again.

The proposal of a computer magazine is to change the cursor as following:

```
CALL VPOKE(1008,60,126,219,255,231,189,195,126)
```

Beginning on address 1008 (1. value) will be stored 8 subsequent decimal values, corresponding to the hexadecimal pattern identifiers, e used in CALL CHAR statements.

By the way, the cursor modification will be preserved also after the break of programs.

VPOKE

FORMAT

CALL VPOKE(ADDRESS,NUMERIC VALUE LIST)

DESCRIPTION

The VPOKE subprogram allows to write bytes directly to addresses of the VDP RAM.

Numeric data will be subsequently written into memory beginning with the location ADDRESS.

Uncovered use of VPOKE can result malfunctions of the computer. A system crash may occur, which forces the user to switch off his system for a moment, getting it running again. However, this will cause the total loss of program and data, if no back up is availlable.

EXAMPLES

The screen image with 32 x 24 = 768 characters is contained in VDP segment 0000 through 02FF (decimal 0 through 767). The characters are stored with an offset of 96 of their ASCII codes. This means the stored byte is the ASCII code of the appropriate character plus 96.

To poke the word "COMPUTER" in the screen following command has to be entered:

```
   CALL VPOKE(355,163,175,173,176,181,180,165,178)
```

355 is the screen address (0 represents the upper left, 767 the lower right corner of the screen). Beginning with this address the ASCII codes plus 96 of the "COMPUTER" characters will be subsequently stored. The value 163 is the result of the sum of ASCII code 67 (for C) plus the offset 96.

More details about VDP RAM use will be found in the Addendum (part 4).

GPEEK

FORMAT

GPEEK(ADDRESS,NUMERIC VARIABLE LIST)

DESCRIPTION

The GPEEK subroutine allows to read the contents of addresses in GROMs in the computer. The functions are corresponding to CALL PEEK.

More details about GROMs and the prorgramming language GPL, also used in the TI-99/4A, can be found in the book:

TI-99/4A INTERN

written by Heiner Martin. (Verlag fur Technik und Handwerk GmbH, Baden-Baden, Germany, 1985.)

ALLSET

FORMAT

CALL ALLSET

DESCRIPTION

CALL ALLSET resets the characters with the ASCII codes 32 through 126 to their initial definitions.

CALL ALLSET works correspondingly to the statement CALL CHARSET from Extended BASIC, but considers also the lower cases. This can be interesting e.g. in programs being loaded after the execution of ones, used redefinded character images.

WAIT

FORMAT

```
CALL WAIT(DURATION)
```

DESCRIPTION

CALL WAIT causes delays. DURATION can get assigned a value from 0 to 16382. The delay time in seconds is the value of DURATION devided by 50. So the assignment of 1000 represents a delay of 20 seconds.

The delay will be broken if any key is pressed. Values exceeding the allowed range do not get useful time delays or cause an error message.

MOVE

FORMAT

CALL MOVE(MODUS,START ADDRESS,TARGET ADDRESS,BYTES)

DESCRIPTION

CALL MOVE allows to move contents of memory blocks within the RAM. There are four different modes available:

1 = Source VDP RAM, drain VDP RAM
2 = Source VDP RAM, drain CPU RAM
3 = Source CPU RAM, drain VDP RAM
4 = Source CPU RAM, drain CPU RAM

Mode values lower than 1 or higher than 4 cause an error message.

CALL MOVE can be used to save the screen. If there are loaded no assembly language programs the screen area can be moved (copied) by:

```
   CALL MOVE(2,0,8192,768)
```

into the Low Memory Expansion and can be removed by:

```
   CALL MOVE(3,8192,0,768)
```

back to the screen.

Uncovered use of CALL MOVE may cause malfunctions of the computer. Total loss of program and data can be the result. Parts of program or variables may be overwritten. Detailed knowledge of the memory mapping use is essentially before applying CALL MOVE.

EXAMPLES

```
 100 REM *** MOVING DISPLAY ***
 110 CALL CLEAR
 120 CALL VPOKE(1072,233,153,233,137,142,0,0,0,46,48,44,34,220,0,0,0)
 130 DISPLAY AT(2,2):"Moved screen segment by": :" CALL MOVE"
 140 DISPLAY AT(20,2):"That is the power of": :"   MECHATRONIC
 ": :"  EXTENDED BASIC II%'"
 150 CALL MOVE(1,608,448,160)
 160 FOR X=1 TO 450
 170 CALL MOVE(1,164,163,445)
 180 NEXT X
 190 CALL WAIT(200)
 200 GOTO 140
```

MSAVE

FORMAT

CALL MSAVE("FILE NAME",START ADDRESS,BYTES)

DESCRIPTION

CALL MSAVE saves segments of the CPU RAM contents in program format to an external device.

This allows also to store assembly programs on cassette recorder.

```
)   CALL MSAVE("CS1",8192,8192)
```

saves the complete available RAM segment for assembly language programs. There may be saved up to 8192 bytes. Of course, as FILE NAME can be used every valid file name, like "DSK1.SCREEN".

EXAMPLES

```
 100 REM *** SAVE SCREEN ***
 110 CALL CLEAR 'Test also without this line
 120 REM DEFINE PLUS MARK
 130 CALL VPOKE(1072,233,153,233,137,142,0,0,0,46,48,44,34,220,0,0,0)
 140 REM WRITE TEXT
 150 CALL HCHAR(10,1,32,160)
 160 CALL VPOKE(363,173,165,163,168,161,180,178,175,174,169,163)
 170 CALL VPOKE(423,165,184,180,165,174,164,165,164,128,162,161,179,
 169,163,128,1 69,169,134,135)
 180 CALL WAIT(100)
 190 REM MOVE SCREEN IMAGE
 200 CALL MOVE(2,0,8192,768)
 210 CALL CLEAR
 220 PRINT "SCREEN IMAGE      WILL BE      SAVED ON DISKETTE.....": :"..
 AND WILL BE RELOADED NOW"
 230 CALL MSAVE("DSK1.SCREEN",8192,768)
 240 REM ..GET IT BACK
 250 CALL MLOAD("DSK1.SCREEN")
 260 CALL MOVE(3,8192,0,768)
 270 CALL WAIT(300)
 280 REM COPY SEGMENT
 290 CALL MOVE(1,363,5,100)
 300 CALL WAIT(300)
```

MLOAD

FORMAT

```
CALL MLOAD("FILE NAME",MODUS)
```

DESCRIPTION

CALL MLOAD loads program files into CPU RAM being saved before by CALL MSAVE. So CALL MLOAD is just the opposite statement of CALL MSAVE.

As FILE NAME may be used any valid file name.

MODUS may be left out in Extended BASIC.

The numeric MODUS value 0 will cause an automatic start of an assembly language program saved in program format. Before starting, the VDP RAM in this case will be organized using the assembler standard mapping.

CALL MLOAD can load anv file in program format (also BASIC programs), without causing an error message. However, correct working is only possible with assembler files in program format or RAM contents, being saved before by CALL MSAVE.

CALL MSAVE and CALL MLOAD claim a pritty big area of the VDP RAM, which sometimes can cause a memory overflow.

BYE

FORMAT

```
CALL BYE
```

DESCRIPTION

With the statement CALL BYE can be quit the BASIC mode of the TI-99/4A. CALL BYE has the same function as the command BYE, excepted that it can be used in programs.

NEW

FORMAT

```
CALL NEW
```

DESCRIPTION

CALL NEW as a statement clears memory and screen und prepares the computer for input of new programs. After execution of the CALL NEW statement appears * READY *,on the screen. Opposite to the command NEW from Extended BASIC the CALL NEW statement allows a program controlled erasing of the loaded BASIC program.

RESTORE

FORMAT

```
CALL RESTORE(NUMERIC VARIABLE)
```

DESCRIPTION

CALL RESTORE prepares the computer for the next DATA statement to be considered. VARIABLE contains that line number, the computer starts to look for the next DATA statement, executing a READ statement.

The RESTORE statement of Extended BASIC can only use a numeric value (not a variable), which can cause a relatively high programming effort.

CAUTION

If numeric values instead of numeric variables are used for line numbers in the CALL RESTORE statement, there will be no line number correction after execution of a RESEQUENCE command.

QUITOF - QUITON

FORMAT

```
CALL QUITOF
CALL QUITON
```

DESCRIPTION

CALL QUITOF suspends the function of the QUIT key (FCTN =). This means, that unwished pressing of QUIT causes not longer fatal actions like the total loss of program and data.

CALL QUITON reactivates the QUIT function. This means after pressing the QUIT key, the master screen will appear.

ACHTUNG

CALL QUITOF is not reset by the operating system. This effect will be preserved even after execution of BYE or NEW. The QUIT function can only be reactivated by executing CALL QUITON or switching off the system for some moments.

)

SPROF — SPRON

FORMAT

CALL SPROF
CALL SPRON

DESCRIPTION

CALL SPROF stopps any motion of all sprites at once.

CALL SPRON reactivates the motion of all sprites at once. Of course, the motion of sprites has to been enabled before by CALL MOTION or CALL SPRITE.

CAUTION

CALL SPROF is not reset by the operating system. This effect will be preserved even after execution of BYE or NEW. The function can only be reactivated by executing CALL SPRON or switching off the system for some seconds.

EXAMPLES

```
100 REM *** SPRITE STOP - SPRITE GO ***
110 CALL CLEAR
120 DISPLAY AT(2,2):"SPRITE STOP AND GO using": :" CALL SPROF and
CALL SPRON"
130 A=1
140 CALL CHAR(126,"FFFFFFFFFFFFFFFF")
150 CALL MAGNIFY(2)
160 FOR X=1 TO 4
170 CALL SPRITE(#X,126,2,124,124)
180 NEXT X
190 CALL COLOR(#1,5,#2,7,#3,7,#4,5)
200 CALL LOCATE(#2,124,140,#3,140,124,#4,140,140)
210 CALL WAIT(200)
220 CALL SPROF
230 CALL MOTION(#4,0,124,#3,0,124,#2,0,124,#1,0,124)
240 FOR I=1 TO 5
250 CALL SPRON
260 CALL WAIT(200)
270 CALL SPROF
280 CALL WAIT(200)
290 NEXT I
300 CALL DELSPRITE(ALL)
310 CALL ALLSET
```

SCREENOF - SCREENON

FORMAT

```
CALL SCREENOF
CALL SCREENON
```

DESCRIPTION

CALL SCREENOF is used to switch the screen display off. Hereby the contents of the VDP RAM will not be lost, only the data transfer to the monitor will disabled.

CALL SCREENON reactivates the screen display.

After a program break the screen display will be reactivated automatically.

FIND

FORMAT

CALL FIND("GET-STRING","STRING ARRAY"(),RETURN VARIABLE)

DESCRIPTION

The FIND Subroutine lools in a onedimensional string array for a term assigned to GET STRING. The numeric RETURN VARIABLE contains the element number, in which the term is found. If the term is not found the return variable becomes -1.

EXAMPLES

The following sample demonstrates how fast a 10 digit string will be found in an array with 1000 elements, if the array has to be completely tested.

```
100 REM *** FINDTEST ***
110 CALL CLEAR
120 PRINT "PERFORMANCE TEST OF CALL FIND": : :
130 PRINT "LOADING THE ARRAY": :
140 B$="TESTTESTTEST"
150 DIM A$(1000)
160 FOR X=1 TO 1000
170 A$(1)="TESTTESTTESt"
180 NEXT X
190 A$(1000)="TESTTESTTEST"
200 PRINT "SEARCHING"
210 CALL FIND(B$,A$(),B)
220 PRINT "FOUND IN ELEMENT";B
230 END
```

$(X)→

PRINZIPLE OF THE GRAPHIC

PRINZIPLE OF THE HIGH RESOLUTION GRAPHIC

Mechatronic Extended BASIC II^{PLUS} meets the desire of the TI-99/4A owner for High Resolution Graphics! Using the graphic mode, the powerful graphic capabilities of the computer are displayed.

