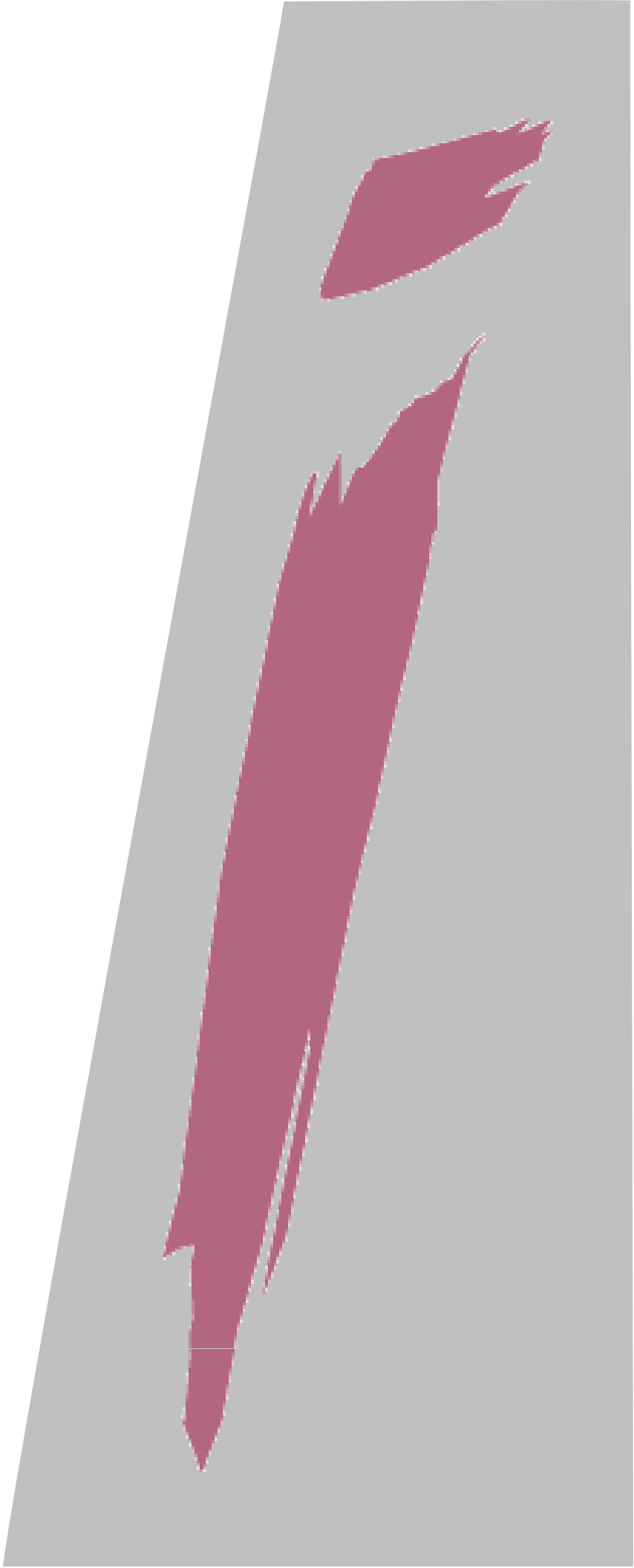


Reference Guide

Version 6.3

Copyright Notice

The OptioDCS product is undergoing continual revision, refinement, and expansion in order to include additional features. This documentation is believed to be accurate and reflects the product at the time of publication.

However, no responsibility is assumed by Optio for its use; nor for any infringements of patents or other rights of third parties which may result from its use. Optio reserves the right to change this product at any time without notice.

This manual reflects OptioDCS for Windows NT and UNIX version 6.3 software.

Trademarks

Published By

Optio Software, Inc.
3015 Windward Plaza
Windward Fairways II
Alpharetta, Georgia 30005
Tel: +1.770.576.3500
Fax: +1.770.576.3699

Contents

Chapter 1

DCL Language Reference

DCL, or Document Customization Language, is the set of commands used to create OptioDCS documents. This chapter explains the basic concepts of the language, and contains an alphabetical reference for each keyword. It also contains a list of valid commands for each module type.

Introduction to DCL

DCL, or Document Customization Language, is the set of commands used to create OptioDCS documents. DCL is a free-format language, so a command statement may start in any column, and may extend across several lines if necessary because of length or desired for clarity. When using DCL to create documents, you can define and name your own variables and processing modules, and add (non-processing) comments wherever desired.

Keyword Types

Each DCL keyword may be a command, clause, or function. Some keywords are used in more than one way. A command, unlike a clause or function, stands on its own. Examples of commands are LET, DRAW LINE, and SET SEQUENCE.

A clause or function cannot stand on its own, and must be used as part of a command. Each clause can be used only with certain commands. Examples of clauses are THICKNESS and ALIGN. There are also header clauses, such as FEEDER, which are used with the commands that start modules. Functions are most often used within the LET command, which assigns the value it returns to a variable, but they may be used within other commands as well.

Commands, clauses, and functions must always be in all upper case.

Variables

Names

A variable name may be comprised of any combination of numbers, letters, periods (.), at signs (@), or underbars (_), but it cannot begin with a number or period. A variable name may be up to 255 characters long.

Variable names may be in all lower case or mixed cases. In order to avoid conflict with commands, clauses, or functions, do not use all upper case characters for variable names.

Special variables

There are several variable names that are reserved for certain special variables. The variable **@** is the name used for the standard input buffer. This buffer holds the current page of input data. **@** is an array of strings, and parts of it may be addressed as any other array of strings (see the "Arrays" section below). This variable is most often used in data map modules, region modules, and input channel modules.

There are other special variables for use in data map modules and region modules. Their values indicate certain lines on the current input data page by line number, as listed below.

Variable	Indicates
#	the current scan line
?	the starting line of the current region
!	the line above the starting line of the current region
$	the difference between **!** and **#**, which is the offset between the line above the starting line of the region and the current scan line

Data Types

A variable may contain one of the following three types of data:

- Boolean values - a value of either TRUE or FALSE.
  The DCL keywords for these values are as follows:
  TRUE - TRUE, YES, or On
  FALSE - FALSE, NO, or Off

- numeric values - integer or floating point values.
  A numeric value is expressed using digits, an optional period as a decimal point, and an optional plus sign or hyphen for a positive or negative number.

- character strings - sequences of ASCII characters.
  A character string must be enclosed in single or double quotation marks.

The data type of a variable is determined by the type of value that is assigned to it. If a variable is used without being assigned a value, it is assumed to have the value FALSE, 0.0, or an empty string. If a variable is used in a situation where a variable of a different data type is expected, the variable is automatically converted to the appropriate data type, as follows:

- a Boolean value of TRUE converted to numeric becomes 1.0; converted to string becomes "TRUE"
- a Boolean value of FALSE converted to numeric becomes 0.0; converted to string becomes "FALSE"
- a numeric value of 0.0 converted to Boolean becomes FALSE; converted to string becomes "0.0"
- a numeric value that is not 0.0 converted to Boolean becomes TRUE; converted to string becomes the floating point value represented as a string
- a string converted to Boolean becomes TRUE if it is "TRUE", "YES", "T", "Y", "1", or "ON"; otherwise it becomes FALSE
- a string converted to numeric becomes 0.0 if not recognizable as a number; otherwise it is converted to the value represented by the digits, up to the first character that is not a digit, plus sign (+), hyphen (-), period (.), or space. For example, "54,321.7" becomes 54.0.

Arrays

Any variable may be an ordered array of values, and any part of the array may be addressed using an index enclosed in square brackets, following the array name, as shown below.

arrayname[*index*]

A range of elements may also be addressed using a colon (:), as shown below.

arrayname[*startindex*:*endindex*]

To address a range from one element to the end of the array, use a 0 as the ending index. To address all elements of an array, omit the index.

For arrays of strings, substrings may be addressed by using a two-dimensional index with a comma, as shown below.

arrayname[*stringindex*,*startindex*:*endindex*]

A range of substrings may also be addressed, as shown below.

arrayname[*startstring*:*endstring*,*startindex*:*endindex*]

To address substrings of all elements of an array, use a 0 as the string index.

For example, the expressions listed below represent the indicated parts of the array **phonenums**.

Expression	Represents
`phonenums[3]`	the third element
`phonenums[3:5]`	the third, fourth, and fifth elements
`phonenums`	all elements
`phonenums[3:0]`	all elements except the first and second
`phonenums[5,1:3]`	the first three characters of the fifth element
`phonenums[4:6,1:3]`	the first three characters of the fourth, fifth, and sixth elements
`phonenums[0,1:3]`	the first three characters of all elements

These principles can be applied to addressing data in the standard input buffer, as listed below.

To address:	Use the format:
a single character	`@[`*`row`*`,`*`column`*`]`
a range of lines	`@[`*`startrow`*`:`*`endrow`*`]`
a string	`@[`*`row`*`,`*`startcol`*`:`*`endcol`*`]`
a column of data	`@[`*`startrow`*`:`*`endrow`*`,`*`startcol`*`:`*`endcol`*`]`

When assigning values to an array, you can use an index for each element, or use the special symbols << and >> to assign multiple element values at once. For example, the following statements assign values to three elements of the array **names**.

```
LET names[1] = "Jack"
LET names[2] = "Jill"
LET names[3] = "Jan"
```

The following statement assigns these values to the same elements.

```
LET names = <<"Jack", "Jill", "Jan">>
```

You can also use this syntax to define a literal array, without assigning the values to an actual array name. For example, the following two statements print the same text.

```
DRAW TEXT AT 2,3 USING names
DRAW TEXT AT 2,3 USING <<"Jack", "Jill", "Jan">>
```

To assign a value to the first empty element of an array, use a 0 as the index. For example, the following statements assign the values Jan, Dot, and Liz, in that order, to the array **names**.

```
LET names[2] = Dot
LET names[0] = Jan
LET names[0] = Liz
```

Worksheets

A worksheet is a work area where variables and their values are stored temporarily. You may use worksheet commands to create worksheets, place variables on them, and manipulate the worksheet stack. However, for most documents this is not necessary, because OptioDCS provides default worksheets and manages them automatically.

