

# MYARC 128K.0S EXTENDED BASIC II LEVEL IV

## Enhanced and additional commands to TI Extended BASIC

### CHAR subprogram

#### Format

CALL CHAR (character-code, pattern-string[,...])

#### Cross Reference

CHARPAT, CHARSET, COLOR, DCOLOR, GRAPHICS, HCHAR, SCREEN, SPRITE, VCHAR, WRITE

#### Description

The CHAR subprogram enables you to define your own characters so that you can create graphics on the screen.

CHAR is the inverse of the CHARPAT subprogram.

If you use HCHAR or VCHAR to display a character on the screen and then later use CHAR to change the definition of that character, the result depends on the graphics mode:

In Pattern and Text Modes, the displayed character changes to the newly defined pattern.

In High-Resolution Mode, the displayed character remains unchanged.

#### Pattern and High-Resolution Modes

In Pattern and High-Resolution Modes, each character is composed of 64 pixels in a grid eight pixels high and eight pixels wide, as explained below.

In Pattern Mode, you can use the DISPLAY, DISPLAY USING, PRINT, and PRINT USING instructions and the HCHAR and VCHAR subprogram to display characters on the screen.

In High-Resolution Mode, you can use the HCHAR, VCHAR, and WRITE subprogram to display characters on the screen. Only characters 0-215 are available.

#### Text Mode

In Text Mode, each character is composed of 48 pixels in a grid eight pixels high and six pixels wide. The eight by eight grid described below is used to define characters; however, the last two pixels in each pixel-row are ignored.



In Text Mode, you can use the DISPLAY, DISPLAY USING, PRINT, and PRINT USING instructions and the HCHAR and VCHAR subprogram to display characters on the screen, You cannot display sprites in Text Mode.

## **CIRCLE subprogram**

Format

```
CALL CIRCLE (line-type, pixel-row, pixel-column, radius  
[, pixel-row2, pixel-column2, radius2[,...]])
```

cross reference

DCOLOR, DRAW, DRAWTO, FILL, GRAPHICS, POINT, RECTANGLE, WRITE

Description

The CIRCLE subprogram enables you to draw or erase circles around a specified point.

Note that CIRCLE draws and erases the outside edges (perimeter) of a circle, not the area it encircles,

Line-type is a numeric-expression whose value specifies the action taken by the CIRCLE subprogram.

Type Action:

2 Reverses the status of each pixel of the circle. (If a pixel is on, it is turned off; if a pixel is off, it is turned on). This effectively reverses the color of the circle.

1 Draws a circle of the foreground color specified by the DCOLOR subprogram. This is accomplished by turning on each pixel of the specified circle.

0 Erases a circle. This is accomplished by turning off each pixel of the specified circle.

Pixel-row and pixel-column are numeric expressions whose values represent the screen portion of the point the circle is drawn around (center-point). Radius is a numeric expression whose value represents

the distance from the center-point of the circle to it's outer edge.

You can optionally draw more circles by specifying additional sets of pixels (center-points), and additional radii.

Pixel-row must have a value from 1 to 192. Pixel-column must have a value from 1 to 256. Radius must have a value from 1 to 320.

The pixel-row/pixel-column you specify (center-point), becomes the current position used by the DRAWTO subprogram.

CIRCLE can only be used in High-Resolution Mode. An error results if you



use CIRCLE in Pattern or Text Modes.

## **CLEAR subprogram**

Format

CALL CLEAR

Cross Reference

DCOLOR, DELSPRITE

Description

The CLEAR subprogram erases the screen.

In Pattern and Text Modes, CLEAR places a space character (ASCII code 32) in every screen position.

In High-Resolution Mode, CLEAR erases the screen by turning off all pixels and restoring the default graphics colors (black on transparent).

## **COLOR subprogram**

Format

CALL COLOR (#sprite-number, foreground-color[,...1)

Cross Reference

CHAR, DCOLOR, GRAPHICS, SCREEN, SPRITE

Description

The COLOR subprogram enables you to specify the colors of characters or sprites.

If a color is transparent, the color actually displayed is the color specified by the SCREEN subprogram.

Pattern Mode

In Pattern Mode, the 256 available characters are divided into 32 sets of 8 characters each. When you assign a color combination to a particular set, you specify the colors of all 8 characters in that set.

The character-set is a numeric-expression whose value specifies the number (0-31) of the 8-character set.

Foreground-color and background-color are numeric expressions whose values specify colors that can be assigned from among the 16 available colors.