The addressing of each pixel of the screen becomes possible - all this in 16 foreground and background colors! 40 powerful graphic routines are available and can be applied by Mechatronic Extended BASIC II^{PLUS}.

The graphic copy feature by a matrix printer is implemented by software, too. Generating of the graphics works superfast. All the routines are written in the 16 bit TMS 9900 assembler code. It is creative und fun to watch the graphic mode at work.

For application of the graphic mode the following hardware configuration is essential:

TI-99/4A	+	Mechatronic Extended BASIC II^{PLUS} module
	+	32 K RAM expansion
		useful but not essential:
	+	Floppy disk system
	+	Printer (with graphic capabilities) and appropriate interface

WARNING

The power supply of the RAM expansion must n o t be provided by the TI-99/4A - O V E R L O A D I N G !

WORKING IN THE GRAPHIC MODE

WORKING IN THE GRAPHIC-MODE

The principle of the statements in the graphic mode is very simple. The graphic is a cursor or plotter graphic.

The screen is the drawing sheet, and when initiating the "GRAFIC" mode the cursor or drawing pen will appear at position 1, 120 in the drawing window (left bottom corner), ready to start operating along the horizontal axis at the angle of 0 degrees. Simultaneously it is the 0-point of the system of user defined coordinates.

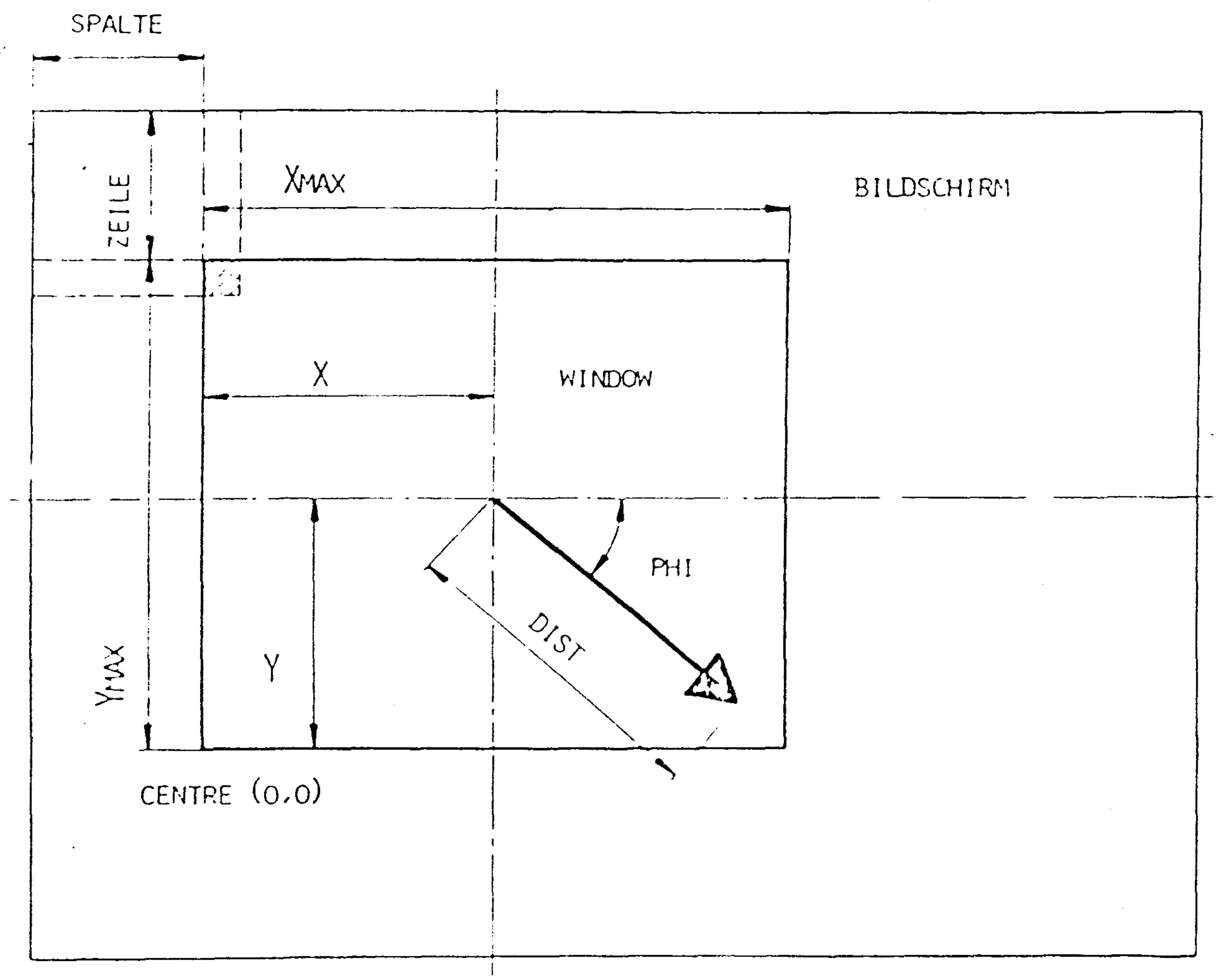
Using simple graphic commands, the cursor is directed across the screen (see fig. 1). All coordinates are identical to mathematical ones.

Thus lines, circles, squares, ellipses, arcs and many other figures can be drawn or erased. Complex geometrical figures, diagrams and so on can be generated on the screen in a very high speed.

All this is available for the TI-99/4A in all colors with the resolution of 256 * 192 pixels. The graphics can be stored on a floppy disk for later use.

Using the graphic mode you have a powerful programming-tool for the interesting realisation of your programs. The graphic statements are simple and easy to learn.

CONTROLLING THE CURSOR



```
CALL LINK ("WINDOW",ROW,COLUMN)
CALL LINK ("SETTO",X,Y)
CALL LINK ("TURNTO",PHI)
CALL LINK ("MOVE",DIST)
```

Fig. 1) PRINZIPLE OF CURSOR CONTROLLING

LOADING THE GRAPHIC INSTRUCTION SET

LOADING THE GRAPHIC INSTRUCTION SET

1) Insert Mechatronic Extended BASIC II PLUS Module
2) Switch on disk drives (if available)
3) Switch on 32K-RAM-Expansion or Peripheral System
4) Switch on Monitor und TI-99/4A
5) Select MECHATRONIC EXTENDED BASIC on main menue
6) Enter > CALL AFESOFT

The entry of graphic programs is now enabled.

WARNING

Entering

> CALL AFESOFT

after loading any BASIC program will erase this program.

CALL AFESOFT closes all opened files, loads the graphic routines into the RAM expansion, reserves required space in the VDP RAM, and executes a "NEW", deleting stored BASIC programs.

NOTE

The number of opened files in graphic mode is limited to 2 files. After entering CALL AFESOFT should no CALL FILES be executed.

ERROR MESSAGES

Error messages in graphic mode are not allways correct, after execution of graphic statements.

All conventional errors are trapped correctly by Mechatronic Extended BASIC II PLUS. Sometimes the TI-99/4A can fail after an interrupt with "FCTN CLEAR" or by choosing a subroutine and can be initialized only by being switched off for a short period.

GRAPHIC STATEMENTS

GRAPHIC STATEMENTS

The BASIC control of the high resolution graphic consists of a number of very powerful commands. They simplify the development of complicated graphics tremendously. They are BASIC support commands and are constituted:

```
CALL LINK("PROCEDURE NAME",PARAMETER *(,PARAMETER,.....)*)
```

CALL LINK connects the BASIC and ASSEMBLER program. PROCEDURE NAME actuates a certain TMS 9900 routine. PARAMETERs are optional.

Terms in *()* can be repeated at random. A block up to 15 parameters can be entered at a time.

Depending on the requirements the following parameters are allowed:

- Numeric constants
- Numeric variables
- Numeric array elements
- Numeric terms
- String constants
- String variables
- String array elements
- String terms

APESOFT

FORMAT

CALL APESOFT

DESCRIPTION

The CALL APESOFT command transfers the high resolution statement set into the RAM expansion - with reservation it is available and operational.

Please note that CALL APESOFT is a command, which can be only executed in direct mode.

CALL APESOFT has following functions:

- All opened files will be closed
- The graphic statements will transferred into RAM expansion.
- Reservation of required VDP RAM space for graphik executions.
- NEW will be executed -
  this will erase any loaded BASIC program.

After execution of CALL APESOFT any CALL FILES should be avoided!

CLRAPE

FORMAT

CALL CLRAPE

DESCRIPTION

CALL CLRAPE restores the initial condition, just like after selection of MECHATNIC EXTENDED BASIC. It can be performed only after execution of CALL APESOFT, otherwise a Syntax Error will be the result.

CALL CLRAPE performs following actions:

- Transacting CALL INIT
- Reset of the VDP RAM
- Closing of all opened files
- Executing a NEW,
  erasing the stored program in memory.

GRAFIC

FORMAT

CALL LINK("GRAFIC",MODUS)

DESCRIPTION

This command signalizes to the computer the graphic mode and initializes all its registers.

A graphic table with maximum 128 vertical and maximum 120 horizontal lines is defined. This table section is in graphic mode available for random addressing of every pixel.

The restriction to 128 * 120 = 15.360 pixels is necessary. Otherwise there would not remain enough storage space in the VDP RAM of the console for BASIC programs, string variables and so on.

In addition the direct addressing of 192 * 256 = 50.176 pixels needs the storage of the 12K bytes VDP RAM: character table and color tables overwrite the buffer addresses of the BASIC interpreter.

Since the graphic table can be transmitted on the screen at random there are 256 * 196 = 50.176 pixels to be addressed individually.

CALL LINK("GRAFIC",MODUS) defines the following internal parameters:

PHI=0	Starting angle of the cursor
TABLEWIDTH	16
FOREGROUND COLOR	Green
BACKGROUND COLOR	Black

Some more internal parameters are depending on MODUS:

```
MODUS = 0      : Graphic mode (255 rows for graphic)
                 15 * 16 table rows and columns
X=1, Y=120     : Starting position of the cursor
                 (0,0)-point

MODUS > 0      : Text mode (192 rows for graphic)
                 12 * 16 table rows and columns
                 The commands "DSPLAY" and "ACCEPT"
                 are available for input/output
                 operations.
X=1, Y=88      : Starting position of the cursor
                 (0,0)-point.
```

GRAFIC

CALL LINK("GRAFIC",MODUS) must be the first command before any graphic statement is entered. As soon as this command is executed the standard characters are lost. The convenional screen I/O commands do not get the expected results.

CALL LINK("BYEBYE") removes the graphic mode and all I/O statements will work as usual.

With BREAK ("FCTN CLEAR") the program is interrupted and the standard status of the computer restored.

CONTINUE

however does not lead back to the graphic mode, unless the first command following CON is CALL LINK("GRAFIC",MODUS)'

EXAMPLES

```
    100 REM SPIRAL
    110 REM ******
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 CALL LINK("SETTO",64,60)
    140 FOR DIST=5 TO 50 STEP 5
    150 CALL LINK("MOVE",DIST)
    160 CALL LINK("TURN",90)
    170 NEXT DIST
    180 GOTO 180
```

draws a pale green squareshaped spiral on a black background.