As each module of a document is processed, a worksheet is created to hold the variables used in the module. Most worksheets are placed on top of the worksheet stack, which is the ordered set of worksheets that are searched for the value of a variable when the variable is used. The stack is searched from top to bottom, and the value of the first occurrence of the variable is used.

There are four types of worksheets, as follows:

- A global worksheet is created for the entire document, and placed on the stack. You use the global worksheet to hold any variables that should exist and maintain their values, regardless of which part module is processing. The name of the global worksheet is **globals**.

- A public worksheet is created and placed on the stack for each part module as it begins processing. When the part is finished processing, its public worksheet is removed from the stack, and any unprotected variables are deleted; when the document is finished processing, the public worksheets are deleted. A public worksheet has the same name as the corresponding part.

- An auto worksheet is created and placed on the stack for each data map, region, format, segment, or function module as it begins processing. When the module is finished processing, its auto worksheet is deleted. If a module is called recursively, an auto worksheet is created and placed on the stack for each call. Auto worksheets are not named.

- A private worksheet is also created for each data map, region, format, segment, or function module as it begins processing. The private worksheet is not placed on the stack, but the private worksheet of the current module is searched before the stack. The private worksheets are not deleted until the document is finished processing. If a module is called recursively, the same private worksheet is used for all calls. Private worksheets are not named.

If a variable is used without being assigned a value, it is assumed to have the value FALSE, 0.0, or an empty string, depending on the expected data type for the situation. If a new variable is assigned a value using the LET command, it is placed on the public worksheet for the current part (or the global worksheet, if no part is being processed). You may change this default to another worksheet using the SET WORKSHEET command. You may also create variables and place them on specific worksheets, using

the commands GLOBAL, PUBLIC, PRIVATE, and AUTO. The first time these commands are executed, they assign initial values to the variables; the AUTO command reassigns the initial value each time it is executed.

You may delete variables using the DESTROY command; you may delete all unprotected variables from a worksheet using the CLEAN command.

All unprotected variables are deleted from a public worksheet when the corresponding part is finished processing. Variables created using the PUBLIC command are protected, and those created using the LET command are unprotected. You may change the status of a variable using the PROTECT and UNPROTECT commands.

You may create a worksheet using the CREATE WORKSHEET command. User-created worksheets are not placed on the stack, and are not deleted until the document is finished processing. You may place a user-created worksheet on the stack using the SEARCH WORKSHEET command. You may replace the global worksheet or the public worksheet for the current part using the SET GLOBAL or SET PUBLIC command.

You may delete a worksheet using the DESTROY WORKSHEET command.

You may specify a variable on a specific worksheet using the syntax

```
sheetname::varname
```

where *sheetname* is the name of the worksheet and *varname* is the name of the variable.

Expressions

An expression is a combination of values, variables, comparisons, and operations that can be evaluated to produce a specific value. An expression may include literal numeric values, literal Boolean values, literal strings, variables (including arrays and array elements), mathematical calculations, functions, and comparisons.

Within DCL, an expression can be used in place of a literal value within a command statement. For example, the first statement below uses a literal string to assign a value to the variable **greeting**, and the second uses an expression.

```
LET greeting = "Dear Mr. Brown,"
LET greeting = "Dear" & name & ","
```

As another example, the first statement below uses literal numeric values to place a line, and the second uses expressions.

```
DRAW LINE AT 1,1 TO 2,2
DRAW LINE AT y3,x3 TO a+b-3,4*c
```

The only exceptions to this are that you must use a literal string when naming modules, and you may not substitute an expression for the variable **@**.

An expression may include any of the following values:

- literal numeric values
- literal character strings
- literal Boolean values
- variables, including arrays and array elements

An expression may also involve any of the following:

- DCL functions or user-defined functions
- comparisons between values or expressions
- mathematical operations
- string operations

Comparisons

An expression may involve any of the following comparisons:

- `>` for greater than
- `<` for less than
- `=` for equal to
- `<>` for not equal to
- `>=` for greater than or equal to
- `<=` for less than or equal to

Comparisons evaluate to Boolean values. Comparisons may be combined using the following operations:

- AND - combines two comparisons, evaluates to TRUE if both comparisons evaluate to TRUE. For example:

  ```
  (a > b) AND (c = d)
  ```

  evaluates to TRUE if **a** is greater than **b** and **c** is equal to **d**; evaluates to FALSE if **a** is not greater than **b** or if **c** is not equal to **d**.

- OR - combines two comparisons, evaluates to TRUE if either comparison evaluates to TRUE. For example:

  ```
  (a > b) OR (c = d)
  ```

  evaluates to TRUE if **a** is greater than **b** or if **c** is equal to **d**; evaluates to FALSE if **a** is not greater than **b** and **c** is not equal to **d**.

- NOT - reverses a comparison, evaluates to TRUE if the comparison evaluates to FALSE. For example:

  ```
  NOT (c = d)
  ```

  evaluates to TRUE if **c** is not equal to **d**; evaluates to FALSE if **c** is equal to **d**.

Mathematical Operations

An expression may involve mathematical calculations, using any of the following operations:

- + for addition
- - for subtraction and negation
- * for multiplication
- / for division
- ^ for exponentiation
- % for modulus, producing the remainder of integer division

Parentheses may also be used to affect the order of calculation.

String Operations

An expression may involve concatenating literal strings and variables containing strings, using an ampersand (&). For example, the expression below combines two literal strings and the variable **name**.

```
"Dear" & name & ","
```

An expression may contain parentheses that group portions of string expressions. Parentheses indicate the order of string operations, just as they do for mathematical operations.

Special String Characters

When using expressions to test for matching strings (such as in IF, FIND, and SWITCH statements) you may use special characters in the expressions to match multiple strings.

To match one of several choices, separate the choices using a vertical bar. For example, the expression "(4 | 5 | 6) times" matches each of the strings "4 times", "5 times", and "6 times". The expression "((trans)|(inter))action" matches the strings "transaction" and "interaction".

To match one from a set of characters, use brackets to designate the set. For example, the expression "Site [XQZ]" matches "Site X", "Site Q", and "Site Z", but not "Site D".

You may also use a hyphen to specify a range of characters in a set. For example, "level [A-M]" matches "level A", "level B", and so on, through "level M". You may also specify only the characters that are not in the set, by placing a caret at the beginning of the set. For example, the expression "Site [^XQZ]" matches "Site D", "Site ?", "Site 4", etc., but not "Site X".

To match any character, use a period. For example, the expression "type . page .." matches "type B page 34", "type 2 page 01", "type % page !x", etc.

To match one of more occurences of a certain character, place a plus sign after the character. To match zero or more occurences, place an asterisk after the character. To match zero or one occurrence (an optional character), place a question mark after the character. The following table shows examples of expressions using these features.

Expression	Matching Strings			
"0+75"		"075"	"0075"	"0000075"
"0*75"	"75"	"075"	"0075"	"0000075"
"0?75"	"75"	"075"		

To match any string, use the expression ".*".

To match a string only when it occurs at the beginning of a line, place a caret at the beginning of the expression, for example, "^Page 3". To match a string only when it occurs at the end of a line, place a dollar sign at the end of the expression, for example, "balance$".

To match any of the special characters used in expressions, place a backslash before the character. For example, the expression "Code *" matches the string "Code *". A list of the special characters that require this notation follows.

Expression	Matching Character
"*"	asterisk
"\\"	backslash
"\{"	opening brace
"\["	opening bracket
"\^"	caret
"\$"	dollar sign
"\("	opening parenthesis
"\)"	closing parenthesis
"\."	period
"\+"	plus sign
"\?"	question mark
"|"	vertical bar

Modules

The commands that compose a document are grouped into processing modules. Each document must have a document module to control the processing of the document. A document may have multiple parts, such as a customer copy, an accounting copy, and a shipping copy. For each part, the document should have a part module containing any specifications that differ from those of the entire document. If your document has multiple parts, the document module specifies the order in which the parts should be processed.

Commands in the document and part modules also indicate the input and output channels, select the output device, and call the data map and format modules. The input channel module specifies how the input data is received, and the data map module contains instructions on how to scan and processes the input data. The format module indicates how the data should be to formatted and merged with a form. The processed document is sent through output channels to printers or other applications, as specified in the output channel module and device module.

Comments

Unstructured comments may be added wherever desired in DCL files; they do not affect processing. To create a multiline comment, enclose the comment in curly braces ({}). To create a single-line comment, place two slashes (//) to the left of the comment. A single-line comment may be on a line by itself, or it may be to the right of a command to clause. See the sample documents for examples.

Structured comments are used to state the name and contents of a file; see the structured comments at the top of the sample documents for examples. Structured comments begin with two exclamation marks (!!).