An error occurs if you use the COLOR subprogram to assign character colors in Text Mode. Use the SCREEN subprogram to assign character colors in Text Mode.

In Text Mode, using the COLOR program to assign colors to sprites has no



effect (Text Mode does not display sprites).

## High-Resolution Mode

In High-Resolution Mode, you can use COLOR only to assign colors to sprites; any other use of the COLOR subprogram causes an error. Use the DCOLOR subprogram to specify character and graphics colors in High-Resolution Mode.

## Sprites

A sprite is assigned a foreground-color when it is created with the SPRITE subprogram. The background-color of a sprite is always transparent.

To re-assign colors to sprites you must use the sprite parameters, no matter what graphics mode the computer is in.

The sprite-number is a numeric expression whose value specifies the number of a sprite as assigned by the SPRITE subprogram.

Foreground-color is a numeric expression whose value specifies a color that can be assigned from among the 16 available colors.

## DCOLOR subprogram Draw Color

### Format

CALL DCOLOR (foreground-color, background-color)

### Cross Reference

CIRCLE, COLOR, DRAW, DRAWTO, FILL, GRAPHICS, HCHAR, POINT, RECTANGLE, VCHAR  
WRITE

### Description

The DCOLOR subprogram enables you to set the graphics colors.

The graphics colors are used by the CIRCLE, DRAW, DRAWTO, FILL, HCHAR, POINT, RECTANGLE, VCHAR, and WRITE subprogram in High-Resolution Mode.

Foreground-color and background-color are numeric expressions whose values specify colors that can be assigned from among the 16 available colors.

When you enter MYARC Extended BASIC II, the foreground-color is set to black and the background-color is set to transparent. These default graphics colors are restored only when you change graphics mode. They are not restored when you enter RUN.

DCOLOR is effective only in High-Resolution Mode. DCOLOR has no effect in Pattern or Text Mode.

## DEFINT

### Format

DEFINT numeric-variable-list



DEFINT ALL

Cross Reference

DEF, DIM, OPTION BASE, REAL, SUB

Description

The DEFINT instruction enables you to declare the data-type of specified numeric variables as DEFINT.

DEFINT variables are processed faster and require less memory than do REAL variables.

You can use DEFINT as either a program statement or a command.

The numeric-variable-list consists of one or more numeric variables separated by commas. The variables are all assigned the DEFINT data-type. A DEFINT statement with a numeric-variable-list must have a lower line number than any program reference to any variable in that list.

If you enter the ALL option, all numeric variables in your program are assigned the DEFINT data-type unless specifically declared as REAL. A DEFINT statement with the ALL option must have a lower line

number than any program reference to any numeric variable or array.

A DEFINT ALL statement in your main program does not affect the data-type of a numeric variable in a subprogram.

A numeric variable of the DEFINT data-type is a whole number greater than or equal to -32768 and less than or equal to 32767.

## **DELETE**

Format

DELETE file-specification

Cross Reference

CLOSE

Description

The DELETE instruction removes a file from an external storage device. Although the file is not physically erased, the space it occupies becomes available for you to store another file in the future.

You can use DELETE as either a program statement or a command,

The file-specification indicates the name of the file to be deleted. The file-specification is a string expression; if you use a string constant, you must enclose it in quotation marks,

DELETE has no effect on a file stored on an audio cassette.



You can also remove files stored on some external devices by using the DELETE option in the CLOSE instruction,

For more information about the options available with a particular device, refer to the owner's manual that comes with that device.

## **DIM Dimension**

### Format

```
DIM array-name(integer1[,... integer7])[,array-name... ]data-types
```

### Cross Reference

DEFINT, OPTION BASE, REAL

### Description

The DIM instruction enables you to dimension (reserve space for) arrays. If a program includes an OPTION BASE 1 statement, the first element is element 1, so the number of elements is equal to the integer plus 1.

A string array cannot have more than 16383 elements. For numeric arrays, a DEFINT array cannot have more than 32767 elements and a REAL array cannot have more than 8191 elements. The number of integers in parentheses following the array-name determines the number of dimensions (1-7) in the array.

You can optionally specify the data-type (DEFINT or REAL) of a numeric array by replacing DIM with the data-type.

You cannot use OPTION BASE as a command.

If you reference an array without first using a DIM instruction to dimension it, each dimension is assumed to have 11 elements (elements 0-10), or 10 elements (elements 1-10) if your program includes an OPTION BASE 1 statement.