BYEBYE

FORMAT

```
CALL LINK("BYEBYE")
```

DESCRIPTION

"BYEBYE" removes the graphic mode. It reloads the standard character set and reinitializes the standard mode. SOUNDs and SPRITES can be used again. The Computer works as usual.

Before Executing of another graphil-statement must be passed a new CALL LINK("GRAFIC",MODUS), otherwise the program execution will be interrupted with an error message.

WINDOW

FORMAT

```
CALL LINK("WINDOW",ROW,COLLUMN)

CALL LINK("WINDOW",Z,S,ZA,SA,DZ,DS)
```

DESCRIPTION

This command transmits sections of the graphic table or the complete graphic table to the screen and contains 2 formats.

The total graphic window (16*8 columns, 15*8 rows) is set on the screen position column (0-32) and row (0-24). The graphic window can be positioned partly or totally outside of the screen (24 rows, 32 columns).

Every graphic window, defined before the execution of

```
   CALL LINK("WINDOW",COLUMN,ROW)
```

will be deleted, if one or both parameters are negative. Only the absolute terms of the parameters are evaluated.

The statement

```
   CALL LINK("WINDOW",Z,S,ZA,SA,DZ,DS)
```

transmits sections of the graphic table to the screen.

The parameters mean:

Z,S	Screen row (Z) and column (S) at which the upper left corner point of the graphic table is projected.
ZA,SA	Upper left corner point of the graphic table where the projection will start.
DZ	Number of characters of the graphic table in direction of rows
DS	Number of characters of the graphic table in direction of columns

If Z or S or both are negative, every graphic window which is on the screen will be deleted before the new section is transmitted.

WINDOW

EXAMPLES

```
100 REM CIRCLES
110 REM *******
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",1,1)
140 FOR R=2 TO 42 STEP 2
150 CALL LINK("CIRCLE",64,60,R)
160 NEXT R
170 CALL LINK("WINDOW",12,18)
180 CALL LINK("WINDOW",1,19)
190 FOR I=1 TO 1000
200 NEXT I
210 CALL LINK("SETCOL",16,5)
220 FOR I=1 TO 500
230 NEXT I
240 CALL LINK("INVERT",1,1,128,120)
250 CALL LINK("WINDOW",4,-8)
260 GOTO 260
```

At first a number of concentric circles will appear at the left hand top corner of the screen.
These will be copied line by 170 and 180 downwards to the left and right (trippled). Line 240 will move these circles back to the middle of the screen and the remaining circles are deleted. On its way the graphic is changing its color; it is inverted.

```
100 REM PYRAMIDES
110 REM *********
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",1,1,1,1,10,10)
140 CALL LINK("WINDOW",17,1,1,13,17,13)
142 CALL LINK("WINDOW",13,1)
144 CALL LINK("WINDOW",13,17)
150 CALL LINK("TURNTO",45)
160 CALL LINK("SETTO",64,48)
170 FOR A=1 TO 36 STEP 2
180 CALL LINK("RECT",A,A,-A,A,-A,-A,A-A)
190 NEXT A
200 GOTO 200
```

In this example 4 top views of pyramides are drawn simultaneously. The statements in line 130 and 140 are performing this.

The upper left window is only a fragment because the statement 130 transmits only a section of the graphic table.

SETBLE

FORMAT

```
CALL LINK("SETBLE",WIDTH)
```

DESCRIPTION

This command dimensions the graphic table.

WIDTH	Number of columns of the graphic table It can be varied from 1 to 32.

191 characters in the text mode and 255 characters in the graphic mode are available for the graphic table.

The heigth of the graphic table is depending on the width and is calculated:

HEIGTH = INT(255/WIDTH)	for graphic mode
HEIGTH = INT(191/WIDTH)	for text mode

In this way higher and wider graphics can be generated.

CAUTION

A "WINDOW" statement must follow every "SETBLE" statement. The "WINDOW" statement rearanges the screen, otherwise the graphic generation is not all right.

Simultaneously "SETBLE" defines the centre-point of the system of user defined coordinates at pixel position:

```
CENTRX = 1
CENTRY = HEIGTH * 8 = YMAX
```

CLTBLE

FORMAT

```
CALL LINK("CLTBLE")
```

DESCRIPTION

This command erases the graphic table and also the graphic.

But the table sections which are transmitted by "WINDOW" statements to the screen remain for the input of new graphics.

TABLE

FORMAT

```
CALL LINK("TABLE",Z,S,XMAX,YMAX,BYTES)
```

DESCRIPTION

This statement returns the present parameters of the graphic table to the following variables:

Z	Number of rows of the table
S	Number of columns of the table
XMAX	Maximal pixel columns of the table
YMAX	Maximal pixel rows of the table
BYTES	Number of bytes available for the graphic

Starting with row 1 and column 12 the character bytes in the graphic table are always arranged in ascending order.

The byte number is calculated:

```
CHAR# = (ROW - 1) * WIDTH + ROW - 1
```

ROW	Row of the graphic table
COLUMN	Column of the graphic table
WIDTH	Absolute width of the graphic table
CHAR#	Character byte number of the table

SETCOL

FORMAT

```
CALL LINK("SETCOL",FOREGROUND COLOR,BACKGROUND COLOR)

CALL LINK("SETCOL",N,FG,BG*(,N1,FG1,BG1,...)*)
```

DESCRIPTION

This command has two formats and defines foreground and background colors of the graphic.

All the 16 colors known in BASIC can be used either as foreground or as background colors. Several different foreground and background colors can be used simultaneously in one graphic.

If only two parameters are present in the parameter list, the foreground and background colors are altered simultaneously in the entire graphic.

If there are more then two parameters, "SETCOL" defines the foreground color (FG) and background color (BG) for the charctersets spezified by N.

Hereby the character sets for the rows of the graphic windows are defined as following:

8 following bytes construct a character set (0-7, 8-15, ... usw.).

Within the specifications of the above mentioned table, the foreground and background colors may be defined at random.

Thus a multitude of color combinations is possible.
Using a parameter list, up to 5 color sets can be passed.

SETCOL

Colors for rows and columns of the graphic area (width = 16):

ROWS	COLUMNS	
	1-7	8-16
1	1	2
2	3	4
3	5	6
4	7	8
5	9	10
6	11	12
7	13	14
8	15	16
9	17	18
10	19	20
11	21	22
12	23	24
13	25	26
14	27	28
15	29	30

EXAMPLES

```
    100 REM LEAF
    110 REM ****
    120 RANDOMIZE
    130 CALL LINK("GRAFIC",0)
    135 CALL LINK(WINDOW",3,8)
    140 FOR PHI=0 TO 90 STEP 5
    150 CALL LINK("SETTO",64,119)
    160 CALL LINK("TURNTO",PHI)
    170 CALL LINK("MOVE",1.2*PHI)
    180 CALL LINK("TURNTO",180-PHI)
    190 CALL LINK("SETTO",64,119)
    200 CALL LINK("MOVE",1.2*PHI)
    210 NEXT PHI
    220 FOR DELAY=1 TO 500
    230 NEXT DELAY
    240 FG=2+14*RND
    250 BG=2+14*RND
    260 CALL LINK("SETCOL",FG,BG)
    270 FOR DELAY=1 TO 500
    280 NEXT DELAY
    290 CALL LINK("INVERT",1,1,128,120)
    300 GOTO 220
```

Line 260 defines foreground and background color of the graphic, line 290 inverts foreground and background color.

INVERT

FORMAT

```
CALL LINK("INVERT",X,Y,DX,DY)
```

DESCRIPTION

When "INVERT" is called, foreground and background color of the graphic are swapped. The following parameters are necessary:

X,Y	Pixel position of the upper left corner of the graphic section which is interchanged.
DX	Column pixel position of the section
DY	Row pixel position of the section

Hereby in the section pixels which are set are deleted and vice versa.

CLSCRN

FORMAT

CALL LINK("CLSCRN")

DESCRIPTION

This command is similar in its effect to "CALL CLEAR" in BASIC!
It deletes the graphic, the stored internal cursor parameters remain untouched.

EXAMPLES

```
    100 REM RANDOM STRAIGHT LINES
    110 REM *********************
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",-3,8)
    130 VG =2+13*RND
    140 IF VG =3 THEN 130
    150 CALL LINK("SETCOL",VG,2)
    160 FOR I=1 TO 20
    170 X=124*RND
    180 Y=120*RND
    190 CALL LINK("MOVETO",X,Y)
    200 NEXT I
    210 FOR J=1 TO 250
    210 NEXT J
    220 CALL LINK("CLSCRN")
    230 FOR I=1 TO 500
    240 NEXT I
    250 CALL LINK("WINDOW",1,1)
    260 GOTO 125
```

This program example draws 20 lines successive, choosing direction randomly and starting from position (1,120) (line 190), clears the graphic (line 220) and starts the graphic again choosing the colors at random.

After a short time the statement "WINDOW" (line 250) shows that only the screen with line 220 has been erased, but the graphic in the table has remained untouched.

"WINDOW" with a negative parameter (line 125) effects "CLSCRN", before the section of the graphic table is brought in the screen'

CENTRE

FORMAT

```
CALL LINE("CENTRE",X,Y)
```

DESCRIPTION

This statement defines the system of user defined coordinates:

X .... X-coordinate of the 0-point in the graphic table

Y .... Y-coordinate of the 0-point in the graphic table

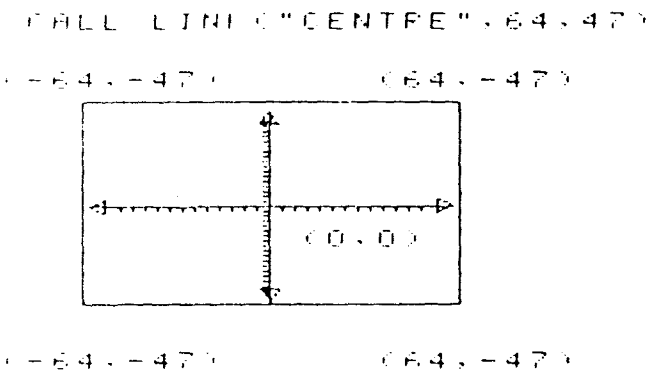


Fig. 2): SYSTEM OF USER DEFINED COORDINATES

After a graphic statement the centre point is exact at position (1,120) (left bottom corner of the table).
With "CENTRE" this 0-point can be moved optionally, even outside of the table'

CENTRE

EXAMPLES

```
100 REM SYSTEM OF COORDINATES
110 REM *********************
120 CALL LINK("GRAFIC",1)
130 CALL LINK("WINDOW",7,8)
132 CALL LINK("SETTO",1,1)
134 CALL LINK("RECT",127,-87)
140 CALL LINK("CENTRE",64,44)
150 CALL LINK("AXIS",0,60,60,4,0,40,40,2)
160 CALL LINK("WRITE",8,10,"(0,0)")
170 CALL LINK("DSPLAY",1,5,26,">CALL LINK(""CENTRE"",
64,44)")
180 CALL LINK("DSPLAY",5,5,22,"(-64,+44)  (64,+44)")
190 CALL LINK("DSPLAY",20,5,22,"(-64,-44)  (64,-44)")
200 OPEN #1:"RS232.BA=9600.DA=8.CR",OUTPUT
210 PRINT #1:CHR$(27)&"A"&CHR$(8)
220 CLOSE #1
230 CALL BHCOPY("RS232.BA=9600.DA=8.CR","L")
240 STOP
```

This sample program produces the drawal of fig. 2 with an EPSON MX 80 or the similar Texas Instruments Line Printer PHP 2500.