Valid Commands by Module Type

Commands Valid Outside Modules

The following commands are valid for use outside of any module:

CHANNEL
DATAMAP
DEVICE
DOCUMENT
FONTPACK
FORMAT
FUNCTION
PALETTE
PART
REGION
SEGMENT
SQL
TRANSLATE

Document Module Commands

All DCL functions are valid for use in a document module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a document module:

ALTER CHANNEL
AUTO
BREAK ON
CALL
CLEAN
CLOSE
COMMIT WORK
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
DRAW
END
EXIT
FAX
FAX RESET
FETCH
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
MAIL
MAIL RESET
MAP
OPEN
PRIVATE
PROCESS
PROCESS PARTS
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
RETURN
REWIND
ROLLBACK WORK
RUN
SEARCH WORKSHEET
SET COLLATE
SET COPIES
SET DEVICE
SET GLOBAL
SET INPUT
SET LOG FILE
SET OUTPUT
SET PUBLIC
SET SEQUENCE
SET SQL ACCESSKEY
SET SQL AUTOROLLBACK
SET SQL DATABASE
SET SQL TRANSACTIONS
SET SQL VENDOR
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SQLRUN
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Part Module Commands

All DCL functions are valid for use in a part module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a part module:

ALTER CHANNEL
AUTO
BREAK ON
CALL
CLEAN
CLOSE
COMMIT WORK
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
DRAW
END
EXIT
FAX
FAX RESET
FETCH
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
MAIL
MAIL RESET
MAP
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
RETURN
REWIND
ROLLBACK WORK
RUN
SEARCH WORKSHEET
SET DEVICE
SET GLOBAL
SET INPUT
SET LOG FILE
SET OUTPUT
SET PUBLIC
SET SQL ACCESSKEY
SET SQL AUTOROLLBACK
SET SQL DATABASE
SET SQL TRANSACTIONS
SET SQL VENDOR
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SQLRUN
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Function Module Commands

All DCL functions are valid for use in a function module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a function module:

AUTO
CALL
CLEAN
CLOSE
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
DRAW
END
EXIT
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
MAP
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
RETURN
REWIND
RUN
SEARCH WORKSHEET
SET
SET GLOBAL
SET LOG FILE
SET PUBLIC
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SQLERROR
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Data Map Module Commands

All DCL functions are valid for use in a data map module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a data map module:

ADVANCE LINE
AUTO
CALL
CLEAN
CLOSE
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
END
EXIT
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
MAP
NEXT LINE
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
REPEAT LINE
RETURN
REWIND
RUN
SEARCH WORKSHEET
SET GLOBAL
SET LOG FILE
SET PUBLIC
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Region Module Commands

All DCL functions are valid for use in a region module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a region module:

ADVANCE LINE
AUTO
CALL
CLEAN
CLOSE
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
END
EXIT
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
MAP
NEXT LINE
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
REPEAT LINE
RETURN
REWIND
RUN
SEARCH WORKSHEET
SET GLOBAL
SET LOG FILE
SET PUBLIC
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Format Module Commands

All DCL functions are valid for use in a format module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a format module:

AUTO
CALL
CLEAN
CLOSE
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
DRAW
DRAW ARC
DRAW BARCODE
DRAW BOX
DRAW CIRCLE
DRAW COMB
DRAW CURVE
DRAW ELLIPSE
DRAW IMAGE
DRAW LINE
DRAW OBJECT
DRAW TEXT
END
EXIT
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
RESET MACRO
RETURN
REWIND
RUN
SEARCH WORKSHEET
SET
SET GLOBAL
SET LOG FILE
SET MACRO
SET PUBLIC
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Segment Module Commands

All DCL functions are valid for use in a segment module. For a complete list of DCL functions, see page 1-20.

The following commands are valid within a segment module:

AUTO
CALL
CLEAN
CLOSE
CREATE WORKSHEET
DESTROY
DESTROY WORKSHEET
DRAW
DRAW ARC
DRAW BARCODE
DRAW BOX
DRAW CIRCLE
DRAW COMB
DRAW CURVE
DRAW ELLIPSE
DRAW IMAGE
DRAW LINE
DRAW OBJECT
DRAW TEXT
END
EXIT
FLUSH
FOR
GLOBAL
IF
IGNORE WORKSHEET
LET
LOAD
LOG
OPEN
PRIVATE
PROTECT
PUBLIC
READ
READ PAGE
REDIRECT
RESET MACRO
RETURN
REWIND
RUN
SEARCH WORKSHEET
SET
SET GLOBAL
SET LOG FILE
SET MACRO
SET PUBLIC
SET TRACE FILE
SET TRACE MASK
SET WARNING FILE
SET WARNING MASK
SET WORKSHEET
SWITCH
TRACE
UNPROTECT
WARNING
WHILE
WRITE

Channel Module Commands

The following commands are valid within a channel module:

END

Device Module Commands

The following commands are valid within a device module:

ALIAS
END
FEEDER
FONT
FONTPACK
PAPER
STACKER

Fontpack Module Commands

The following commands are valid within a fontpack module:

ALIAS
END
FONT

Palette Module Commands

The following commands are valid within a palette module:

COLOR
END

SQL Module Commands

The following commands are valid within an SQL module:

END

Translate Module Commands

The following commands are valid within a TRANSLATE module:

END

Functions

Following is a list of the DCL functions. Functions are valid within document, part, function, data map, region, format, and segment modules.

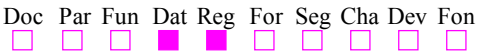
ABS
ACOS
ASIN
ATAN
AVG
BREAK
CANON
CLIP
COLUMNS
COS
CPUNAME
DATE
EMPTY
EOF
ESCAPE
EXPORT
FATALFILE
FIND
FIX
FLATTEN
FROMHEX
HOSTNAME
IMPORT
ISALNUM
ISALPHA
ISBLANK
ISDATE
ISDIGITS
ISMONEY
ISNUMERIC
LENGTH
LOGFILE
LOOKUP
MATCHES
MAX
MIN
NOW
NUM
OSCPUID
OSNAME
PAGE
PAGES
PARAMETER
PATCH
PICTURE
PLATFORM
PRUNE
RELEASE
ROUND
ROWS
ROWS2
SERIAL
SIN
SORT
SPAN
SQLERRMSG
SQLERROR
SQUISH
SUM
SYSTEM
TAN
TIME
TOBOOLEAN
TODOLLARS
TOHEX
TOLOWER
TONUMBER
TOSTRING
TOUPPER
TOWORDS
TRACEFILE
TRACEMASK
TRACENAMES
TRIM
UNCANON
USERNAME
VERSION
VISIBLE
VISIBLEX
WARNINGFILE
WARNINGMASK
WARNINGNAMES
WHERE

Reference

This section defines each keyword of DCL.

Reference Format

The commands, clauses, and functions of DCL are listed alphabetically in this reference. At the top of each command entry, the valid modules are indicated by the key shown below.



The shaded boxes of this key indicate which modules are valid for the use of the command. For example, the key above indicates that the command may be used in a data map module or a region module. The key abbreviations indicate modules types as listed below.

Abbrev.	Module	Abbrev.	Module
Doc	document	For	format
Par	part	Seg	segment
Fun	function	Cha	channel
Dat	data map	Dev	device
Reg	region	Fon	fontpack

Each keyword is defined using the format described below. Any categories that do not apply to a keyword are omitted from its reference entry. If a keyword is used in more than one way, the reference includes a numbered entry for each mode of use.

Type: the type of keyword: command, clause, or function; special variables are also defined.

Purpose: a description of what the command, clause, or function does, and how it is used.

Affects: which types of objects are affected when the keyword is used.

Syntax: the syntax for using the command, clause, or function; see the "Syntax Notation" section below for more information.

Default: the default value for the command or clause.

Example: an example of how the command, clause or function is used.

Parameters: definitions of each parameter associated with the command, clause, or function.

See also: a list of related reference entries.

Syntax Notation

The following notational conventions are used in syntax statements:

- Parameters are printed in italics, indicating that a literal value, variable, or expression must be provided when using the command, clause, or function.
- Optional parts of a syntax statement are enclosed in brackets. For example, the statement below indicates that both IF BLANK and IF NOT BLANK are valid.

  ```
  IF [NOT] BLANK
  ```

  Do not enter brackets as part of a statement, except for array indices (or addressing data on the input data page).
- Choices of clauses or keywords are separated by vertical bars. For example, the statement below indicates that any of CHOCOLATE or VANILLA or STRAWBERRY are valid, and if VANILLA is used, SPRINKLES may also be included. Because the choices are enclosed in square brackets, they are optional and may be omitted.

  ```
  [CHOCOLATE | VANILLA [SPRINKLES] | STRAWBERRY]
  ```

  Do not enter a bar as part of a statement.
- Required choices of clauses or keywords are separated by vertical bars and enclosed in curly braces. For example, the statement below indicates that any of CHOCOLATE or VANILLA or STRAWBERRY are valid, but you are required to use one of them.

  ```
  {CHOCOLATE | VANILLA | STRAWBERRY}
  ```

  Do not enter curly braces as part of a statement, except to enclose multiline comments.
- Default options are underlined. For example, the statement below indicates that the default option is VANILLA.

  ```
  [CHOCOLATE | VANILLA | STRAWBERRY]
  ```

- Possible series of parameters or clauses are indicated by ellipses (...). For example, the statement below indicates that the RETURNS clause may be followed by a single variable or a series of variables.

  ```
  RETURNS retval1 [,retval2 ...]
  ```

@

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ ■ ■ □ □ ■ □ □

Type: Special variable

Purpose: Is the name of the standard input buffer; holds the current page of input data as an array of strings.

Syntax: `@`

Example: `LET quantity[$] = @[#,1:5]`

See also: BUFFER

$

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ ■ ■ □ □ □ □ □

Type: Special variable

Purpose: Has the value of the offset between the line above the starting line of the region and the current scan line

Syntax: $

See also: REGION, ?, !, #

!

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ ■ ■ □ □ □ □ □

Type: Special variable

Purpose: Has the value of the line above the starting line of the current region.

Syntax: !

See also: REGION, ?, $, #

#

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ ■ ■ □ □ □ □ □

Type: Special variable

Purpose: Has the value of the current scan line.

Syntax: #

See also: REGION, ?, !, $

?

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ ■ ■ □ □ □ □ □

Type: Special variable

Purpose: Has the value of the starting line of the current region.

Syntax: ?

See also: REGION, !, $, #

ABS

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Calculates the absolute value of a value.

Syntax: `ABS(`*`number`*`)`

Parameters: *number*— a constant, variable, or expression, that evalutates to a numerical value

Examples:
```
LET difference = ABS(difference)

IF ABS(x *c 0.25 - y) = 5 THEN
```

See also: AVG, ROUND, SUM

ACCEPTS

This clause is used with the commands below. See the entries for those commands.

See: DATAMAP, DOCUMENT, FORMAT, FUNCTION, PART, REGION, SEGMENT

ACCESSKEY

1

This clause is used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL, FAX, MAIL, SQL

2

This keyword is part of the command below. See the entry for that command.