## **DRAW subprogram**

### Format

```
CALL DRAW (line-type, pixel-row1, pixel column1, pixel-row2, pixel- column2  
[, pixel-row3, pixel column3, pixel-row4, pixel -column4[,...]])
```

### Cross Reference

CIRCLE, DCOLOR, DRAWTO, FILL, GRAPHICS, POINT, RECTANGLE, WRITE

### Description

The DRAW subprogram enables you to draw or erase lines between specified pixels.

The value of the numeric-expression line-type specifies the action taken by the DRAW subprogram.



#### TYPE ACTION:

1 Draws a line of the foreground-color specified by the DCOLOR subprogram. This is accomplished by turning on each pixel in the specified line.

0 Erases a line. This is accomplished by turning off each pixel in the specified line.

2 Reverses the status of each pixel on the specified line. (If a pixel is on, it is turned off; if

a pixel is off it is turned on.) This effectively reverses the color of the specified line.

Pixel-row and pixel-column are numeric expressions whose values specify the pixels to be connected by the line. You must specify at least two pixels to define the beginning and end points of a line.

Pixel-row must have a value from 1 to 192. Pixel-column must have a value from 1 to 256.

You can optionally draw more lines by specifying additional pairs of pixels. The lines are not connected; each line extends from the first pixel of the pair to the second pixel of the pair. You must specify an even number of pixels.

The last pixel you specify becomes the current position used by the DRAWTO subprogram.

DRAW can be used only in High-Resolution Mode. An error results if you use DRAW in Pattern or Text Mode.

In High-Resolution Mode the computer divides each pixel-row into 32 groups of 8 pixels each. (This is most obvious when you assign a background color other than cyan or transparent.) The computer can assign 1 foreground color and 1 background color, from among the 16 available colors, to each 8-pixel group.

## **ERR subprogram: Error**

#### Format

CALL ERR (error- code, error-type[, error -severity, [line-number]])

#### Cross Reference

ON ERROR

#### Description

The ERR subprogram enables you to analyze the conditions that caused a



program error. ERR is normally called from a subroutine accessed by an ON ERROR statement.

The ERR subprogram returns the error-code and error-type, and optionally the error-severity and line-number, of the most recent program error.

An error is "cleared" when another program error occurs or when the program ends. A RETURN statement in a subroutine accessed by an ON ERROR statement also clears the error.

ON ERROR will not trap an error caused by the RUN command.

ERR returns a two-digit or three-digit number to the numeric variable error-code.

An error-code of 130 indicates an input/output (I/O) error.  
An error-code of 0 indicates that no error has occurred.  
The error-type is a numeric variable.

When an I/O error occurs, the value returned in error-type is the number (as assigned in an OPEN instruction) of the file in which the error occurred.

A negative error-type indicates that the error occurred during program execution.

An error-type of 0 indicates that no error has occurred.

#### Options

The value returned to the numeric variable error-severity is always nine.

The value returned to the numeric variable line-number is the line number of the program statement that was executing when the error occurred.

## **FILL subprogram**

#### Format

CALL FILL (pixel-row, pixel-column[, character-code] )

#### Cross Reference

CIRCLE, DCOLOR, DRAW, DRAWTO, GRAPHICS, POINT, RECTANGLE, WRITE

#### Description

The FILL subprogram enables you to fill in the area surrounding a specified pixel with a specified pattern and/or color.

Pixel-row and pixel-column are numeric expressions whose values specify the pixel that you want to surround with a color or pattern.

Character-code is a numeric expression with a value from 0-215 specifying the character with which to fill the area surrounding the specified pixel.

Pixel-row must have a value from 1 to 192, pixel-column must have a value



from 1 to 256.

If you do not specify a character-code, the default character is a solid square.

The color of the pattern that surrounds the specified pixel is the foreground color specified by the DCOLOR subprogram. If you have not called the DCOLOR subprogram, the default fill color is black.

The area surrounding the specified pixel is filled with the fill pattern until a screen edge or a foreground pixel (a pixel that is turned on) is encountered.

The boundaries of the area to be filled can be defined by lines drawn with the CIRCLE, DRAW, DRAWTO, POINT, RECTANGLE, WRITE subprogram.

FILL can be used only in High-Resolution Mode. An error results if you use FILL in Pattern or Text Mode.

In High-Resolution Mode the computer divides each pixel-row into 32 groups of 8 pixels each. The computer can assign a foreground color and a background color (from among the 16 available colors) to each 8-pixel group.