SETTO

FORMAT

```
CALL LINK("SETTO",X,Y*(,X1,Y1,...)*)
```

DESCRIPTION

"SETTO" sets pixels at the coordinates row Y, column X. On the screen are columns 1 through 128 and rows 1 through 120.

The range of the values has no restrictions. The coordinates may be positive or negative, its value high or low at random and may also be floating point numbers. They are internally rounded to the nearest integer number.

Numbers greater than 32.768 and less than -32.767 are displayed incorrectly. There will be no error message when this range is exceeded' When a parameter list is used up, to 7 pixels can be defined simultaneously.

The internal angle PHI of the cursor remains unchanged.

RESET

FORMAT

```
CALL LINK("RESET",X,Y*(,X1,Y1,...)*)
```

DESCRIPTION

"RESET" deletes the pixels with the coordinates X,Y. All conditions made for "SETTO" are valid for "RESET", too.

IFSET

FORMAT

CALL LINK("IFSET",X,Y,VAR*(,X1,Y1,VAR1,...)*)

DESCRIPTION

This statement checks whether a pixel with the coordinates X,Y is set and the following values in the variables VAR are stated:

Pixel	(X,Y)	set	VAR=-1
Pixel	(X,Y)	deleted	VAR= 0
Pixel	(X,Y)	outside the graphic window	VAR=+1

With one parameter list, up to 5 pixels can be checked simultaneously. Otherwise the conditions set up for "SETTO" and "RESET" apply.

EXAMPLES

```
100 REM SINE
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
127 CALL LINK("CENTRE",4,60
130 PI180=4*ATN(1)/180
140 CALL LINK("AXIS",0,0,118,5,0,50,50,5)
150 REM
160 FOR PHI=0 TO 360 STEP 2
170 X=PHI/3+20
180 Y=20*SIN(PHI*PI180)
190 CALL LINK("SETTO",X,Y,X,Y*1.5,X,Y*2)
200 NEXT PHI
210 FOR PHI=0 TO 360 STEP 2
220 X=PHI/3+20
230 Y=20*SIN(PHI*PI180)
240 CALL LINK("RESET",X,Y*1.5,X,Y*2)
250 NEXT PHI
260 X=INT(128*RND)+1
270 Y=INT(120*RND)+1
280 CALL LINK("IFSET",X,Y,A)
290 IF A=0 THEN 260
300 CALL LINK("BYEBYE")
310 CALL CLEAR
320 PRINT "POINT X=";X;"Y=";Y;":IS SET"
330 STOP
```

This program example draws 3 curves of sines (160-200), deletes it again (210-250) and stays in a holding loop until it has found a set pixel (280-290).

MOVE

FORMAT

```
CALL LINK("MOVE",DIST)
```

DESCRIPTION

This statement draws a line of the length DIST, starting from the present cursor position with the internal angle PHI. The position DIST horizontally and vertically corresponds exactly with the number of pixels; the number of pixels themselves depends on the set angle. After performing DIST the cursor stops at the end coordinates (last stored pixel). PHI remains unchanged.

Positive values of DIST work in the present direction of the cursor, negative values work 180 degrees opposite. DIST can assume any value, although the range limits apply given under item SETTO.

REMOVE

FORMAT

CALL LINK("REMOVE",DIST)

DESCRIPTION

"REMOVE" has the same effect as "MOVE", but here the pixels from position X,Y up to the DIST distant new position of the cursor are deleted.

EXAMPLES

```
100 REM STAR
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("WRITE",15,3,"THAT IS GRAFHIC")
140 FG=3+13*RND
150 CALL LINK("SETCOL",FG,2)
160 CALL LINK("SETTO",2,60)
170 CALL LINK("TURNTO",36)
180 FOR I=1 TO 10
190 CALL LINK("TURN",108)
200 CALL LINK("MOVE",60)
210 NEXT I
220 CALL LINK("SETTO",2,60)
230 CALL LINK("TURNTO",36)
240 FOR I=1 TO 10
250 CALL LINK("TURN",108)
260 CALL LINK("REMOVE",60)
270 NEXT I
280 GOTO 140
```

Through skilled application of a few commands a star is charmed onto the screen, then deleted and then the game starts again in different colors.

MOVETO

FORMAT

```
CALL LINK("MOVETO",X,Y*(,X1,Y1,...)*)
```

DESCRIPTION

"MOVETO" draws a line from the present internal position of the cursor to the next by the parameter pair X and Y defined position.

The list of parameter can hold a maximum of 7 positions. If there are given more than 2 parameters, the line will always be drawn from the previous position to the next.

After the execution of "MOVETO" the cursor will remain at the last position given by the parameter list. The internal angle and the colors will remain unchanged with "MOVETO". Lines can also be drawn outside of the graphic window border.

REMVTO

FORMAT

```
CALL LINK("REMVTO",X,Y*(,X1,Y1,...)*)
```

DESCRIPTION

"REMVTO" works like "MOVETO" with the difference that here the lines are deleted. All the conditions for "MOVETO" also apply to "REMVTO".

EXAMPLES

```
    100 REM THREAD GRAPHIC
    110 REM *************
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 D=2.6
    140 Z=125
    150 S1=128
    160 FOR S=1 TO 126 STEP 5
    170 Z=Z-5
    180 S1=S1-D
    190 CALL LINK("SETTO",S,120)
    200 CALL LINK("MOVETO",S1,
    210 NEXT S
    220 Z=125
    230 S1=1
    240 FOR S=126 TO 1 STEP -5
    250 Z=Z-5
    260 S1=S1+D
    270 CALL LINK("SETTO",S,120)
    280 CALL LINK("MOVETO",S1,Z)
    290 NEXT S
    300 GOTO 300
```

TURN

FORMAT

```
CALL LINK("TURN",PHI)
```

DESCRIPTION

This command adds to the present internal angle of the cursor the angle PHI in decimal degrees.

The limitations of the angle are +/- 2047 degrees. Internally the angle is modulated from 0-360 degrees.

The trigonometrical functions are generated by the computer via interpolation tables. Since the storage capacity is very limited the angles are interpolated in the range of 5 degrees. This may lead to inexact results when using intermediate values.

TURNTO

FORMAT

CALL LINK("TURNTO",PHI)

DESCRIPTION

This command imperatively sets the internal angle of the cursor to PHI (degrees). All limitations made for "TURN" also apply here.

EXAMPLES

```
100 REM OCTAGONS
110 REM ********
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",4,5)
140 CALL LINK("WINDOW",6,19)
150 DIST=2
160 FOR S=28 TO 108 STEP 4
170 CALL LINK("SETTO",S,42)
180 CALL LINK("TURNTO",90)
190 DIST=DIST+2
200 FOR I=1 TO 8
210 CALL LINK("TURN",45)
220 CALL LINK("MOVE",DIST)
230 NEXT I
240 NEXT S
250 GOTO 250
```

RECT

FORMAT

```
CALL LINK("RECT",A,B*(,A1,B1,...)*)
```

DESCRIPTION

Starting from its present position and the internal angle of the cursor "RECT" draws rectangles with the sequence

A -> B -> A -> B

The rectangle turns clock-wise, if B is positive. The example shows the influence of the operational sign of the side length with reference to its ultimate position.

The internal angle and the position of the cursor are not influenced by "RECT". The side length of the rectangle can take any value. Up to 7 rectangles can be passed with one parameterlist. But they all begin at the same starting position and also finish there.

CLRECT

FORMAT

```
CALL LINK("CLRECT",A,B*(,A1,B1,...)*)
```

DESCRIPTION

Works identically to "RECT" with the difference that "CLRECT" deletes the rectangles.

All conditions of "RECT" apply to "CLRECT".

EXAMPLES

```
    100 REM RECTANGLES
    110 REM **********
    120 CALL LINK("GRAFIC",0)
    115 CALL LINK("WINDOW",3,8)
    130 CALL LINK("SETTO",64,60)
    140 A=40
    150 B=20
    160 CALL LINK("RECT",A,B,A,-B,-A,B,-A,-B)
    170 FOR I=1 TO 250
    180 NEXT I
    190 CALL LINK("CLRECT",A,B,A,-B,-A,B,-A,-B)
    200 CALL LINK("TURN",45)
    210 GOTO 160
```

Line 160 draws 4 rectangles with only one command, line 190 deletes them. Line 200 turns the internal angle on by 45 degrees.

CIRCLE

FORMAT

```
CALL LINK("CIRCLE",X,Y,R*(,X1,Y1,R1,...)*)
```

DESCRIPTION

"CIRCLE" draws a circle with the central point X,Y and the radius R. With one parameterlist up to 5 different circles can be drawn.

The parameters can assume any value. For the radius the absolute value is worked out automatically. If the value for the R = 0, "CIRCLE" sets a point (pixel). After the execution of "CIRCLE" the cursor takes the central point of the last drawn circle.

The internal angle PHI remains unchanged.

Due to internal rounding errors the circular arcs may appear not quite smooth.

CLCRCL

FORMAT

```
CALL LINK("CLCRCL",X,Y,R*(,X1,Y1,R1,...)*)
```

DESCRIPTION

"CLCRCL" works identically to "CIRCLE", but here the circles are deleted. All the conditions for "CIRCLE" also apply to "CLCRCL".

EXAMPLES

```
    100 REM CIRCLES
    110 REM *******
    120 PI=4*ATN(1)
    130 CALL LINK("GRAFIC",0)
    135 CALL LINK("WINDOW",3,8)
    137 CALL LINK("CENTRE",64,60)
    140 CALL LINK("CIRCLE",0,0,30)
    150 FOR PHI=0 TO 2*PI STEP PI/16
    160 CALL LINK("CIRCLE",30*COS(PHI),30*SIN(PHI),30)
    170 NEXT PHI
    180 GOTO 180
```

ARCUS

FORMAT

```
CALL LINK("ARCUS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI1,...)*)
```

DESCRIPTION

"ARCUS" draws circular arcs with the following parameters:

X,Y	Centre point of the arc
R	Radius of the arc
PHI	Starting angle of the arc (absolute)
DPHI	Arc angle of the arc

Simultaneously three arcs can be generated with one command. Due to internal rounding errors and generating the trigonometrical functions via interpolation tables the results are not always satisfying.

The coordinates of the cursor describe the last drawn arc pixel after the execution of "ARCUS".

CLARCS

FORMAT

```
CALL LINK("CLARCS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI1,...)*)
```

DESCRIPTION

"CLARCS" works identically to "ARCUS" the difference being that "CLARCS" deletes all arc pixels.

ELLIPS

FORMAT

```
CALL LINK("ELLIPS",X,Y,A,B*(,X1,Y1,A1,B1,...)*)
```

DESCRIPTION

"ELLIPS" draws ellipses with the axis centre point X,Y of the big semi axis A and small semi axis B and the inclination PHI of the big semi axis.

A maximum of three different ellipses can be drawn with one parameterlist. The parameters can assume any values exept 0. The absolute value is automatically used for the big and small semi axis. After the execution of "ELLIPS" the cursor assumes the coordinates of the semi axis points of intersection. The internal angle PHI remains unchanged.

Through internal rounding errors the elliptical arcs may sometimes not appear quite smooth. This occurs particularly when the main axes are inclined to the horizontal or vertical, because the coordinate transformations are carried out by interpolated trigometrical functions.