See: SET SQL ACCESSKEY

ACOS

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Calculates the arc cosine of a number, returns an angle in radians.

Syntax: `ACOS(`*`number`*`)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value, ranging from -1 to +1

Examples:
```
LET angle1 = ACOS(difference)

LET angle2 = ACOS((values[1] - values[2]) * 3)
```

See also: COS, ASIN, ATAN

ADVANCE

This clause is the same as the clause SPACING. See page 1-166.

See also: ADVANCE LINE

ADVANCE LINE

Doc Par Fun Dat Reg For Seg Cha Dev Fon
☐ ☐ ☐ ■ ■ ☐ ☐ ☐ ☐ ☐

Type: Command

Purpose: Moves the current scan line down one line; region or datamap processing continues with the next command. The current scan line is a marker which indicates a certain line in the input data; it is used to refer to locations of data on the input page. When the current scan line is moved outside the region (or datamap), the module is exited.

Syntax: `ADVANCE LINE`

See also: NEXT LINE, REPEAT LINE

ADVANCE ROW

This command is the same as the command ADVANCE LINE. See page 1-26.

ALIAS

This command is the same as the command FONT. See page 1-88.

ALIGN

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the alignment of an object within its bounding box.

Affects: bar codes, images, and text

Syntax: `ALIGN "alignment"`

Default: "Left" for text; "Top Left" for other objects

Parameters: *alignment*—a keyword indicating the alignment of the object within the bounding box, as listed in the table below.

Alignment	Keywords				
default	D	Default			
top left	TL	Top Left	TopLeft		
top center	TC	Top Center	TopCenter		
top right	TR	Top Right	TopRight		
middle left	ML	Middle Left	MiddleLeft		
middle center	MC	Middle Center	MiddleCenter		
middle right	MR	Middle Right	MiddleRight		
bottom left	BL	Bottom Left	BottomLeft		
bottom center	BC	Bottom Center	BottomCenter		
bottom right	BR	Bottom Right	BottomRight		
left, on baseline	L	Text Left	TextLeft	Left	XL
center, on baseline	C	Text Center	TextCenter	Center	XC
right, on baseline	R	Text Right	TextRight	Right	XR

See also: DRAW BARCODE, DRAW IMAGE, DRAW TEXT, SET

ALTER CHANNEL

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ □ □ □ □ □ □ □ □

Type: Command

Purpose: Changes the attributes of an existing channel module temporarily. The changes only affect the document or part module containing the ALTER CHANNEL command. For more information on channels, see the entry for the CHANNEL command on page 1-38.

Syntax: **For an Input Channel**

```
ALTER CHANNEL channame
[MODE "READ"] [FILE filename | COMMAND unixcommand | CGI readcgi]
[ROWS maxrows] [COLUMNS maxcolumns]
[BUFFER buffername] [FORMFEEDS usefeeds]
[RETURNS usereturns] [NEWLINES usenewlines]
[CONTROLS readcontrols] [TABS readtabs] [SEARCHABLE searchon]
[TRANSLATE tablename] [FIELDS field1, field2 [, field3…]
[SEPARATOR sepchar] [DELIMITER delchar]]
```

For an Output Channel

```
ALTER CHANNEL channame
[MODE "channelmode"]
[FILE filename [SMART smarton]|COMMAND unixcommand [SMART smarton]
| PRINTER printername | [SERVER servername] [PORT portnum]
[[ACCESSKEY keyname] | [USERNAME faxuser PASSWORD faxpass]]]
[RETURNS usereturns] [NEWLINES usenewlines]
[FORMFEEDS usefeeds] [ROWS maxrows] [FIELDS field1, field2 [, field3…]
[SEPARATOR sepchar] [DELIMITER delchar]]
```

Parameters: *channelname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the channel module to be altered.

channelmode—a keyword representing the operational mode of the channel, as listed in the table below. The mode involves reading from or writing to a target object, which may be a file or printer.

Mode	Keyword
read data from a target object; fail if the object does not exist	READ
write data to a target file; if it does not exist, create it	WRITE
append data to an existing target object; if it does not exist, create it	APPEND
write data to an existing target object; fail if the object does not exist	REWRITE
create a new target object and append data to it; fail if it already exists	CREATE

filename—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of a file for the channel. An input channel reads data from the file; an output channel writes data to the file.

smarton—a Boolean expression that indicates whether a channel can accept other information in addition to documents. This is used to send FAX information through a channel to fax products other than OptioFAX. The default is FALSE. *This parameter is valid for UNIX systems only.*

unixcommand—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a UNIX command to issue. This is used to change the source of data for an input channel, or the UNIX destination of an output channel. For example, an input channel might issue a grep or cat command, and an output channel might issue an lp command. *This parameter is valid for UNIX systems only.*

readcgi—a Boolean expression that indicates whether the input is a CGI data stream so that OptioDCS will automatically create variables. The default is FALSE. For more information, see Chapter 4 of the *Advanced Features Guide*.

maxrows—a numeric expression indicating the maximum number of rows (or records) of data allowed per page. For an input channel reading from a delimited text file, this specifies how many records to read at one time (if there are fewer records in the file, the extra variables are set to NUL). For an output channel, this only applies when using the ASCII text driver or writing to a delimited text file.

maxcolumns—a numeric expression indicating the maximum number of columns allowed per page of input data. Any extra columns of data are discarded.

buffername—a name for the buffer variable that holds the data for the input channel. The default is @.

usefeeds—a Boolean expression. For an input channel, indicates whether to recognize any formfeeds in the input data. If this is off, the formfeeds are ignored. For an output channel, indicates whether to output a formfeed at the end of each format (when using the ASCII text driver). The default is FALSE.

usereturns—a Boolean expression. For an input channel, indicates whether to recognize any carriage returns in the input data. If this is off, the carriage returns are ignored. For an output channel, indicates whether to output a carriage return at the end of each line (when using the ASCII text driver).

usenewlines—a Boolean expression. For an input channel, indicates whether to recognize any newlines in the input data. If this is off, the newlines are ignored. For an output channel, indicates whether to output a newline at the end of each line (when using the ASCII text driver). The default is TRUE.

readcontrols—a Boolean expression that indicates whether to recognize any ANSI control codes in the input data. If this is off, ANSI control codes are ignored. The default is FALSE.

readtabs—a Boolean expression that indicates whether to recognize any tabs in the input data. In OptioDCS, tabs are positioned eight spaces apart. If this is off, the tabs are ignored. The default is FALSE.

searchon—a Boolean expression that indicates whether to copy the input data to a temporary file and use that file as input. This provides random access, so that you can use commands such as READ PAGE and REWIND. The temporary file is destroyed when the channel is closed. This setting is useful when reading data from a data stream such as **stdin**, a pipeline, or a socket; it does not affect data read from a file. The default is FALSE. *This parameter is valid for UNIX systems only.*

tablename—the name of the translate module to use to translate characters in the input data before it is processed. For more information on data translation tables, see Chapter 4 of the *Advanced Features Guide*.

printername—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the UNC name of a shared Windows NT printer for an output channel. *This parameter is valid for Windows NT systems only.*

servername—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the the SMTP mail server used to send E-mail messages, or the remote OptioFAX API server used to send faxes. If the channel uses sockets, this clause is required.

portnum—a constant, variable, or expression, that evaluates to a numerical value, indicating the port on the server (or socket number) used for E-mail or faxing. If the channel uses sockets, this clause is required. The default value is 1 for a socket channel, 25 for an SMTP (E-mail) channel, and 3000 for a fax channel.

keyname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key used by the client system to log on to the secure application server (OptioFAX, MAPI e-mail, or SQL database). For more information on access keys, see Chapter 2 of the *User's Guide*.

faxuser, *faxpass*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the user name and password used by the client system to log into the OptioFAX API server. The default values are "optio".

For UNIX systems, if the OptioFAX security database has been implemented, the user name for the security database should be specified.

For NT systems:

- If OptioFAX is running on an NT server, use the NT user account name prefaced with its domain name (i.e., USERNAME "*domain\username*"). The user name must be valid to log in locally to both the OptioFAX server and the OptioDCS server.
- If OptioFAX is running on a UNIX server, the user name must be valid both in OptioFAX security and on the NT server.

sepchar—a string containing a single character that is used to separate fields in a delimited text file. The default value is a comma (,). To specify a tab as the separator character, use SEPARATOR TABS.

delchar—a string containing a single character that is placed on each end of a text string in a delimited text file. The default value is a quotation mark (“).

field1, *field2*, *field3*—variable names for the fields in a delimited text file. For an input channel, the field data is assigned to these variables in the order listed. For an output channel, the values of these variables are written to the file in the order listed.

Examples:

```
ALTER CHANNEL "stdin" ROWS 60 COLUMNS 128

ALTER CHANNEL "acctg"
   SERVER "maildrop.com" PORT 25
END CHANNEL
```

See also: CHANNEL, FAX, MAIL, PAGES, READ PAGE, REDIRECT, REWIND, SET INPUT, SET OUTPUT, SYSTEM, TRANSLATE

ALTER INPUT

This command is the same as the command ALTER CHANNEL. See page 1-27.

ALTER OUTPUT

This command is the same as the command ALTER CHANNEL. See page 1-27.

ALTERNATE

This clause is used with the command below. See the entry for that command.

See: DRAW COMB

ASIN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Calculates the arc sine of a number, returns an angle in radians.

Syntax: `ASIN(number)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value, ranging from -1 to +1

Examples:
```
LET angle1 = ASIN(difference)

LET angle2 = ASIN((values[1] - values[2]) * 3)
```

See also: SIN, ACOS, ATAN

AT

This clause is used with the commands listed below. See the entries for those commands.

See: DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT

ATAN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Calculates the arc tangent of a number, returns an angle in radians.