## **GCHAR subprogram: Get Character**

Format

Pattern and Text Modes

CALL GCHAR (row, column, numeric-variable)

High-Resolution Mode

CALL GCHAR (pixel-row, pixel column, numeric-variable)

Cross Reference

GRAPHICS, HCHAR, VCHAR

Description

The GCHAR subprogram enables you to ascertain the character code of a character on the screen or the status of a screen pixel.

The meaning of the value returned to the specified numeric-variable varies according to the graphics mode,

Pattern and Text Modes

Row and column are numeric expressions whose values specify a character position on the screen.

The value of row must be greater than or equal to 1 and less than or equal to 24.

The value of column must be greater than or equal to 1. In Pattern Mode, column must be less than or equal to 32; in Text Mode, column must be less



than or equal to 40.

GCHAR is not affected by margin settings. Row and column are relative to the upper-left corner of the screen, not to the corner of the window defined by the margins.

The character code of the character at the specified position is returned to the numeric-variable.

#### High-Resolution Mode

The pixel-row and pixel-column are numeric-expressions whose values specify a screen pixel position.

The value of the numeric expression pixel-row must be greater than or equal to 1. In High-Resolution Mode, pixel-row must be less than or equal to 192.

The value of the numeric expression pixel-column must be greater than or equal to 1 and less than or equal to 256.

The status of the specified screen pixel is indicated by the value returned to the numeric-variable. If the pixel is on, the value returned is 1; if the pixel is off, the value returned is 0.

## GRAPHICS subprogram

#### Format

CALL GRAPHICS (graphics mode)

#### Cross Reference

CHAR, CIRCLE, COLOR, DCOLOR, DRAW, DRAWTO, FILL, MARGINS, POINT, RECTANGLE, SCREEN, WRITE

#### Description

The GRAPHICS subprogram enables you to select the graphics-mode that offers you the combination of text and graphics capabilities that best suits the particular needs of your program.

Graphics-mode is a numeric expression whose value is from 1 to 3, specifying one of the three graphics modes available in MYARC Extended BASIC II.

#### NUMBER MODE

- 1 Pattern
- 2 Text
- 3 High-Resolution

When you enter MYARC Extended BASIC II, the computer is in Pattern Mode.

Whenever you use the CALL GRAPHICS subprogram, the computer does the following:



Clears the entire screen.

Restores the default character definitions of all characters.

Restores the default foreground color (black) and background color (transparent) to all characters.

Restores the default graphics foreground color (black) and background color (transparent).

Restores the default screen color (cyan).

Deletes all sprites.

Resets all sprites.

Resets the sprite magnification level to 1.

Restores the default screen margins (3, 30, 1, 24)

Restores the default current position (pixel-row 1, pixel-column 1).

Turns off all sound.

#### Pattern Mode

In Pattern Mode, the screen is considered to be a grid 24 characters high and 32 characters wide. Each character is 8 pixels high and 8 pixels wide. The 256 available characters are divided into 32 sets of 8 characters each. You can use the COLOR subprogram to assign a foreground and a background color from among the 16 available colors, to each character set.

In Pattern Mode, you have access to sprites.

The DCOLOR subprogram has no effect in Pattern Mode. If you use a CIRCLE, DRAW, DRAWTO, FILL, POINT, RECTANGLE, or WRITE subprogram, the error message GRAPHICS MODE ERROR IN (LINE NUMBER) is displayed.

#### Text Mode

In Text-Mode, the screen is considered to be a grid 24 characters high and 40 characters wide. Each character is 8 pixels high and 6 pixels wide. You can use the SCREEN subprogram to assign one foreground and one background color from among the 16 available colors. The colors you select are assigned to all 256 characters.

In Text Mode, you do not have access to sprites (the SPRITE subprogram has no effect in Text Mode).

Using the COLOR subprogram to assign colors to sprites has no effect.

The DCOLOR subprogram has no effect in Text Mode. If you use a CIRCLE, DRAW, DRAWTO, FILL, POINT, RECTANGLE, or WRITE subprogram, the error message GRAPHICS MODE ERROR IN (LINE NUMBER) is displayed.



## High Resolution Mode

In High-Resolution Mode, you have access to sprites, but not to sprite motion.

In High-Resolution Mode, the screen is considered to be a grid 192 pixels high and 256 pixels wide.

You can use the DCOLOR subprogram to assign colors to the graphics you display.

Use the COLOR subprogram only to assign colors to sprites; any other use of the COLOR subprogram causes an error.

In High-Resolution Mode, you have access to sprites.