CLLIPS

FORMAT

```
CALL LINK("CLLIPS",X,Y,A,B*(,X1,Y1,A1,B1,...)*)
```

DESCRIPTION

"CLLIPS" works like "ELLIPS", with the difference, that here the ellipses are deleted. All the conditions listed for "ELLIPS" are appropriate valid.

EXAMPLES

```
100 REM CONE
110 REM ****
120 CALL LINK("GRAFIC",0)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK("SETCOL",15,2)
140 M=1
150 A=42
160 B=22
170 FOR Y=81 TO 1 STEP -8
180 IF M=0 THEN 220
190 M$="ELLIPS"
200 REM
210 GOTO 230
220 M$="CLLIPS"
230 CALL LINK(M$,64,Y,A,B)
240 A=A-4
250 B=B-2
260 NEXT Y
270 IF M=1 THEN 140
180 M=0
290 GOTO 150
```

If M=1, a cone is always drawn due to the control commands 270-290. M$, a string variable, can also be passed as "PROCEDURE NAME".

The program is broken on entering "FCTN CLEAR".

VALUES

FORMAT

```
CALL LINK("VALUES",X,Y,PHI,FG,BG)
```

DESCRIPTION

"VALUES" returns the present internal parameters to the variable list.

```
   X ...... Cursor column
   Y ...... Cursor row
   PHI .... Cursor angle
   FG ..... Foreground color
   BG ..... Background color
```

As the angle is modulated internally, it is always between 0-360 degrees independent of the previous input.

EXAMPLES

```
   100 REM EXAMPLE
   110 REM *******
   120 CALL LINK("GRAFIC",0)
   125 CALL LINK("WINDOW",3,8)
   130 FOR I=1 TO 11
   140 CALL LINK("SETTO",64,60)
   150 CALL LINK("MOVE",30)
   160 CALL LINK("VALUES",X,Y,PHI,FG,BG)
   170 CALL LINK("CIRCLE",X,Y,20)
   180 CALL LINK("TURN",30)
   190 NEXT I
   200 GOTO 200
```

From the centre point of the graphic window, program line 150 draws a line with the length 30.

Line 160 determines the final cursor position, line 170 takes this position as centre point for a circle with radius 20.

AXIS

FORMAT

```
CALL LINK("AXIS",X,LENXR,LENXL,DELTAX,Y,LENYU,LENYD,DELTAY)
```

DESCRIPTION

"AXIS" draws a system of coordinates with the following parameters:

X,Y	Centre point of coordinates
LENXL	Left hand side X-semi-axis (length)
LENXR	Right hand side X-semi-axis (length)
DELTAX	Pitch of the X-grid
LENYU	Top Y-semi-axis (length)
LENYD	Bottom Y-semi-axis (length)
DELTAY	Pitch of the Y-grid

All values are taken as absolute values. If one of the semi axis has the value 0, then this semi axis is not drawn.

If the value for the grid equals 0 or more than that of the corresponding semi axis, no grid will be drawn.

After the execution of "AXIS" the cursor will assume the position at the centre point of the coordinate system.

The internal angle PHI is altered. The system of coordinates may not be completely on the screen.

EXAMPLES

```
    100 REM ZYKLOM
    110 REM ******
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 PI2=8*ATN(1)
    140 CALL LINK("AXIS",8,0,116,4,60,60,59,4)
    150 X=7
    160 CALL LINK("SETTO",X,40)
    170 FOR PHI=0 TO 3*PI2 STEP PI2/16
    180 X=X+2
    190 Y=20*(SIN(2*PHI)+2*COS(PHI))+60
    200 CALL LINK("MOVETO",X,Y)
    210 NEXT PHI
    220 GOTO 220
```

HSTDIA

FORMAT

```
CALL LINK("HSTDIA",X,Y,WIDTH,HEIGTH,DEPTH)
```

DESCRIPTION

"HSTDIA" draws a block diagram with the following parameters:

```
    X,Y ....... Coordinates of the left bottom corner
                of the block
    WIDTH ..... Width of block diagram
    HEIGTH .... Heigth of block diagram
    DEPTH ..... Depth of block diagram
```

Only the absolute values are token.

EXAMPLES

```
    100 REM HISTOGRAMS
    110 REM **********
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 FOR N=2 TO 20
    140 CALL LINK("SETCOL",N,14,2)
    150 NEXT N
    160 CALL LINK("AXIS",8,10,120,8,20,90,0,4)
    170 CALL LINK("HSTDIA",16,22,12,80,6)
    180 CALL LINK("HSTDIA",40,22,12,45,6)
    190 CALL LINK("HSTDIA",70,22,12,67,6)
    200 CALL LINK("HSTDIA",94,22,16,12,6)
    210 CALL LINK("WRITE",15,3,"HISTOGRAMS")
    220 GOTO 220
```

CRCDIA

FORMAT

```
CALL LINK("CRCDIA",X,Y,RADIUS,PHI,DPHI*(,X1,Y1,PHI1,DPHI1,...)*)
```

DESCRIPTION

"CRCDIA" draws a circular diagram with the following parameters:

X,Y	Coordinates of circular segment centre point
RADIUS	Radius of circular segment
PHI	Start angle of circular segment
DPHI	Final angle of circular segment

Only the absolute values are regarded.

EXAMPLES

```
    100 REM CIRCULAR DIAGRAMS
    110 REM ****************
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 FOR N=1 TO 13 STEP 2
    140 CALL LINK("SETCOL",N,16,2,N+1,14,2,N+14,9,2,N+15,
    13,2)
    150 NEXT N
    160 CALL LINK("CRCDIA",64,60,38,180,276)
    170 CALL LINK("CRCDIA",68,60,38,276,450)
    180 CALL LINK("CRCDIA",68,64,38,90,180)
    190 CALL LINK("WRITE",14,2,"CIRC-DIAGRAMS")
    200 GOTO 200
```

WRITE

FORMAT

```
CALL LINK("WRITE",Z,S,STRING*(,Z1,S1,STRING1,....)*)
```

DESCRIPTION

"WRITE" enables the mixing of graphic and text.

At position row (Z) and column (S) of the graphic window "WRITE" displays a string (STRING).

```
   Limitations: Z ...... 1  to  15
                S ...... 1  to  16
```

If the string is too long for the appropriate line, the rest will be cut. A maximum of 5 strings can be entered with one parameter list. The ASCII codes (upper cases, lower cases, digits and special characters) apply.

DSPLAY

FORMAT

```
CALL LINK("DSPLAY",Z,S,SIZE,VAR$)
```

DESCRIPTION

This statement corresponds to the well known EXTENDED BASIC command "DISPLAY AT". It is using the following parameters:

Z	Row of screen	(1-24)
S	Column of screen	(1-32
SIZE	Length of the string	(max. 32)
VAR$	String variable	(max. 32) characters)

With this statement SIZE characters of the string VAR are transmitted to the screen position (Z,S). Hereby graphic values located under the string are erased (difference to "WRITE"!).

If SIZE is negative, SIZE positions are not deleted before the output of the string.

"DSPLAY" is in graphic mode available, only if the mode has been set <> 0.

ACCEPT

FORMAT

```
CALL LINK("ACCEPT",Z,S,SIZE,VAR$)
```

DESCRIPTION

This statement corresponds to the well known EXTENDED BASIC statement "ACCEPT AT" and is using the following parameters:

Z	Row of the screen	(1-24)
S	Column of the screen	(1-32)
SIZE	Length of the string	(max. 32 characters)
VAR$	String variable	(max. 32 characters)

"ACCEPT" accepts SIZE characters of a string at the position (Z,S). If SIZE is positive SIZE positions are deleted previously.

During the input the following keys are active:

UALPHA	ASCII codes (32-96)
<-	Cursor left
->	Cursor right
ERASE	deletes the input
ENTER	Accepts the string
REPEAT	Repeat function

CAUTION

"FCTN QUIT" and "FCTN CLEAR" are ineffective during the execution of this command.

While accepting all key codes can be addressed, but for text purposes only UALPHAs (upper cases) are useful.

ACCEPT

EXAMPLES

```
100 REM STAR1
110 REM *****
120 CALL LINK("GRAFIC",1)
125 CALL LINK("WINDOW",3,8)
130 CALL LINK ("SETCOL",12,2)
140 CALL LINK("SETTO",5,60)
150 CALL LINK("MOVE",120)
160 CALL LINK("SETTO",64,1)
170 CALL LINK("TURN",90)
180 CALL LINK("MOVE",120)
190 S1=5
200 Z1=60
210 Z2=36
220 S2=64
230 S3=124
240 Z3=60
250 Z4=84
260 S4=64
270 FOR I=1 TO 10
280 CALL LINK("SETTO",S1,Z1)
290 CALL LINK(MOVETO",S2,Z2,S3,Z3,S4,Z4,S1,Z1)
300 S1=S1+4
310 Z2=Z2-4
320 S3=S3-4
330 Z4=Z4+4
340 NEXT I
350 CALL LINK("WRITE",15,1,"DAS IST GRAPHIK")
360 CALL LINK("DSPLAY",17,3,27,"BEI EINGABE VON STOP WIRD")
370 CALL LINK("DSPLAY",18,3,27,"DIE GRAPHIK ABGEBROCHEN!")
380 CALL LINK("ACCEPT",22,3,4,M$)
390 IF M$  STOP THEN 120
400 STOP
```

SHIFT

FORMAT

```
CALL LINK("SHIFT",DELTAX,DELTAY)
```

DESCRIPTION

This statement performs a linear transformation of the graphic.

```
   DELTAX ......... Movement in X-direction (collumns)
                    (positive values rightwards)
   DELTAY ......... Movement in Y-direction (rows)
                    (positive values downwards)
```

CAUTION

The graphic windows in the screen will not be transformed!

GSAVE

FORMAT

```
CALL GSAVE("FILE NAME")
```

DESCRIPTION

With this statement it is possible to save screen displavs.

Graphics can only be saved on floppy disks. Any valid file name mav be used. The file name can be entered as a string or a string variable. The name of the file may not be longer than 9 characters.

The saved file is formatted in the "MEMORY IMAGE" format. The colors of the graphic and the position of the graphic window are also stored on total 3 consequent files.

GLOAD

FORMAT

```
CALL GLOAD("FILE NAME")
```

DESCRIPTION

This statement loads a graphic called "FILE NAME" from diskette into VDP RAM, unless being previously stored with "GLOAD".

Using CALL GLOAD without executing previous a "GRAFIC" statement will cause an error message and program execution interrupt.

EXAMPLES

```
    100 REM SAVE GRAPHIC
    110 REM ************
    120 CALL LINK("GRAFIC",0)
    125 CALL LINK("WINDOW",3,8)
    130 FOR R=2 TO 40
    140 CALL LINK("CIRCLE",64,60,R)
    150 NEXT R
    160 CALL GSAVE("DSK2.SAVETEST")
    170 STOP

     NEW

    100 REM LOAD GRAPHIC
    110 REM *************
    120 CALL LINK("GRAFIC",0)
    130 CALL GLOAD("DSK2.SAVETEST")
    154 GOTO 140
```

The previous example causes the generation of the following files on DSK2.:

"SAVETEST"	Contains grapfic table and parameters
"SAVETEST1"	Contains a screen dump
"SAVETEST2"	Contains colors of the graphic

By saving and loading the graphic only the first file name, in our case "SAVETEST" must be stated. The input of the second or third file name leads to incorrect functions.

The statement "WINDOW" can be cancelled by loading the graphic because every window on the screen is overwritten by the loaded graphic.