Syntax: `ATAN(number)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
LET angle1 = ATAN(difference)

LET angle2 = ATAN((values[1] - values[2]) * 3)
```

See also: TAN, ACOS, ASIN

ATTACH

This clause is used with the command below. See the entry for that command.

See: MAIL

AUTO

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Creates variables in the automatic worksheet for the current module.

Syntax: `AUTO variable1 = initvalue1 [, variable2 = initvalue2 ...]`

Parameters: *variable1*, *variable2*—names for the new variables. An AUTO statement may create any number of variables.

initvalue1, *initvalue2*—expressions indicating initial values for the variables, which are assigned to the variables each time the module is executed.

See also: CLEAN WORKSHEET, DESTROY, GLOBAL, PRIVATE, PUBLIC

AVG

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Calculates the average of the values in an array.

Syntax: `AVG(array)`

Parameters: *array*— the name of an array variable, which contains numerical values

Example:
```
LET avgorder = AVG(quantities)
```

See also: ABS, MAX, MIN, ROWS, SUM

BARS

This clause is used with the command below. See the entry for that command.

See: DRAW COMB

BINDING

This clause is used with the command below. See the entry for that command.

See: FORMAT

BORDER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Determines whether a rectangular border is drawn around the bounding box of an object.

Affects: bar codes, combs, images, objects, and text

Syntax: `BORDER` *`borderon`*

Default: Off

Parameters: *borderon*—a Boolean expression that indicates whether to draw a border around the bounding box.

See also: DRAW BARCODE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, SET, THICKNESS

BREAK

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if a new job is starting on a channel; returns FALSE if not. A new job starts when a BREAK ON command is executed.

Syntax: `BREAK(channelstring)`

Parameters: *channelstring*— a literal string or variable containing the name of a channel (predefined channels are **stdin**, **stdout**, **stderr**, **smtp**, and **faxfx**).

Examples:
```
IF BREAK("stdout") THEN

IF BREAK(accntg) THEN
```

See also: EMPTY, EOF

BREAK ON

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Causes an output job break whenever the value of any of the specified variables change. Usually, this command should be placed after the MAP command that calls the datamap and before any DRAW commands that call formats.

Syntax: `BREAK ON var1 [,var2...]`

Examples:
```
BREAK ON faxnumber, acctnumber

BREAK ON emailaddress
```

Parameters: *var1*, *var2*—the variables that should cause a job break when one of their values changes. Any number of variables may be specified.

See also: BREAK

BUFFER

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, DATAMAP, REGION

CALL

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Calls a function module, passing in any parameters needed and receiving any return values. A function module uses DCL commands to perform a specific function, accepting parameters and returning values.

Syntax:
```
CALL functionname [WHEN condition]
[WITH param1 [,param2...]] [RETURNING retval1 [,retval2...]]
```

Parameters: *functionname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the function module; if the module is in a separate file, include the path of the file.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed. The WHEN clause must be before any WITH or RETURNING clauses.

param1, *param2*—expressions to be passed into the function as parameter values. The number of WITH values should match the number of parameters the function accepts, and they should be in the same order as defined by the ACCEPTS clause of the FUNCTION statement.

retval1, *retval2*—variable names for the return values that are received from the function. The number of RETURNING values should match the number of values the function returns, and they should be in the same order as defined by the RETURN statement within the function module.

See also: FUNCTION, RETURN, RUN

CANON

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Detects any escape code character sequences in a string. Returns the same string with each escape code character sequence converted to a special character. This allows you to send an escape code to your printer when printing documents, because you cannot place an escape code directly into an .fgl file. Escape code character sequences begin with a carat (^) or backslash (\).

Syntax: `CANON(`*`string`*`)`

Parameters: *string*— a literal string, a variable containing a string, or an expression that forms a string

Examples:
```
LET escapecode = CANON("\E")

LET mystring = CANON(mystring)
```

See also: UNCANON, VISIBLE, VISIBLEX

CASE

This clause is used with the command below. See the entry for that command.

See: SWITCH

CGI

This clause is used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

CHANNEL

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Module command

Purpose: Defines a channel module. A channel module specifies a pipe or path through which OptioDCS either receives input data or sends a processed document to the output device.

To create channels for	See the *Advanced Features Guide*
CGI data	Chapter 4
Delimited files	Chapter 4
E-mailing	Chapter 3
Faxing	Chapter 3

Syntax:

For an Input Channel

```
CHANNEL channelname [LIKE existchn]
[MODE "READ"] [FILE filename | COMMAND unixcommand | CGI readcgi]
[ROWS maxrows] [COLUMNS maxcolumns]
[BUFFER buffername] [FORMFEEDS usefeeds]
[RETURNS usereturns] [NEWLINES usenewlines]
[CONTROLS readcontrols] [TABS readtabs] [SEARCHABLE searchon]
[TRANSLATE tablename] [FIELDS field1, field2 [, field3…]
[SEPARATOR sepchar] [DELIMITER delchar]]
```

For an Output Channel

```
CHANNEL channelname [LIKE existchn] [MODE "channelmode"]
[FILE filename [SMART smarton]|COMMAND unixcommand [SMART smarton]
| PRINTER printername | [SERVER servername] [PORT portnum]
[[ACCESSKEY keyname] | [USERNAME faxuser PASSWORD faxpass]]]
[RETURNS usereturns] [NEWLINES usenewlines]
[FORMFEEDS usefeeds] [ROWS maxrows] [FIELDS field1, field2 [, field3…]
[SEPARATOR sepchar] [DELIMITER delchar]]
```

Parameters: *channelname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a name for the input or output channel. Do not use the name of a predefined standard channel, as listed below.

existchn—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of an existing channel or one of the predefined standard channels, as listed below. The LIKE clause copies all of the attributes of the existing channel to the new channel.

standard input:	stdin
standard output:	stdout

standard error output: stderr
SMTP e-mail output: smtp
MAPI e-mail output: mapi (for Windows NT only)
OptioFAX output: faxfx

channelmode—a keyword representing the operational mode of the channel, as listed in the table below. The mode involves reading from or writing to a target object, which may be a file or printer.

Mode	Keyword
read data from a target object; fail if the object does not exist	READ
write data to a target file; if it does not exist, create it	WRITE
append data to an existing target object; if it does not exist, create it	APPEND
write data to an existing target object; fail if the object does not exist	REWRITE
create a new target object and append data to it; fail if it already exists	CREATE

filename—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of a file for the channel. An input channel reads data from the file; an output channel writes data to the file.

smarton—a Boolean expression that indicates whether a channel can accept other information in addition to documents. This is used to send FAX information through a channel to fax products other than OptioFAX. The default is FALSE. *This parameter is valid for UNIX systems only.*

unixcommand—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a UNIX command to issue. This is used to change the source of data for an input channel, or the UNIX destination of an output channel. For example, an input channel might issue a grep or cat command, and an output channel might issue an lp command. *This parameter is valid for UNIX systems only.*

readcgi—a Boolean expression that indicates whether the input is a CGI data stream so that OptioDCS will automatically create variables. The default is FALSE. For more information, see Chapter 4 of the *Advanced Features Guide*.

maxrows—a numeric expression indicating the maximum number of rows (or records) of data allowed per page. For an input channel reading from a delimited text file, this specifies how many records to read at one time (if there are fewer records in the file,

the extra variables are set to NUL). For an output channel, this applies only when using the ASCII text driver or writing to a delimited text file. The default value is 66.

maxcolumns—a numeric expression indicating the maximum number of columns allowed per page of input data. Any extra columns of data are discarded. The default value is 132. The maximum value is 8,195.

buffername—a name for the buffer variable that holds the data for the input channel. The default is @.

usefeeds—a Boolean expression. For an input channel, indicates whether to recognize any formfeeds in the input data. If this is off, the formfeeds are ignored. For an output channel, indicates whether to output a formfeed at the end of each format (when using the ASCII text driver). The default is FALSE.

usereturns—a Boolean expression. For an input channel, indicates whether to recognize any carriage returns in the input data. If this is off, the carriage returns are ignored. For an output channel, indicates whether to output a carriage return at the end of each line (when using the ASCII text driver).

usenewlines—a Boolean expression. For an input channel, indicates whether to recognize any newlines in the input data. If this is off, the newlines are ignored. For an output channel, indicates whether to output a newline at the end of each line (when using the ASCII text driver). The default is TRUE.

readcontrols—a Boolean expression that indicates whether to recognize any ANSI control codes in the input data. If this is off, ANSI control codes are ignored. The default is FALSE.

readtabs—a Boolean expression that indicates whether to recognize any tabs in the input data. In OptioDCS, tabs are positioned eight spaces apart. If this is off, the tabs are ignored. The default is FALSE.

searchon—a Boolean expression that indicates whether to copy the input data to a temporary file and use that file as input. This provides random access, so that you can use commands such as READ PAGE and REWIND. The temporary file is destroyed when the channel is closed. This setting is useful when reading data from a data stream such as **stdin**, a pipeline, or a socket; it does not affect data read from a file. The default is FALSE. *This parameter is valid for UNIX systems only.*

tablename—the name of the translate module to use to translate characters in the input data before it is processed. For more information on data translation tables, see Chapter 4 of the *Advanced Features Guide*.

printername—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the UNC name of a shared Windows NT printer for an output channel. *This parameter is valid for Windows NT systems only.*

servername—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the the SMTP mail server used to send e-mail messages, or the remote OptioFAX API server used to send faxes. If the channel uses sockets, this clause is required.

portnum—a constant, variable, or expression, that evaluates to a numerical value, indicating the port on the server (or socket number) used for e-mail or faxing. If the channel uses sockets, this clause is required. The default value is 1 for a socket channel, 25 for an SMTP (e-mail) channel, and 3000 for a fax channel.

keyname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key used by the client system to log on to the secure application server (OptioFAX, MAPI e-mail, or SQL database). For more information on access keys, see Chapter 2 of the *User's Guide*.

faxuser, *faxpass*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the user name and password used by the client system to log into the OptioFAX API server. The default values are "optio".

For UNIX systems, if the OptioFAX security database has been implemented, the user name for the security database should be specified.

For NT systems:

- If OptioFAX is running on an NT server, use the NT user account name prefaced with its domain name (i.e., USERNAME "*domain\username*"). The user name must be valid to log in locally to both the OptioFAX server and the OptioDCS server.
- If OptioFAX is running on a UNIX server, the user name must be valid both in OptioFAX security and on the NT server.

sepchar—a string containing a single character that is used to separate fields in a delimited text file. The default value is a comma (,). To specify a tab as the separator character, use SEPARATOR TABS.

delchar—a string containing a single character that is placed on each end of a text string in a delimited text file. The default value is a quotation mark (").

field1, *field2*, *field3*—variable names for the fields in a delimited text file. For an input channel, the field data is assigned to these variables in the order listed. For an output channel, the values of these variables are written to the file in the order listed.

Examples:

```
CHANNEL "myinput" LIKE "stdin"
   ROWS 66
   COLUMNS 128
```

```
END CHANNEL

CHANNEL "acctfile" LIKE "stdout"
   FILE "\accting\acctrecs"
END CHANNEL
```

Note: The channel module must end with the END CHANNEL command.

See also: ALTER CHANNEL, END, FAX, FLUSH, MAIL, PAGES, READ PAGE, REDIRECT, REWIND, SET INPUT, SET OUTPUT, TRANSLATE

CLEAN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Deletes all unprotected variables from a global, public, or user-created worksheet. CLEAN GLOBAL WORKSHEET cleans the current global worksheet, CLEAN PRIVATE WORKSHEET cleans the private worksheet for the current module, CLEAN PUBLIC WORKSHEET cleans the public worksheet for the current part module.

Note: The CLEAN PUBLIC WORKSHEET command is executed automatically each time the processing completes on a part module.

Syntax: `CLEAN {GLOBAL WORKSHEET | PRIVATE WORKSHEET | PUBLIC WORKSHEET | WORKSHEET "`*`sheetname`*`"}`

Parameters: *sheetname*—the name of a global, public, or user-created worksheet. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

See also: CREATE WORKSHEET, DESTROY, DESTROY WORKSHEET, PROTECT, SET GLOBAL, SET PUBLIC

CLIP

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Removes any blanks from the beginning of a string. If the string parameter is an array, any blanks are removed from the beginning of each element of the array.

Syntax: CLIP(*string*)

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET amount = CLIP(amount)

LET prices[1:5] = CLIP(prices[1:5])

LET namestring = CLIP(lastname & "," & firstname)
```

See also: PRUNE, TRIM, COLUMNS, LENGTH

CLOSE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Closes an open channel. CLOSE INPUT closes the current input channel, CLOSE OUTPUT closes the current output channel, or a channel or SQL module can be specified.

An input channel is opened by the MAP command, an output channel is opened by the DRAW command, and any channel may be opened explicitly using the OPEN command. All channels are closed automatically when document processing is complete. The CLOSE command can be used to separate output jobs by closing the output channel between DRAW commands.

Syntax: CLOSE {INPUT | OUTPUT | "*channelname*"}

Parameters: *channelname*—the name of a channel or SQL module.

See also: CHANNEL, FETCH, OPEN, SET INPUT, SET OUTPUT, SQL, SQLRUN

COLLATE

This clause is used with the command below. See the entry for that command.

See: SET COLLATE

COLOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Defines a color within a palette module. For more information, see Chapter 2 of the *Advanced Features Guide*.

Syntax: `COLOR colorname, Rvalue, Gvalue, Bvalue`

Parameters: *colorname*— a literal string or an expression that evaluates to a string, containing a name for the color.

Rvalue, Gvalue, Bvalue— numerical value or expressions that evaluate to numerical values, indicating the red, green, and blue light values used to produce the color. For RGB type palette modules. Valid values range from 0 to 255. Black is defined as 0, 0, 0; white is defined as 255, 255, 255.

See also: DEVICE, PALETTE

COLUMNS

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

1

Type: Function

Purpose: Counts the columns of a string or an array. If the string parameter is an array, returns the length of the longest element of the array.

Syntax: `COLUMNS(string)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:

```
LET number = COLUMNS("How many characters is this?")

IF COLUMNS(greeting) > limit THEN

LET colwidth = 0.25 * COLUMNS(itemdesc[1:numitems])

LET xcoord = 3 + COLUMNS("Company: " & coname & " of " city)
```

See also: LENGTH, ROWS

2

This clause is used with the command below. See the entry for that command.

See: CHANNEL

COMMAND

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, FONT

COMMIT WORK

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Commits to the SQL database any changes made by SQL modules that have not yet been committed. When SQL transaction control is off, each change made by an SQL module is automatically committed to the database.

Syntax: `COMMIT WORK`

See also: ROLLBACK WORK, SET SQL TRANSACTIONS, SQL

CONTROLS

This clause is used with the command below. See the entry for that command.

See: CHANNEL

COPIES

This clause is used with the command below. See the entry for that command.

See: SET COPIES

COS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Calculates the cosine of an angle in radians.

Syntax: `COS(angle)`

Parameters: *angle*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
IF COS(firstangle) < 0.4 THEN

LET answer = COS((ang1 - ang2) * 3)
```

See also: ACOS, SIN, TAN

CPUNAME

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the name of the CPU.

Syntax: `CPUNAME()`

Examples:
```
LET cpu = CPUNAME()

LET string = "CPU: " & CPUNAME()
```

See also: HOSTNAME, OSCPUID, OSNAME, PLATFORM, USERNAME

CREATE WORKSHEET

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Creates a new worksheet. A worksheet is a work area where variables and their values are stored temporarily. A user-created worksheet behaves like a global worksheet, but is not placed on the worksheet stack.

Syntax: `CREATE WORKSHEET "sheetname"`

Parameters: *sheetname*—a name for the worksheet.

See also: CLEAN, DESTROY WORKSHEET, SEARCH WORKSHEET, SET WORKSHEET

DATABASE

1

This clause is used with the command below. See the entry for that command.

See: SQL

2

This keyword is part of the command below. See the entry for that command.

See: SET SQL DATABASE

DATAMAP

Doc □ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Module command

Purpose: Defines a datamap module. A data map module specifies to OptioDCS how to read the input data and assign it to variables.

Syntax: `DATAMAP "mapname" [LINES maxlines | BUFFER inputbuf] [ACCEPTS param1 [,param2 ...]]`

Parameters: *mapname*—a name for the data map module.

maxlines—a numeric expression indicating the maximum number of lines allowed per page of input data.

inputbuf—the input buffer; this sets the maximum number of lines allowed per page of input data to the length of the input buffer. The default input buffer is @.

param1, *param2*—variable names for the parameter values that are passed into the data map. The values passed in are assigned to these variables, which are used within the data map module. A data map may have any number of parameters.

Note: The data map module must end with the END DATAMAP command.

See also: END, MAP

DATE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the system date as a string, in the form *mm*/*dd*/*yyyy*, where *mm* is a number indicating the month, *dd* is a number indicating the day, and *yyyy* is the year.

Syntax: `DATE()`

Examples:
```
LET printdate = DATE()