ACCEPT, DISPLAY, and DISPLAY USING have no effect in High-Resolution Mode. INPUT, LINPUT, PRINT, and PRINT USING are functional only if they are used to access files. Use the command "WRITE" to display messages in High-Resolution Mode.

When a program running in High-Resolution Mode stops running, the computer returns to Pattern Mode.

## INIT subprogram: Initialize

Format  
CALL INIT

Cross Reference  
LINK, LOAD

Description  
The INIT subprogram initializes the 8K of memory allocated to assembly-language subprogram.  
If no CALL INIT is executed before CALL LINK or CALL LOAD then CALL INIT is assumed and executed.

## LINPUT Line Input

Format  
Keyboard Input  
LINPUT [input-prompt:]string-variable  
File Input  
LINPUT #file-number[,REC record-number]:string-variable

Cross Reference  
ACCEPT, EOF, INPUT, OPEN, TERMCHAR

Description



The LINPUT statement suspends program execution to enable you to enter a line of unedited data from the keyboard. LINPUT can be used also to retrieve an unedited record from an external device.

LINPUT assigns an entire line, a file record, or the remaining portion of a file record (if there is an input-pending condition) to the string-variable.

See INPUT for an explanation of keyboard- and file-input, and input options.

No editing is performed on the input data. All characters (including commas, quotation marks, colons, semicolons, and leading and trailing spaces) are assigned to the string-variable as they are encountered.

The maximum value that can be input from the keyboard is 255 characters.

LINPUT is frequently used instead of INPUT when the input data may include a comma. (A comma is not accepted as input by the INPUT statement, except as part of a string enclosed in quotation marks.)

To use LINPUT for file input the file must be in DISPLAY format.

You normally press ENTER to complete keyboard input; however, you can also use AID, BACK, BEGIN, CLEAR, PROC'D, DOWN ARROW, or UP ARROW. You can use the TERMCHAR function to determine which of these keys was pressed to exit from the previous ACCEPT, INPUT, or LINPUT instruction.

## LIST

Format

List to the screen

LIST [start-line-number , end-line-number]

List to a File (or Device)

LIST "file-specification" [: line-number-range]

Description

The LIST command displays the program (or a portion of it) currently in memory. You can also use LIST to output the program listing to an external device.

## LOAD subprogram

Format

File Only

CALL LOAD (file-specification-list)

Data Only

CALL LOAD (address, byte-list[, "", address, byte-list[, ...]])

File and Data

CALL LOAD (file-specification-list, address, byte-list[, ...])

CALL LOAD (address, byte-list, file-specification-list[, ...])



#### Cross Reference

INIT, LINK, PEEK, PEEKV, POKEV, VALHEX

#### Description

The LOAD subprogram enables you to load assembly-language subprograms into memory. You can also use LOAD to assign values directly to specified CPU (Central Processing Unit) memory addresses. You can use the POKEV subprogram to assign values to VDP (Video Display Processor) memory.

## MARGINS Subprogram

#### Format

CALL MARGINS (left, right, top, bottom)

#### Cross Reference

ACCEPT, CLEAR, DISPLAY, DISPLAY USING, GRAPHICS, INPUT, LINPUT, PRINT, PRINT USING, WRITE

#### Description

The MARGINS subprogram enables you to define screen margins. The margins you specify define a screen window that affects the operation of several instructions.

Left, right, top, and bottom are numeric expressions whose values specify the margins.

The margins cannot "overlap"; that is, the position of the top margin must be higher on the screen than the bottom margin, and the position of the left margin must be farther left on the screen than the right margin.

When creating a screen window, you must leave the window large enough to allow entry of a command.

The valid range for margin location varies according to the graphics mode. Acceptable values for the margins in each mode are as follows:

MODE TOP&BOTTOM LEFT&RIGHT

Pattern 1-24 1-32

Text 1-24 1-40

High-Resolution 1-24 1-32

The upper-left corner of the window defined by the margins is considered to be the intersection of row 1 and column 1 by the ACCEPT, DISPLAY, and DISPLAY USING instructions that use the AT option and the WRITE instruction.

The lower-left corner of the window is considered to be the beginning of the input line by the ACCEPT, INPUT, and LINPUT instructions.

The lower-left corner of the window is considered to be the beginning of



the print line by the DISPLAY, DISPLAY USING, PRINT, and PRINT USING instructions.

When the ACCEPT, INPUT, LINPUT, or PRINT USING instructions cause scrolling, scrolling occurs only in the window.

The CLEAR, GCHAR, HCHAR, and VCHAR subprogram are not affected by the margin settings.