BHCOPY

FORMAT

```
CALL BHCOPY("FILE NAME","ESC-SEQUENCE")
```

DESCRIPTION

"BHCOPY" produces screen dumps with dot matrix printers like EPSON or compatible ones in "BIT IMAGE MODE".

In this way the creation of graphics by matrix printers becomes very easy and fast. These hardcopy routines work in the standard mode of the Extended BASIC II^{PLUS} too.

The parameters are:

FILE NAME	Printer options usual: "RS232.BA=9600.DA=8.CR" or "PIO.CR" The dip switches of the printer have to be set accordingly
ESC-SEQUENCE	Printer adjustment e.g.: "K" or "L" for normal or doobble density in BIT IMAGE MODE

CAUTION

The file name extension .CR is essentially for correct working of the routines. The printer must have set the inverse signal "AUTO FEED XT" internal to "ON".

Using serial interfaces (RS232) the extension .DA=8 must be added. The printer needs to be set to 8 data bits, too.

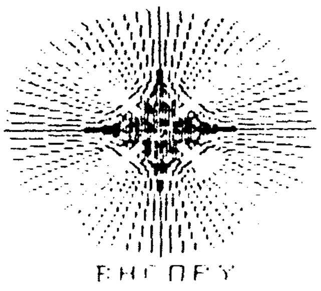
According to the ESC sequence has to be considered the manual of the attached printer' BHCOPY sends at the begin of every line CHR$(27) (=ESC), than the ESC-Sequence (max. 10 characters), than the characters CHR$(0) and CHR$(1) (for the subsequent 256 Bytes), afterwards the 256 Bytes, corresponding to the hardcopy of one screen line and closing CHR$(13) (for CR). This respects the data transfer sequence of TI Line Printer (EPSON) using the Bit Image Mode.
Format-Samples:

```
     CALL BHCOPY("RS232.BA=9600.DA=8.CR","K")
     CALL BHCOPY("PIO.CR","L")
```

BHCOPY

EXAMPLES

```
100 REM EPSONCOPY
110 REM *********
120 CALL LINK("GRAFIC",0)
130 CALL LINK("WINDOW",3,8)
140 CALL LINK("CENTRE",64,60)
150 FOR PHI=0 TO 355 STEP 5
160 CALL LINK("SETTO",0,0)
170 CALL LINK("TURNTO",PHI)
180 CALL LINK("MOVE",50)
190 NEXT PHI
200 CALL LINK("WRITE",15,6,"BHCOPY")
210 PORT$="RS232.BA=9600.DA=8.CR"
215 REM
220 OPEN #1:PORT$,OUTPUT
230 PRINT #1:CHR$(27);"A";CHR$(8)
240 CLOSE #1
245 REM
250 CALL BHCOPY(PORT$,"K")
260 STOP
```



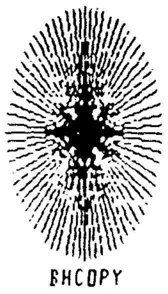


Fig. 3) BHCOPY

The reproduction of the radiating wreath on the printer is generated by "BHCOPY" in the proportion 1:1 in single density ("K"), but is deformed elliptically using double density ("[illegible]").

HARDCOPY DEMONSTRATION 1

HARDCOPY DEMONSTRATION PROGRAM

In the following a program example called "HCOPYDEMO". This program demonstrates how various graphic patterns are created.

This program is a useful aid studiing the output of graphic to a matrix printer.

```
 100 REM   ***********************
 110 REM  *************************
 120 REM  **                     **
 130 REM  **  H C O P Y D E M O  **
 140 REM  **                     **
 150 REM  *************************
 160 REM   ***********************
 170 REM
 180 REM by APESOFT 19850301
 200 REM
 220 PORT$="RS232.BA=9600.DA=8.CR" ' SCHNITTSTELLENOPTION
 230 REM
 240 REM LINE FEED REDUCTION TO 8 PIXEL ROWS
 250 REM ***********************************
 260 LF8$=CHR$(27);"A";CHR$(8)
 270 OPEN #1:PORT$,OUTPUT
 280 PRINT #1:LF8$
 290 CLOSE #1
 300 REM
 310 REM PATTERN FOR THE "§APESOFT" COPYRIGHT SIGN
 320 REM ****************************************
 340 DATA 96,000001070F1F1F3F,97,3F3F7F7F7F7F7F7F,98,003FFFFFFFFFFFF
 F,99.E3C1808080C1E3FF
 350 DATA 100,030F1F3F3F7F7F7F,101,7F7F7F7F7F7F7F7F,102,60787C7E7E7E7
 F7F,103,7F7F7E7E7E7C7860
 360 DATA 104,7F7F7F7F7F7F7F7F
 370 DATA 106,030F1F3F3F3F7F7F,107,7F7F7F3F3F1F0F03,108,60787C7E7E7E7
 F7F,109,00007E7E7E7C7860
 380 DATA 110,030F1F3F3F3F7F7F,111,0000003F3F1F0F03,112,E0F8FCFEFEFE8
 080,113,FFFFFEFEFEFCF8E0
 390 DATA 114,030F1F3F3F3F7F7F,115,7F7F7F3F3F1F0F03,116,60787C7E7E7E7
 F7F,117,7F7F7F7E7E7C7860
 400 DATA 118,000000030F1F3F3F,119,0000001C3E7F7F7F,120,3F7F7F7F7F7F7
 F7F,121,7F7F7F7F7F7F7F7F
 410 DATA 122,3E1C000000006060,123,6060606
 420 DATA 124,0000007F7F7F7F70,125,0000000000000006,126,7F7F7F7F7F7F7
 F7F,127,7F7F7F3F3F1F0F03
 430 DATA 128,6060606060000000,129,001C3E7F7F7F3E1C
 440 DATA 130,00FCFFFFFFFFFFFFF,131,9F878381818080 80,132,000080E0F0F8F
 8FC,133,FCFCFEFEFEFEFEFE
 450 DATA 134,7F7F7F7F7F7F7F3F,135,3F3F1F1F0F070100,136,8080C0C0C0E0F
 0FC,137,FFFFFFFFFFFFFF3F
 460 DATA 138,8080808080808080,139,FFFFFFFFFFC0FFFF,140,FEFEFEFEFEFEF
 CFC,141,FCF8F8F0F000FFFF
```

HARDCOPY DEMONSTRATION 2

```
470 REM
480 REM TRANSFER OF APE CHARACTER
485 REM ************************
490 CALL CLEAR
500 CALL SCREEN(2)
510 READ I,N$
520 CALL CHAR(I,N$)
530 IF I<141 THEN 510
540 PRINT TAB(9);"MICROCOMPUTER": :
550 PRINT TAB(7);"`b";CHR$(130);CHR$(132);"          vwöü"
560 PRINT TAB(7);"ac";CHR$(131);CHR$(133);"dfjlnprtxzß";CHR$(128)
570 PRINT TAB(7);CHR$(134);CHR$(136);CHR$(138);CHR$(140);"egkmoqsuy
a";CHR$(127);CHR$(129)
580 PRINT TAB(7);CHR$(135);CHR$(137);CHR$(139);CHR$(141);"h"
590 PRINT :TAB(11);"SOFTWARE": : : : : : : : : : : : : : : :
600 CALL SCREEN(4)
610 CALL BHCOPY(PORT$,"K")
615 CALL ALLSET
620 REM
630 REM QUADER
640 REM ******
650 CALL LINK("GRAFIC",0)
660 CALL LINK("CENTRE",1,-2)
670 CALL LINK("WINDOW",1,1)
680 CALL LINK("TURNTO",41.8103)
690 FOR Y=-30 TO -95 STEP -3
700 CALL LINK("SETTO",X,Y)
710 CALL LINK("MOVE",40)
720 NEXT Y
730 Y=-55
740 CALL LINK("TURNTO",90)
750 FOR X=32 TO 117 STEP 3
760 Y=Y+1
770 CALL LINK("SETTO",X,Y)
780 CALL LINK("MOVE",65)
790 NEXT X
800 CALL LINK("TURNTO",-18.4349)
810 Y=-30
820 FOR X=1 TO 33 STEP 3
830 CALL LINK("SETTO",X,Y)
840 CALL LINK("MOVE",92)
850 Y=Y-2.5
860 NEXT X
870 CALL BHCOPY(PORT$,"K")
```

HARDCOPY DEMONSTRATION 3

```
880 REM
890 REM PYRAMID
900 REM *******
910 CALL LINK("CLTBLE")
920 CALL LINK("CENTRE",64,1)
930 YG=-96
940 FOR XG=-63 TO -20 STEP 2
950 CALL LINK("SETTO",0,0)
960 CALL LINK("MOVETO",XG,YG)
970 YG=YG-1
980 NEXT XG
990 CALL LINK("SETTO",0,0)
1000 CALL LINK("REMVTO",XG,YG)
1010 FOR XG=XG+3 TO 46 STEP 3
1020 CALL LINK("SETTO",0,0)
1030 CALL LINK("MOVETO",XG,YG)
1040 YG=YG+3
1050 NEXT XG
1070 CALL BHCOPY(PORT$,"K")
1080 REM
1090 REM CYLINDER
1100 REM ********
1110 CALL LINK("CLTBLE")
1120 CALL LINK("SETBLE",12)
1130 CALL LINK("WINDOW",-1,4)
1140 CALL LINK("CENTRE",48,1)
1150 CALL LINK("TURNTO",0)
1160 FOR Y=-120 TO -40 STEP 3
1170 CALL LINK("ELLIPS",0,Y,48,24)
1180 NEXT Y
1190 FOR R=48 TO 2 STEP -2
1200 CALL LINK("ELLIPS",0,Y,R,R/2)
1210 NEXT R
1230 CALL BHCOPY(PORT$,"K")
```

HARDCOPY DEMONSTRATION 4

```
1240 REM
1250 REM CYKLONE
1260 REM *******
1270 PI2=8*ATN(1)
1280 CALL LINK("GRAFIC",1)
1290 CALL LINK("WINDOW",5,8)
1300 CALL LINK("CENTRE",1,40)
1310 CALL LINK("AXIS",8,116,0,4,0,36,40,4)
1320 X=7
1330 CALL LINK("SETTO",X,-20)
1340 FOR PHI=0 TO 3*PI2 STEP PI2/16
1350 X=X+2
1360 Y=20*(SIN(2*PHI)*COS(PHI)*2)
1370 CALL LINK("MOVETO",X,Y)
1380 NEXT PHI
1390 CALL LINK("DSPLAY",16,9,8,"AVERAGE")
1400 CALL LINK("DSPLAY",17,9,13,"VALUES")
1410 M$="TEMPERATURE"
1420 FOR Z=5 TO 15
1430 CALL LINK("DSPLAY",Z,7,1,SEG$(M$,Z-4,1))
1440 NEXT Z
1450 CALL LINK("WRITE",10,3,"day-degrees")
1470 CALL BHCOPY(PORT$,"K")
1480 REM
1490 REM HISTROGRAMS
1500 REM ***********
1510 CALL LINK("GRAFIC",0)
1520 CALL LINK("SETBLE",24)
1530 CALL LINK("WINDOW",1,1)
1540 CALL LINK("HSTDIA",1,68,4,8,2,8,56,8,16,4,24,32,12,32,8)
1550 CALL LINK("HSTDIA",50,0,18,56,11)
1570 CALL BHCOPY(PORT$,"K")
1580 REM
1590 REM TORUS
1600 REM *****
1610 CALL LINK("CLTBLE")
1620 CALL LINK("CENTRE",96,40)
1630 PIARC=4*ATN(1)
1640 FOR PHI=0 TO 2*PIARC STEP PIARC/40
1650 CALL LINK("CIRCLE",66*SIN(PHI),20*COS(PHI),17)
1660 NEXT PHI
1680 CALL BHCOPY(PORT$,"K")
```