LET string = "Date processed: " & DATE()
```

See also: NOW, SHOWDATE, TIME

DELIMITER

This clause is used with the command below. See the entry for that command.

See: CHANNEL

DENSITY

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the width of the narrow bars of a bar code.

Affects: bar codes

Syntax: `DENSITY` *`density`*

Default: the default density for the device

Parameters: *density*—the width of the narrow bars in the current units. See Chapter 2 or 3 in this book for the valid densities for each bar code type for each printer driver. This value is ignored for bar code types which do not vary in size.

See also: DRAW BARCODE, RATIO, SET, UNITS

DESTROY

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Deletes the specified variables.

Syntax: `DESTROY [`*`worksheet1`*`::]`*`variable1`*
`[, [`*`worksheet2`*`::]`*`variable2`* `...]`

Parameters: *worksheet1*, *worksheet2*—the names of the worksheets containing the variables. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

variable1, *variable2*—the names of the variables to be removed. If no worksheet is specified, the first occurrence of the variable is deleted; any private worksheet is searched first, then the worksheets on the stack are searched in order.

See also: AUTO, DESTROY WORKSHEET, GLOBAL, PRIVATE, PUBLIC

DESTROY WORKSHEET

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Deletes a global, public, or user-created worksheet.

Syntax: `DESTROY WORKSHEET "sheetname"`

Parameters: *sheetname*—the name of a global, public, or user-created worksheet. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

See also: CREATE WORKSHEET, CLEAN WORKSHEET, DESTROY, SET GLOBAL, SET PUBLIC

DEVICE

Doc □ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Module command

Purpose: Defines a device module. A device module specifies the attributes of a type of output device, such as a printer type or a fax application.

For information on	See
printer macros	the PCL section of Chapter 2
producing PDF output	the PDF section of Chapter 2
producing RTF output	the RTF section of Chapter 2
using label printers	Chapter 3
GDI devices	the GDI section of Chapter 2 (for Windows NT only)

Syntax:
```
DEVICE "devicename" UNITS unittype [LIKE "existdevice"]
[LANGUAGE "devlanguage"] [RESOLUTION vertres,horzres]
[DUPLEX "duplexon"] [MODEL printdriver] [PALETTE "palname"]
[MACROS {"macroson" | rangemin:rangemax}]
```

Parameters: *devicename*—a name for the device.

unittype—a keyword indicating the units of measurement for the paper size definitions in the device module.

existdevice—the name of an existing device. The LIKE clause copies all of the specifications in the existing device to the new device.

devicelanguage—a keyword for the page description language used by the device, as listed in the table below.

Language	Keyword
HP-PCL 4	PCL4
HP-PCL 5	PCL5
HP-PCL 5 GL2	PCL5GL2
PostScript level 1	PS1
PostScript level 2	PS2
ASCII text	TEXT, TXT
Graphics Device Interface	GDI
PDF	PDF
RTF (Rich Text Format)	RTF
Intermec	ICL
Monarch	MCL
Zebra	ZPL

vertres—the maximum vertical resolution for the device, in dots per inch.

horzres—the maximum horizontal resolution for the device, in dots per inch.

duplextype—a Boolean expression that indicates whether the device supports duplex printing. The default is "Off".

printdriver—the path and name of the printer driver that this device uses to print. This is for GDI devices on Windows NT systems only.

palname—the name of a color palette module. This makes the colors defined in the palette available for use in the document. This is for RTF, PDF, and GDI devices, and PostScript devices that support color output.

macroson—a Boolean expression that indicates whether the device supports the use of macros. This enables macros for PCL devices that support macros. The default is FALSE.

rangemin, *rangemax*—constants, variables, or expressions that evaluate to numerical values, indicating the range of memory slots that are available in printer memory for macros. Using this implies that the device supports macros. This enables macros for PCL devices that support macros.

Note: The device module must end with the END DEVICE command.

See also: END, PALETTE, SET DEVICE, SET MACRO, UNITS

DIRECTION

This clause is used with the command below. See the entry for that command.

See: DRAW COMB

DOCUMENT

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ □ □ □ □ □

Type: Module command

Purpose: Defines a document module. A document module controls the processing of a document.

Syntax: `DOCUMENT "docname" [ACCEPTS param1 [,param2 ...]]`
`[EXTENSION "PARAMETER=value"]`

Parameters: *docname*—a name for the document module.

param1, *param2*—variable names for the parameter values that are passed into the document. The values passed in are assigned to these variables, which are used within the document module. A document may have any number of parameters.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

Note: The document module must end with the END DOCUMENT command.

See also: END, RETURN, RUN

DRAW

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ □ □ ■ ■ □ □ □

Type: Command

Purpose: Calls a format module or segment module for processing. A format module or segment module specifies how to format the data, and what other text and graphics to place on the document.

Syntax: DRAW "*formatname*" [WHEN *condition*]
[WITH *param1* [,*param2*...]] [RETURNING *retval1* [,*retval2*...]]

Parameters: *formatname*—the name of the format module or segment module; if the module is in a separate file, include the path of the file.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed. The WHEN clause must be before any WITH or RETURNING clauses.

param1, *param2*—expressions to be passed into the format or segment as parameter values. The number of WITH values should match the number of parameters the module accepts, and they should be in the same order as defined by the ACCEPTS clause of the FORMAT or SEGMENT statement.

retval1, *retval2*—variable names for the return values that are received from the format or segment. The number of RETURNING values should match the number of values the module returns, and they should be in the same order as defined by the RETURN statement within the format or segment module.

See also: FORMAT, SEGMENT

DRAW ARC

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Graphics command

Purpose: Draws an arc at the specified position. For more information, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

Syntax: DRAW ARC PIVOT *pivot_v*,*pivot_h* {AT *at_v*,*at_h* ROTATION *rotangle* |
RADIUS *radius* START *startangle* STOP *stopangle*}
[TYPE "SIMPLE"|"CHORD"|"PIE"] [LCOLOR *linecolor*]
[THICKNESS *thickvalue*] [LINESTYLE "*stylename*"] [LINECAP "*capname*"]
[PATTERN "*patname*" [FCOLOR *fillpatcolor*]] [OPAQUE *opaqueon*]
[SHADOW *offset*,["*shadpat*"] [,"*corner*"] [SCOLOR *shadpatcolor*]]
[WHEN *condition*] [MACRO "*macroname*"]

Parameters: *pivot_v*—the vertical measurement of the pivot point (an arc is a section of a circle; this is the center of the circle), in the current units.

pivot_h—the horizontal measurement of the pivot point (an arc is a section of a circle; this is the center of the circle), in the current units.

at_v—the vertical measurement where the arc should start, in the current units. The distance between this point and the pivot point is the radius.

at_h—the horizontal measurement where the arc should start, in the current units.

rotangle—the counter-clockwise angle of rotation from the starting point, in degrees (e.g., ROTATION 90 draws a quarter of a circle).

radius—the radius in the current units. The default radius can be set using the SET RADIUS command.

startangle—the angle where the arc should start, counter-clockwise from the horizontal to the right of the pivot point, in degrees (e.g., START 90 is directly over the pivot point).

stopangle—the angle where the arc should end, counter-clockwise from the horizontal to the right of the pivot point, in degrees (e.g., STOP 270 is directly below the pivot point).

TYPE—a keyword indicating the type of arc; "SIMPLE" (the default) draws only the arc, "CHORD" draws the arc and a line connecting the endpoints, "PIE" draws the arc and lines connecting the endpoints to the pivot point.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the arc outline. The default line color can be set using the SET LCOLOR command.

thickvalue—a value specifying the thickness of the line in the current units. The default thickness may be set using the SET THICKNESS command.

stylename—a keyword indicating a line style. The default line style may be set using the SET LINESTYLE command.

capname—a keyword indicating a line cap. This applies only to simple arcs. The default line cap may be set using the SET LINECAP command.

patname—a keyword indicating a fill pattern. This applies only to chord and pie arcs. The default pattern may be set using the SET PATTERN command.

fillpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the fill pattern. The default fill pattern color can be set using the SET FCOLOR command.

opaqueon—a Boolean expression that indicates whether the arc should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

offset—the offset between the arc and the shadow, in the current units. This applies only to chord and pie arcs. An arc with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow. This applies only to chord and pie arcs. The default pattern may be set using the SET PATTERN command.

corner—a keyword indicating the direction of the shadow. This applies only to chord and pie arcs.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the arc is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

See also: FCOLOR, LCOLOR, INECAP, LINESTYLE, OPAQUE, PATTERN, RADIUS, SCOLOR, SHADOW, THICKNESS, UNITS

DRAW BARCODE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a bar code at the specified position

Syntax:
```
DRAW BARCODE AT start_v,start_h USING data [TYPE "barcodetype"]
[STYLE "style"] [TO end_v,end_h | [HEIGHT height] [WIDTH boxwidth]]
[DENSITY density] [RATIO ratio] [ROTATION angle] [ALIGN "alignment"]
[BORDER borderon [THICKNESS thickvalue]] [LCOLOR linecolor]
[OPAQUE opaqueon] [WHEN condition] [MACRO "macroname"]
[EXTENSION "PARAMETER=value" [LINK "url"]]
```

Parameters: *start_v*—the vertical measurement where the top left corner of the bar code should print, in the current units.

start_h—the horizontal measurement where the top left corner of the bar code should print, in the current units.

data—the data to print as a bar code; a literal string, variable, or expression.

barcodetype—a keyword indicating the bar code type. See Chapter 2 or 3 in this book for a list of bar code types for your printer driver. The default type may be set using the SET TYPE command.

style—a keyword indicating the bar code style (a variation within the type). See Chapter 2 or 3 in this book for a list of styles for each bar code type for your printer driver. The default style may be set using the SET STYLE command.

end_v—the vertical measurement where the bar code bounding box should end, in the current units. The TO point defines a bounding box, used to size the bar code, to align the bar code, or to draw a border. The height of the bounding box is the height of the bar code.

end_h—the horizontal measurement where the bar code bounding box should end, in the current units.

height—the height of the bar code and its bounding box, in the current units. This value is ignored for bar code types which do not vary in size. The default height may be set using the SET HEIGHT command.

boxwidth—the width of the bar code bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box, used to align the bar code or to draw a border. The default width may be set using the SET WIDTH command.

density—the width of the narrow bars in the current units. See Chapter 2 or 3 in this book for the valid densities for each bar code type for your printer driver. This value is ignored for bar code types which do not vary in size. The default density may be set using the SET DENSITY command.

ratio—the width of the wide bar, in terms of the narrow bar width. Valid values are 2.0 (for a 2.0 to 1 ratio), 2.5 , and 3.0. This value is ignored for bar code types which do not vary in size. The default ratio may be set using the SET RATIO command.

angle—the counter-clockwise angle of rotation (a bar code rotates about the AT point). Valid values are 0, 90, 180, and 270. The default angle may be set using the SET ROTATION command.

alignment—a keyword indicating the alignment of the bar code within the bounding box; the default is top left. The default alignment may be set using the SET ALIGN command.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box. The default border setting may be set using the SET BORDER command.

thickvalue—a value specifying the thickness of the border in the current units. The default thickness may be set using the SET THICKNESS command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the arc outline. The default line color can be set using the SET LCOLOR command.

opaqueon—a Boolean expression that indicates whether the bar code should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the bar code is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

url—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a destination URL. This creates a hyperlink within the text bounding box for PDF output. For more information, see Chapter 4 of the *Advanced Features Guide*.

See also: Bar Code References in Chapters 2 and 3, ALIGN, BORDER, HEIGHT, LCOLOR, OPAQUE, ROTATION, THICKNESS, UNITS, WIDTH

DRAW BOX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a rectangle at the specified position

Syntax:
```
DRAW BOX AT start_v,start_h
{TO end_v,end_h | HEIGHT height WIDTH width}
[THICKNESS thickvalue] [LINESTYLE "stylename"] [LCOLOR linecolor]
[PATTERN "patname" [FCOLOR fillpatcolor]] [OPAQUE opaqueon]
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]
[WHEN condition] [MACRO "macroname"] [EXTENSION "PARAMETER=value"]
```

Parameters: *start_v*—the vertical measurement where the box should start, in the current units.

start_h—the horizontal measurement where the box should start, in the current units.

end_v—the vertical measurement where the box should end, in the current units.

end_h—the horizontal measurement where the box should end, in the current units.

height—the height of the box, in the current units. The default height may be set using the SET HEIGHT command.

width—the width of the box, in the current units. The default width may be set using the SET WIDTH command.

thickvalue—a value specifying the thickness of the outline in the current units. The default thickness may be set using the SET THICKNESS command.

stylename—a keyword indicating a line style. The default line style may be set using the SET LINESTYLE command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the box outline. The default line color can be set using the SET LCOLOR command.

patname—a keyword indicating a fill pattern. The default pattern may be set using the SET PATTERN command.

fillpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the fill pattern. The default fill pattern color can be set using the SET FCOLOR command.

opaqueon—a Boolean expression that indicates whether the box should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

offset—the offset between the box and the shadow, in the current units. A box with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the box is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

See also: FCOLOR, HEIGHT, LCOLOR, LINESTYLE, OPAQUE, PATTERN, SCOLOR, SHADOW, THICKNESS, UNITS, WIDTH

DRAW CIRCLE

This command is the same as the command DRAW ELLIPSE. See page 1-63.

DRAW COMB

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a horizontal or vertical comb at the specified position; a comb is a series of parallel lines, patterned bars, or both

Syntax:
```
DRAW COMB AT start_v,start_h
{TO end_v,end_h | HEIGHT height WIDTH width} BARS numbars
[DIRECTION {"HORIZONTAL"|"VERTICAL"}] [LINES linethick]
[LCOLOR linecolor] [PATTERN "patname" [FCOLOR fillpatcolor]]
[ALTERNATE alttrue] [OPAQUE opaqueon]
[BORDER borderon [THICKNESS thickvalue] [LINESTYLE "stylename"]]
```