In all Modes, the margins can extend to the edges of the screen.

When you enter MYARC Extended BASIC II, the left margin is set to 3 and the right margin to 30, The top and bottom margins are set to 1 and 24 respectively. When a program running in High-Resolution Mode ends, these default margin settings are restored.

## **PEEK subprogram: Peek at CPU RAM**

Format

```
CALL PEEK (address, numeric-variable-list  
[, "", address, numeric-variable-list[,...]])
```

Cross Reference

LOAD, PEEKV, POKEV, VALHEX

Description

The PEEK subprogram enables you to ascertain the contents of specified CPU memory addresses.

You can use the PEEKV subprogram to ascertain the contents of VDP memory.

## **PEEKV Subprogram Peek at VDP RAM**

Format

```
CALL PEEKV (address, numeric-variable-list  
[, "", address, numeric-variable-list[,...]])
```

Cross Reference

LOAD, PEEK, POKEV, VALHEX

Description

The PEEKV subprogram enables you to ascertain the contents of specified VDP memory addresses. You can use the PEEK subprogram to ascertain the contents of CPU memory.

The address is a numeric expression whose value specifies the first VDP (Video Display Processor) memory address at which you want to peek.

The address must have a value from 0 to 16383 inclusive.

If you know the hexadecimal value of the address (0000-3FFF), you can use the VALHEX function to convert it to a decimal numeric expression.



If necessary, the address is rounded to the nearest integer.

The numeric-variable-list consists of one or more numeric-variables separated by commas. Bytes of data starting from the specified VDP memory address are assigned sequentially to the numeric variables in the numeric-variable-list.

One byte, with a value from 0 to 255 inclusive, is returned to each specified numeric variable.

You can specify multiple addresses and numeric-variable-lists by entering a null string (two adjacent quotation marks) as a separator between a numeric-variable-list and the next address.

If you call the PEEKV subprogram with invalid parameters, the computer may function erratically. If this occurs, turn off the computer, wait several seconds, then turn the computer back on.

## **POINT Subprogram**

### Format

```
CALL POINT (pixel-type, pixel-row, pixel-column[,pixel-row2, pixel-  
column2[,...]])
```

### Cross Reference

CIRCLE, DCOLOR, DRAW, DRAWTO, FILL, GCHAR, GRAPHICS, RECTANGLE, WRITE

### Description

The POINT subprogram enables you to place, or erase specific points (pixels) on the screen, one or more at a time.

Pixel-type is a numeric-expression whose value specifies the action taken by the POINT subprogram.

### TYPE ACTION

2 Reverses the status of the specifies point (pixel). (If a pixel is on, it is turned off; if a pixel is off, it is turned on). This effectively reverses the color of the specified pixel.

1 Places a point, of the foreground color specified by the DCOLOR subprogram, at a specified pixel-row and pixel-column. This is accomplished by turning on the pixel at the designated row and column.

0 Erases a point at a specified pixel-row and pixel-column, This is accomplished by turning on the pixel at the designated row and column.

Pixel-row and pixel-column are numeric expressions whose values represent the screen position where the point will be placed (turned on or off).

You can optionally place more points by specifying additional sets of pixels.

Pixel-row must have a value from 1 to 192, pixel-column must have a value



from 1 to 256.

The last pixel-row/pixel-column you specify becomes the current position used by the DRAWTO subprogram.

POINT can only be used in High-Resolution Mode. An error results if you use POINT in Pattern or Text Modes.

## **POKEV Subprogram**

Format

```
CALL POKEV (address, byte-list[, "", address, byte-list[, ...]])
```

Cross Reference

LOAD, PEEK, PEEKV, VALHEX

Description

The POKEV subprogram enables you to assign values directly to specified VDP memory addresses.

You can use the LOAD subprogram to assign values to CPU.

## **REAL**

Format

```
REAL numeric-variable-list
```

```
REAL ALL
```

Cross Reference

DEF, DIM, DEFINT, OPTION BASE, SUB

Description

The REAL instruction enables you to declare the data type of specified numeric variables as REAL.

REAL variables have a greater range of values than do DEFINT variables and can contain decimal portions. You can use REAL as either a program statement or a command.

The numeric-variable-list consists of one or more numeric variables separated by commas. The variables are all assigned the REAL data type. A REAL statement with a numeric-variable-list must have a line number lower than any program reference to any variable in that list.

If you enter the ALL option, all numeric variables in your program are assigned the REAL data-type unless specifically declared as DEFINT. A REAL statement with the ALL option must have a line number lower than any program reference to any numeric variable or array.