HARDCOPY DEMONSTRATION 5

```
1690 REM
1700 REM FLOWERS
1710 REM *******
1720 CALL LINK("GRAFIC",0)
1730 CALL LINK("WINDOW",1,4)
1740 CALL LINK("CENTRE",54,48)
1750 CALL LINK("SETTO",0,0)
1760 FOR PHI=0 TO 22.5 STEP 22.5
1770 CALL LINK("TURNTO",PHI)
1780 FOR X=1 TO 7
1790 FOR W=0 TO 15
1800 CALL LINK("MOVE",4)
1810 CALL LINK("TURN",W)
1820 NEXT W
1830 FOR W=16 TO 4 STEP -1
1840 CALL LINK("MOVE",4)
1850 CALL LINK("TURN",W)
1860 NEXT W
1870 CALL LINK("MOVETO",0,0)
1880 CALL LINK("TURN",6)
1890 NEXT X
1900 NEXT PHI
1910 FOR R=1 TO 12
1920 CALL LINK("CIRCLE",0,0,R)
1930 NEXT R
1950 CALL BHCOPY(PORT$,"K")
1960 REM ***
1965 LF$=CHR$(10)' LINE FEED
1970 OPEN #1:PORT$,OUTPUT
1975 PRINT #1:CHR$(27);CHR$(2)' RESET TO STANDARD LINE FEED
1980 PRINT #1:LF$;LF$;LF$;LF$;" H I G H   R E S O L U T I O N";LF$;
LF$
1990 PRINT #1:LF$;"           G R A P H I C";LF$;LF$
2010 CLOSE #1
2020 CALL LINK("BYEBYE")
2030 END
```

QUICK REFERENCE

ALPHABETIC QUICK REFERENCE

The order of statements disregards any preceeding CALLs or CALL LINKs.

CALL LINK("ACCEPT",Z,S,SIZE,VAR$)

accepts SIZE characters of the string VAR$ at position (Z,S).

CALL ALLSET

resets the ASCII characters 32 to 126 to their standard definitions.

CALL APESOFT

transfers the high resolution graphic routines into the RAM expansion. This command must be entered before loading of BASIC programs.

CALL LINK("ARCUS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI1,...)*)

draws arcs with the centre point X,Y, the radius R, the starting angle PHI and the arc angle DPHI.

CALL LINK("AXIS",X,LENXL,LENXR,DELTAX,Y,LENYU,LENYD,DELTAY)

draws a system of coordinates with the centre point X, Y, the left X-semi-axis LENXL, the right X-semi-axis LENXR, the pitch of the X-grid DELTAX, the upper Y-semi-axis LENYU, the bottom Y-semi-axis LENDY and the pitch of the Y-grid DELTAY.

CALL BHCOPY("FILE NAME","ESC-SEQUENCE")

generates screen copies on EPSON or compatible printers in "BIT IMAGE MODE". "FILE NAME" respect the interface options and "ESC-SEQUENCE" means the pRinter adjustment.

QUICK REEFERENCE

CALL BYE

erases the loaded programs and data and calls the master screen. The BASIC-Mode will be left.

CALL LINK("BYEBYE")

cancels the graphic mode, loads the standard character sets and reestablishes standard mode of Extended BASIC II^{PLUS}.

(CALL LINK("CENTRE",X,Y)

defines the system of user defined coordinates with the (0,0)-point at the position (X,Y) of the graphic table.

CALL LINK("CIRCLE",X,Y,R*(,X1,Y1,R1,...)*)

draws a circle with the centre point X,Y and the radius R.

CALL CLRAPE

initializes Extended BASIC just like after selction of MECHATRONIC EXTENDED BASIC from the main menue. It may only be entered after execution of CALL APESOFT otherwise it will cause a syntax error.

CALL LINK("CLARCS",X,Y,R,PHI,DPHI*(,X1,Y1,R1,PHI1,DPHI1,...)*)

erases the arc pixels.

CALL LINK("CLCRCL",X,Y,R*(,X1,Y1,R1,...)*)

erases the circles.

CALL LINK("CLLIPS",X,Y,A,B*(,X1,Y1,A1,B1,...)*)

erases the ellipses.

CALL LINK("CLRECT",A,B*(,A1,B1,...)*)

erases the rectangles

QUICK REFERENCE

CALL LINK("CLSCRN")

clears the screen. The graphic in the table together with all other internal parmeters remain.

CALL LINK("CLTBLE")

clears the graphic table and thus the graphic.

CALL LINK("CRCDIA",X,Y,RAD,PHI,DPHI*(,X1,Y1,RAD1,PHI1,DPHI1,...)*)

draws a circular diagram with the coordinates of the circular segment centre point X,Y, the radius RAD, the starting angle PHI and the final angle of the circular segment DPHI.

CALL LINK("DSPLAY",Z,S,SIZE,VAR$)

sets SIZE characters of the string VAR$ at position (Z,S).

CALL LINK("ELLIPS",X,Y,A,B*(,X1,Y1,A1,B1,...)*)

draws ellipses with the axis centre point X,Y, of the big semi axis A, of the small semi axis B and the inclination PHI of the big semi axis.

CALL FIND("GET-STRING","STRING ARRAY"(),RETURN VARIABLE)

is looking in a one-dimensional string array for "GET-STRING". The RETURN VARIABLE carries the number of the wanted element. If WANTSTRING is not founded, RETURN VARIABLE gets the value -1.

CALL GLOAD("FILE NAME")

loads a graphic called "FILE NAME" from diskette into VDP-RAM.

CALL GPEEK(ADDRESS, NUMERIC VARIABLE LIST)

reads the contents of subsequent addresses in GROMs of the TI-99/4A.

QUICK REFERENCE

CALL LINK("GRAFIC",MODUS)

signalizes the graphic mode, initializes all the computer registers and defines a graphic table (max. 128 vertical and 120 horizontical lines) depending on the MODUS (graphic or text mode).

CALL GSAVE("FILE NAME")

saves the colors of the graphic, the position of the graphic window, and the graphik parameters in "MEMORY IMAGE" format on diskette.

CALL LINK("HSTDIA",X,Y,WIDTH,HEIGTH,DEPTH)

draws a block diagram with the coordinates of the left bottom corner of the block X,Y, the width WIDTH, the heigth HEIGTH and the depth DEPTH.

CALL LINK("IFSET",X,Y,VAR*(,X1,Y1,VAR1,...)*)

checks if a pixel with the coordinates X, Y is set or not and returns the result to the variable VAR.

CALL LINK("INVERT",X,Y,DX,DY)

inverts sections of the graphic; with X, Y the pixel position of the upper left corner, with DX the pixel column position and with DY the pixel row position of the inverted graphic section are stated.

CALL MLOAD("FILE NAME",MODUS)

loads saved CPU RAM contents back into the CPU of the TI-99/4A. Extended BASIC II PLUS doesn't require MODUS. On positive values of "MODUS" a self start of assembly language program file execution will be performed.

CALL MOVE(MODUS,START ADDRESS,TARGET ADDRESS,BYTES)

moves the contents of memory blocks with the length BYTES depending of the MODUS (1-4) between VDP RAM and CPU RAM or within VDP RAM or CPU RAM.

QUICK REFERENCE

CALL LINK("MOVE",DIST)

draws a line of the length DIST, starting from the present position X, Y of the cursor with the present internal angle PHI.

CALL LINK("MOVETO",X,Y*(,X1,Y1,...)*)

draws a line from the present internal position of the cursor to the next one by the parameter pair X,Y defined position.

CALL MSAVE("FILE NAME",START ADDRESS,BYTES)

saves memory blocks of the CPU-RAM with the length BYTES to an external device in program image format.

CALL NEW

erases the BASIC program and data in RAM and prepares the computer for receiving of new BASIC programs.

CALL QUITOF

desactivates the QUIT function (FCTN =).

CALL QUITON

reactivates the desactivated QUIT function.

CALL LINK("RECT",A,B*(,A1,B1,...)*)

draws rectangles in the order A - B - A - B using the actual cursor parameters.

CALL LINK("REMOVE",DIST)

deletes all pixels from position X,Y up to the DIST distant new position of the cursor.

CALL LINK("REMVTO",X,Y*(,X1,Y1,...)*)

deletes the lines

QUICK REFERENCE

CALL LINK("RESET",X,Y*(,X1,Y1,...)*)

erases pixels with the coordinates X,Y.

CALL RESTORE(NUMERIC VARIABLE)

prepares the computer to process the next DATA Statement using the line number carried by the NUMERIC VARIBLE.

CALL SCREENOF

switches the screen off.

CALL SCREENON

reactivates the screen.

CALL LINK("SETBLE",BREITE)

dimensions the graphil table.

CALL LINK("SETCOL",FOREGROUND COLOR,BACKGROUND COLOR)

changes foreground and background color simultaneously for the entire graphic.

CALL LINK("SETCOL",N,FG,BG*(,N1,FG1,BG1,...)*)

defines the foreground color (FG) and background color (BG) for the character set specified by N.

CALL LINK("SETTO",X,Y*(,X1,Y1,...)*)

sets pixels with the coordinates X and Y.

CALL LINK("SHIFT",DELTAX,DELTAY)

transformes a graphic linear by the values DELTAX in column direction and DELTAY in row direction.

QUICK REFERENCE

CALL SPROF

stopps the movement of all sprites at once.

CALL SPRON

restarts all stopped sprites.

CALL LINK("TABLE",Z,S,XMAX,YMAX,BYTES)

returns the present parameters of the graphic table to the given variables, whereby Z corresponds to the number of rows, S to the number of columns, XMAX to the maximal pixel columns, YMAX to the maximal pixel rows of the table, and BYTES to the number of available bytes.

CALL LINK("TURN",PHI)

adds to the present internal angle of the cursor the angle PHI in decimal degrees. PHI turns clockwise.

CALL LINK("TURNTO",PHI)

sets the internal angle imperatively to PHI (degrees).

CALL LINK("VALUES",X,Y,PHI,FG,BG)

returns the present internal parameters to the variable list, whereby X means the cursor column, Y the cursor row, PHI the cursor angle, FG the foreground color and BG the background color.

CALL VPEEK(ADDRESS,NUMERIC VARIABLE LIST)

reads the contents of the subsequent addresses of the VDP RAM, starting with ADDRESS, and returns them to the NUMERIC VARIABLE LIST.

CALL VPOKE(ADDRESS,NUMERIC DATA)

writes numeric data direct into subsequent addresses of the VDP RAM, starting with ADDRESS.

QUICK REFERENCE

CALL WAIT(DURATION)

results delays with those to the numeric variable DURATION assigned number of 1/50 seconds.

CALL LINK("WINDOW",ROW,COLUMN)

sets the entire graphic table (16*8 columns, 15*8 rows or 11*8 rows in the text mode) at position column (1-32) and row (1-24).

CALL LINK("WINDOW",Z,S,ZA,SA,DZ,DS)

transmits sections of the graphic table to the screen; Z means the row, S the column, ZA and SA mean the upper left corner point of the graphic table, DZ the number of graphic table characters in row direction and DS the number of graphic table characters in column direction.

CALL LINK("WRITE",Z,S,STRING*(,Z1,S1,STRING1,...)*)

enables the mixing of graphic and text in the graphic table and writes a string (STRING) at position row (Z) and column (S) of the graphic table.

REFERENCE LIST

REFERENCE LIST FOR MECHATRONIC EXTENDED BASIC II^{PLUS}

Following abbreviations are used:

(D) allowed only in D)irect mode
(C) subroutine preceeding C)ALL
(L) assembler routine preceeding CALL L)INK

reference name	D	C	L	in Mode STRD	GRFC	see ref.manual of XBASIC	XBASIC II
ABS				X	X	X	
ACCEPT				X		X	
ACCEPT			L		X		X
ALLSET		C		X			X
APESOFT	D	C		X			X
ARCUS			L		X		X
ASC				X	X	X	
ATN				X	X	X	
AXIS			L		X		X
BHCOPY		C		X	X		X
BREAK				X	X	X	
BYE	D			X		X	
BYE		C		X	X		X
BYEBYE			L	X	X		X
CALL				X	X	X	
CENTRE			L		X		X
CHAR		C		X	X	X	
CHARPAT		C		X	X	X	
CHARSET		C		X		X	
CHR$				X	X	X	
CLARCS			L		X		X
CLEAR		C		X		X	
CLLIPS			L		X		X
CLOSE				X	X	X	
CLRAPE	D	C		X	X		X
CLSCRN			L		X		X
CLTBLE			L		X		X
COINC		C		X		X	
COLOR		C		X		X	
CON	D			X		X	
CONTINUE	D			X		X	
COS				X	X	X	
CRCDIA			L		X		X

REFERENCE LIST

reference	!		!	in	Mode	!see ref.manual	of !
name	!D	C	L!	STRD	GRFC	!XBASIC	XBASIC II !
DATA				X	X	X	
DEF				X	X	X	
DELETE				X	X	X	
DELSPRITE		C		X		X	
DIM				X	X	X	
DISPLAY				X		X	
DISPLAY USING				X		X	
DISTANCE		C		X		X	
DSPLAY			L		X		X
ELLIPS			L		X		X
END				X	X	X	
EOF				X	X	X	
ERR		C		X	X	X	
EXP				X	X	X	
FILES		C		X	X	X	
FIND		C		X	X		X
FOR TO STEP				X	X	X	
GCHAR		C		X	X	X	
GLOAD		C			X		X
GOSUB				X	X	X	
GOTO				X	X	X	
GRAFIC			L	X	X		X
GPEEK		C		X	X		X
GSAVE		C			X		X
HCHAR		C		X		X	
HSTDIA			L		X		X
IFSET			L		X		X
IF-THEN-ELSE				X	X	X	
IMAGE				X	X	X	
INIT		C		X		X	
INPUT				X		X	
INT				X	X	X	
INVERT			L		X		X
JOYST		C		X	X	X	
KEY		C		X	X	X	

REFERENCE LIST

reference				in	Mode	see ref.manual of	
name	D	C	L	STRD	GRFC	XBASIC	XBASIC II
LEN				X	X	X	
LET				X	X	X	
LINK		C		X	X	X	
LINPUT				X		X	
LIST	D			X		X	
LOAD		C		X	X	X	
LOCATE		C		X		X	
LOG				X	X	X	
MAGNIFY		C		X		X	
MAX				X	X	X	
MERGE	D			X		X	
MIN				X	X	X	
MLOAD		C		X	X		X
MOTION		C		X		X	
MOVE		C		X	X		X
MOVE			L		X		X
MSAVE		C		X	X		X
NEW	D			X		X	
NEW		C		X	X		X
NEXT				X	X	X	
NUM	D			X		X	
NUMBER	D			X		X	
OLD	D			X		X	
ON BREAK				X	X	X	
ON ERROR				X	X	X	
ON-GOSUB				X	X	X	
ON-GOTO				X	X	X	
ON WARNING				X	X	X	
OPEN				X	X	X	
OPTION BASE				X	X	X	
PATTERN		C		X		X	
PEEK		C		X	X	X	
PI				X	X	X	
POS				X	X	X	
POSITION		C		X		X	
PRINT				X		X	
PRINT USING				X		X	
QUITON			L	X	X		X
QUITOF			L	X	X		X

REFERENCE LIST

reference	!		!	in	Mode	!see ref.manual	of !
name	!D	C	L!	STRD	GRFC	!XBASIC	XBASIC II !
RANDOMIZE				X	X	X	
READ				X	X	X	
REC				X	X	X	
REM				X	X	X	
RES	D			X		X	
RESEQUENCE	D			X		X	
RESET			L		X		X
RESTORE				X	X	X	
RESTORE		C		X	X		X
RETURN				X	X	X	
RND				X	X	X	
RPT$				X	X	X	
RUN				X	X	X	
SAVE	D			X		X	
SAY		C		X	X	X	
SCREEN		C		X	X	X	
SCREENON		C		X	X		X
SCREENOF		C		X	X		X
SEG$				X	X	X	
SETBLE			L		X		X
SETCOL			L		X		X
SETTO			L		X		X
SGN				X	X	X	
SIN				X	X	X	
SIZE	D			X		X	
SOUND		C		X		X	
SPGET		C		X	X	X	
SPRITE		C		X		X	
SPRON		C		X			X
SPROF		C		X			X
SQR				X	X	X	
STOP				X	X	X	
STR$				X	X	X	
SUB				X	X	X	
SUBEND				X	X	X	
SUBEXIT				X	X	X	
TAB				X		X	
TABLE			L		X		X
TAN				X	X	X	
TRACE				X		X	
UNBREAK				X	X	X	
UNTRACE				X		X	

REFERENCE LIST

reference				in	Mode	see ref.manual	of
name	D	C	L	STRD	GRFC	XBASIC	XBASIC II
VAL				X	X	X	
VALUES			L		X		X
VCHAR		C		X		X	
VERSION		C		X	X	X	
VPEEK		C		X	X		X
VPOKE		C		X	X		X
WAIT		C		X	X		X
WINDOW			L		X		X
WRITE			L		X		X

MEMORY MAPPING

Commands of the extended statement set of Mechatronic Extended BASIC II^{PLUS} like CALL VPEEK, VPOKE, MOVE, MSAVE, MLOAD require detailed knowledge of the system organisation, if their whole power shall be used.

In the following is written a short survey of the memory mapping of the TI-99/4A, which may give the user some orientation. Anyway, this Extended BASIC reference manual can not provide details of the operating system or the BASIC interpreter. More detailed information can be found in the Editor/Assembler manual.

MEMORY MAPPING OF THE TI-99/4A

The TMS 9900 microprocessor has an adress space of 64k bytes. In the Home Computer, some of this adress space contains RAM and some contains ROM. In addition, some adresses are used for access to special devices, such as sound and speech, and other areas of memory, such as VDP RAM and GROMs.

In the following the memory map directly adressable:

CPU Memory Use - General Case

>0000	(Console ROM) Two 4k ROM chips	0000
>2000	Low Memory Expansion (8 kByte)	8192
>4000	Peripheral ROMs (mapped) up to 8k bytes for Device Service Routine	16384
>6000	Application ROMs in Command Module	24576
>8000	Memory-mapped devices for VDP, GROM, Sound, and Speech	32768
>A000	High Memory Expansion (24 kByte)	40960
>FFFF		65535

ROMS AND GROMS

ROMs

All the ROMs (Read Only Memory) are directly accessable by an assembly language program. Two 4k byte console ROMs are located at adresses >0000 through >1FFFF. They contain the operating system, the GPL interpreter, and parts of the TI BASIC interpreter.

GROMs

A GROM (Graphics Read Only Memory) is another type of ROM. It is designed to contain GPL (GPL = Graphics Programming Language) programs which are executed by the GPL interpreter in the console. GPL is commonly used in applications software and can only be executed through a GROM.

VDP-RAM

The Video Display Processor (Random Access Memory) RAM, located in the console, is chiefly used for common video functions, such as screen images, character pattern tables, color table etc.

When Extended BASIC is in use, VDP RAM also contains the BASIC program the program symbol table, the value stack, and the string space. The VDP RAM is also used as storage space by application programs. Part of VDP RAM is used as a data buffer. Another part of VDP RAM functions as a PAB (Peripheral Access Block) to pass information from a file to appropriate DSR (Device Service Routine). Assembly language programs cannot be executed from VDP RAM.

VDP RAM is a memory-mapped area of 16k (16384 or >4000) bytes numbered >0000 through >3FFF. VDP RAM adresses are automatically incremented, so only one adress in CPU RAM is required to read or write a specific block of data.

VDP MEMORY USE

VDP MEMORY USE BY EXTENDED BASIC

>0000		0000
	Screen	
>02FF		0767
>0300		0768
	Sprite Attribute List	
>0370		0880
>0371		0881
	BASIC Temporaries	
>03EF		1007
>03F0		1008
	Character Tables	
>077F		1019
>0780		1920
	Sprite Motion Table	
>07FF		2047
>080F		2063
	Color Table	
>081F		2079
>0820		2080
	BASIC Crunch Buffer	
>0957		2391
>0958		2392
	Value Stack	
	String Space	
	Symbol Tabels	
	Line Number Table	
	Crunched Programm	
	PAB	
>3FFF		16383

REMARKS TO VDP RAM USE

REMARKS TO VDP RAM USE

Screen Images

The 768 Characters (32 columns x 24 rows) of screen are aranged line by line by the adresses 0 through 767 starting in the upper left corner.

Every character is represented by one byte. This byte corresponds to an offset of 96 to the value of the ASCII Code of the displayed character. (ASCII value + 96).

Sprite Attributes

In this area are stored the data of the sprites # 1 through 28 subsequently. Every sprite is represented by 4 bytes in following order:

1. Y-Position (Pixel row 1 = 255)
   (Pixel row 2 = 0)
   (Pixel row 3 = 1) etc.
2. X-Position (Pixel column 1 = 0)
   (Pixel column 2 = 1) etc.
3. Character Code (Offset of ASCCI codes of sprite character)
4. Color Code (BASIC color code minus 1)

Character Patterns

The images of the ASCII codes 30 through 143 are subsequently by 8 byte defined. The character definition correspondes to that of BASIC with one exception: Each byte in hexadecimal notation has to be converted in decimal notation. (1 Byte = 2 hex digits. >00 - >FF corresponds to dec 00 - 255).

Example:

```
CALL CHAR(32,007C7C7C7C7C7C7C)

CALL VPOKE(1024,0,124,124,124,124,124,124,124)
```

If no sprite motion occurs, the ASCII Codes 144 - 159 may be applied.

REMARKS TO VDP RAM USE

Sprite Motions

In this area are the velocities of the sprites #1 through #28 subsequently stored. Every sprite is represented by 4 bytes in the following order:

1. Vertical speed
2. Horizontal speed
3. Used by operating system
4. Used by operating system

Color Tabel

Informations about both, foreground color and background color, of the character sets 0 through 14 are subsequently stored in one byte. The value is the result of foreground color code minus 1 times 16 plus background color code minus 1. E. g. entering CALL VPOKE(2063,96) results a dark red cursor during program execution.

Crunch Buffer

Beginning with address 2240 through 2391 are the maximum 151 characters of the last keyboard entry subsequently stored (by their offsets), which may be recalled by REDO (FCTN 8).

BASIC Programs

User depending data as there are crunched BASIC programm lines and the appropriate line numbers, symbol tabels, string space etc. do not have fixed start addresses. They depend on the system configuration and the program itself.

CAUTION

Uncovered use of the statements VPOKE or MOVE commonly causes a system crash in connection with the total loss of stored program and data. Therefore, before starting experiments backing up the used programs is strongly recommended.