When REAL ALL is used as a command, it does not affect any variables unless they follow it on a



multiple-statement line.

A REAL ALL statement in your main program does not affect the data type of a numeric variable in a subprogram.

A numeric variable of the REAL data-type can be of any number that can be expressed by the computer.

If you do not specify the data type of a numeric variable, it is assigned the REAL data-type (unless your program includes a DEFINT ALL statement or defines the specific variable as an integer).

REAL statements are evaluated during pre-scan, and are not executed.

You can also declare REAL variables by using the data-type option in the DEF, DIM, and SUB statements.

## **RECTANGLE subprogram**

### Format

```
CALL RECTANGLE (line-type, pixel-row1, pixel-column1,  
pixel-row2, pixel-column2, pixel-row3, pixel-column3[,...])
```

### Cross Reference

CIRCLE, DCOLOR, DRAW, DRAWTO, FILL, GRAPHICS, POINT, WRITE

### Description

The RECTANGLE subprogram enables you to place rectangles of various types and proportions on the screen,

Rectangles may be hollow (only the perimeter of the rectangle is drawn), or solid (both the perimeter and the entire area enclosed by the perimeter is drawn).

Line-type is a numeric-expression whose value specifies the action taken by the RECTANGLE subprogram.

### TYPE ACTION

5 Reverses the status of each pixel of the specified rectangle (solid). (If a pixel is on, it is turned off; if a pixel is off, it is turned on). This effectively reverses the color of the specified rectangle.

4 Draws arectangle (solid), of the foreground color specified by the DCOLOR subprogram. This is accomplished by turning on each pixel in the specified rectangle.

3 Erases a rectangle (solid), This is accomplished by turning off each pixel in the specified rectangle.

2 Reverses the status of each pixel in the perimeter of the specified rectangle. (If a pixel is on, it is turned off; if a pixel is off, it is



turned on). This effectively reverses the color of the perimeter.

1 Draws the perimeter of a rectangle, of the foreground color specified by the DCOLOR subprogram. This is accomplished by turning on each pixel in the specified rectangle.

0 Erases the perimeter of a rectangle. This is accomplished by turning off each pixel in the specified rectangle.

Pixel-row(#), and pixel-column(#), are numeric expressions whose values represent the screen positions of specific points of the rectangle. There are three points needed to define the rectangle, as shown below.

Pixel-row1 / pixel-column1 specify the TOP LEFT corner of the rectangle.

Pixel-row2 / pixel-column2 specify the TOP RIGHT corner of the rectangle.

Pixel-row3 / pixel-column3 specify the BOTTOM LEFT corner of the rectangle. All pixel-rows must have a value from 1 to 192. All pixel-columns must have a value from 1 to 256.

Note that the first pixel set (pixel-row1 and pixel-column1) represents the top leftmost point of the rectangle and must have a lower column value than the second pixel set. The second pixel set represents the top rightmost point of the rectangle. In the same manner, the third pixel set, which represents the bottom leftmost point of the rectangle, must have a higher row value than set1 or set2.

If the procedure outlined above is not followed, an error is issued.

You can optionally draw more rectangles by specifying additional sets of pixels. You must specify three sets of pixels for each rectangle.

The bottom-rightmost point of the last rectangle drawn becomes the current position used by the DRAWTO subprogram.

RECTANGLE can only be used in High-Resolution Mode. An error results if you use RECTANGLE in Pattern or Text Modes.

## SCREEN Subprogram

Format

CALL SCREEN ([foreground-color, ] background-color)

Cross Reference

COLOR, DCOLOR, GRAPHICS

Description

The SCREEN subprogram enables you to change the screen color. The screen color is the color of the border and the color displayed when transparent is specified as the foreground-color or background-color of a character or pixel.

In Text Mode, SCREEN enables you to change the color of the displayed



characters, as well as the color of the screen.

Background-color is a numeric expression whose value specifies a screen color from among the 16 available colors.

In Text Mode, foreground-color is a numeric expression whose value specifies a color from among the 16 available colors, representing the foreground-color of all 256 characters.

If you specify a foreground-color and the computer is not in Text Mode, it has no effect. If the computer is in Text Mode and you do not specify foreground-color, the foreground-color remains unchanged.

When you enter MYARC Extended BASIC II, the background-color is cyan and the foreground-color is black. When your program ends (either normally or because of an error), stops at a breakpoint, or changes graphics mode, the default colors are restored.

## **TERMCHAR function   Termination Character**

Format  
TERMCHAR  
Type  
DEFINT

Cross Reference  
ACCEPT, INPUT, LINPUT

Description  
The TERMCHAR function returns the character code of the key pressed to exit from the previously executed INPUT, ACCEPT, or LINPUT statement.

In a program, the value returned by TERMCHAR depends on the key pressed to exit from the last instruction that accepted input from the keyboard.

VALUE RETURNED KEY

1 AID  
2 CLEAR  
10 DOWN ARROW  
11 UP ARROW  
12 PROC'D  
13 ENTER  
14 BEGIN  
15 BACK

If you use TERMCHAR as part of a command (unless it is preceded by ACCEPT, INPUT, or LINPUT), the value returned depends on the key pressed to enter the command (ENTER, UP ARROW, or DOWN ARROW).

Note that pressing CLEAR during keyboard input normally causes a break in the program. However, if your program includes an ON BREAK NEXT statement, you can use CLEAR to exit from an input field.



## VALHEX function   Value of Hexadecimal Number

### Format

VALHEX (string-expression)

### Type

DEFINT

### Description

VALHEX returns the numeric value of the hexadecimal number represented by the string-expression.

The string-expression specifies the hexadecimal (base 16) number to be converted to a decimal (base 10) number. If you use a string constant, it must be enclosed in quotation marks.

The string-expression must contain only valid hexadecimal digits (0-9,A-F). Alphabetic hexadecimal digits must be upper-case letters. VALHEX can convert a hexadecimal number from one to four digits long. If the length of the string-expression is greater than four, VALHEX uses only the last four characters.

VALHEX returns an integer greater than or equal to -32768 (hexadecimal 8000) and less than or equal to 32767 (hexadecimal 7FFF).

## VCHAR subprogram   Vertical Character

### Format

CALL VCHAR(row, column, character-code[, number-of-repetitions])

### Cross Reference

DCOLOR, GCHAR, GRAPHICS, HCHAR

### Description

The VCHAR subprogram enables you to place a character on the screen and repeat it horizontally.

Row and column are numeric expressions whose values specify the position on the screen where the character is displayed.

The value of row must be greater than or equal to 1, row must be less than or equal to 24.

The value of column must be greater than or equal to 1. In Pattern or High-Resolution Mode, the column must be less than or equal to 32; in Text Mode, column must less than or equal to 40.

VCHAR is not affected by margin settings.

Character-code is a numeric expression with a value from 0-255, specifying



the number of the character.  
See Appendix B for a list of ASCII character codes.

The optional number-of-repetitions is a numeric expression whose value specifies the number of times the character is repeated horizontally. If the repetitions extend past the end of a column, they continue from the first character of the next column. If the repetitions extend past the end of the last column, they continue from the first character of the first column.

If you use VCHAR to display a character on the screen, and then later use CHAR, COLOR, or DCOLOR to change the appearance of that character, the result depends on the graphics mode:

In pattern and Text Modes, the displayed character changes to the newly specified pattern and/or color(s).

In High-Resolution Mode, the displayed character remains unchanged.

## VERSION subprogram

Format  
CALL VERSION (numeric-variable)

Description  
The VERSION subprogram returns a value indicating the version of BASIC being used.

In MYARC Extended BASIC II, VERSION returns a value of 200 to the numeric-variable you specify.

## WRITE subprogram

Format  
CALL WRITE (type, row, column, string-expression  
[, row2, column2, string-expression2[,...]])  
Cross Reference  
GRAPHICS, HCHAR, MARGIN, VCHAR

Description  
The WRITE subprogram enables you to display strings on the screen in High-Resolution Mode. The string-expression may be a constant or a variable.

Type is a numeric-expression whose value specifies the action taken by the WRITE subprogram.

TYPE ACTION  
0,1 Displays the string-expression horizontally within the screen boundaries. Margins are disregarded. Display begins at the specified row and column.



2 Displays the string-expression vertically within the screen boundaries. Margins are disregarded. Display begins at the specified row and column.

If the string to be displayed will not fit within the screen, the string will wrap around on the screen.

For WRITE, the screen is considered to be 24 rows by 32 columns, As in HCHAR and VCHAR, blocks of pixels 8x8 are the unit of measurement not single pixels as in most other High-Resolution subprogram.

Row must have a value from 1 to 24, column must have a value from 1 to 32.

WRITE can only be used in High-Resolution Mode. An error results if you use WRITE in Pattern or Text Modes.