```
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]
[WHEN condition] [MACRO "macroname"] [EXTENSION "PARAMETER=value"]
```

Parameters: *start_v*—the vertical measurement where the comb should start, in the current units.

start_h—the horizontal measurement where the comb should start, in the current units.

end_v—the vertical measurement where the comb should end, in the current units.

end_h—the horizontal measurement where the comb should end, in the current units.

height—the height of the comb, in the current units.

width—the width of the comb, in the current units.

numbars—the number of bars in the comb; the number of lines will be one less than the number of bars.

DIRECTION "VERTICAL"—changes the comb to vertical lines and bars; the default is horizontal.

linethick—a value specifying the thickness of the lines in the current units. To draw bars without lines, specify a thickness of 0, or do not use this parameter.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the lines and border. The default line color can be set using the SET LCOLOR command.

patname—a keyword indicating a fill pattern for the bars. The default pattern may be set using the SET PATTERN command. To draw lines without bars, specify a clear or white pattern.

Note: If the pattern is clear or white, and the line thickness is 0 or not specified, the comb does not print.

fillpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the fill pattern. The default fill pattern color can be set using the SET FCOLOR command.

alttrue—a Boolean expression that indicates whether to alternate white bars with the patterned bars. If it evaluates to "True", alternating bars are white, starting with the second bar; if "False", all bars are patterned. The default is "False".

opaqueon—a Boolean expression that indicates whether the comb should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box. The default border setting may be set using the SET BORDER command.

thickvalue—a value specifying the thickness of the border in the current units. The default thickness may be set using the SET THICKNESS command.

stylename—a keyword indicating a line style for the border. The default line style may be set using the SET LINESTYLE command.

offset—the offset between the comb and the shadow, in the current units. A comb with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the comb is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

See also: BORDER, FCOLOR, HEIGHT, LCOLOR, LINESTYLE, OPAQUE, PATTERN, SCOLOR, SHADOW, THICKNESS, UNITS, WIDTH

DRAW CURVE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a Bezier curve at the specified position

Syntax: DRAW CURVE AT *start_v*,*start_h* VIA *via1_v*,*via1_h* VIA *via2_v*,*via2_h* TO *end_v*,*end_h* [THICKNESS *thickvalue*] [LINESTYLE "*stylename*"] [LINECAP "*capname*"] [LCOLOR *linecolor*] [WHEN *condition*] [MACRO "*macroname*"]

Parameters: *start_v*—the vertical measurement where the curve should start, in the current units.

start_h—the horizontal measurement where the curve should start, in the current units.

via1_v, *via2_v*—the vertical measurements of the two points within the Bezier curve.

via1_h, *via2_h*—the horizontal measurements of the two points within the Bezier curve.

end_v—the vertical measurement where the curve should end, in the current units.

end_h—the horizontal measurement where the curve should end, in the current units.

thickvalue—a value specifying the thickness of the curve in the current units. The default thickness may be set using the SET THICKNESS command.

stylename—a keyword indicating a line style. The default line style may be set using the SET LINESTYLE command.

capname—a keyword indicating a line cap. The default line cap may be set using the SET LINECAP command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the curve. The default line color can be set using the SET LCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the curve is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

See also: LCOLOR, LINECAP, LINESTYLE, THICKNESS, UNITS

DRAW ELLIPSE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a circle or ellipse within the specified bounding box at the specified position.

Syntax:
```
DRAW ELLIPSE AT start_v,start_h
{TO end_v,end_h | RADIUS radius | HEIGHT height WIDTH width}
[THICKNESS thickvalue] [LINESTYLE "stylename"] [LCOLOR linecolor]
[PATTERN "patname" [FCOLOR fillpatcolor]] [OPAQUE opaqueon]
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]
[WHEN condition] [MACRO "macroname"] [EXTENSION "PARAMETER=value"]
```

Parameters: *start_v*—the vertical measurement where the bounding box should start, in the current units; if drawing a circle using RADIUS, the vertical measurement of the center.

start_h—the horizontal measurement where the bounding box should start, in the current units; if drawing a circle using RADIUS, the horizontal measurement of the center.

end_v—the vertical measurement where the bounding box should end, in the current units. The vertical distance between the starting and ending points determines the height of the ellipse.

end_h—the horizontal measurement where the bounding box should end, in the current units. The horizontal distance between the starting and ending points determines the width of the ellipse. To draw a circle, the width must equal the height.

radius—the radius of the circle, in the current units. This forces the ellipse to be a circle, and causes the AT point to be the center. The default radius can be set using the SET RADIUS command.

height—the height of the bounding box, in the current units. This is also the height of the ellipse.

width—the width of the bounding box, in the current units. This is also the width of the ellipse. To draw a circle, the width must equal the height.

thickvalue—a value specifying the thickness of the outline in the current units. The default thickness can be set using the SET THICKNESS command.

stylename—a keyword indicating a line style. The default line style may be set using the SET LINESTYLE command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the ellipse outline. The default line color can be set using the SET LCOLOR command.

patname—a keyword indicating a fill pattern. The default pattern may be set using the SET PATTERN command.

fillpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the fill pattern. The default fill pattern color can be set using the SET FCOLOR command.

opaqueon—a Boolean expression that indicates whether the ellipse should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

offset—the offset between the ellipse and the shadow, in the current units. An ellipse with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the ellipse is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

See also: FCOLOR, HEIGHT, LCOLOR, LINESTYLE, OPAQUE, PATTERN, SCOLOR, SHADOW, THICKNESS, UNITS, WIDTH

DRAW IMAGE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a graphical image at the specified position

Syntax:
```
DRAW IMAGE AT start_v,start_h USING imagename
[{TO end_v,end_h | HEIGHT height WIDTH width} [ALIGN "alignmnt"]
[BORDER borderon [THICKNESS thickvalue] [LCOLOR linecolor]]]
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]
[ROTATION angle] [OPAQUE opaqueon] [REVERSE revtrue] [WHEN condition]
[MACRO "macroname"] [EXTENSION "PARAMETER=value" [LINK "url"]]
```

Parameters: *start_v*—the vertical measurement where the image should start (the upper left corner), in the current units.

start_h—the horizontal measurement where the image should start (the upper left corner), in the current units.

imagename—a literal string or variable containing the path and file name of the image. The image may be in .BMP, .TIF, .PCX, or .GIF format. The default path is ~images. OptioDCS supports 256 color images for PostScript, GDI, PDF, and RTF devices, in .BMP, .TIF, .PCX, .GIF, or .PNG format (300 dpi only).

end_v—the vertical measurement where the image bounding box should end, in the current units. The TO point defines a bounding box, used to crop the image, to align the image, or to draw a border. If the bounding box is smaller than the image, the image is cropped to fit the box.

end_h—the horizontal measurement where the image bounding box should end, in the current units.

height—the height of the image bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box, used to crop the image, to align the image, or to draw a border. If the bounding box is smaller than the image, the image is cropped to fit the box. The default height may be set using the SET HEIGHT command.

width—the width of the image bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box, used to crop the image, to align the image, or to draw a border. If the bounding box is smaller than the image, the image is cropped to fit the box. The default width may be set using the SET WIDTH command.

alignmnt—a keyword indicating the alignment of the image within the bounding box; the default is top left. If the bounding box is smaller than the image, the alignment determines which edges of the image are not cropped. The default alignment may be set using the SET ALIGN command.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box. The default border setting may be set using the SET BORDER command.

thickvalue—a value specifying the thickness of the border in the current units. The default thickness may be set using the SET THICKNESS command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the border. The default line color can be set using the SET LCOLOR command.

offset—the offset between the image and the shadow, in the current units. An image with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

angle—the counter-clockwise angle of rotation. Valid values are 0, 90, 180, and 270. The default angle may be set using the SET ROTATION command.

opaqueon—a Boolean expression that indicates whether the image should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

revtrue—a Boolean expression that indicates whether to print the image in reverse, meaning that white pixels become black and black pixels become white. The default is "False".

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the image is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

url—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a destination URL. This creates a hyperlink within the text bounding box for PDF output. For more information, see Chapter 4 of the *Advanced Features Guide*.

See also: BORDER, HEIGHT, LCOLOR, OPAQUE, REVERSE, ROTATION, SCOLOR, SHADOW, THICKNESS, UNITS, WIDTH

DRAW LINE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a line at the specified position

Syntax:
```
DRAW LINE AT start_v,start_h TO end_v,end_h [THICKNESS thickvalue]
[LINESTYLE "stylename"] [LINECAP "capname"] [LCOLOR linecolor]
[WHEN condition] [MACRO "macroname"] [EXTENSION "PARAMETER=value"]
```

Parameters: *start_v*—the vertical measurement where the line should start, in the current units.

start_h—the horizontal measurement where the line should start, in the current units.

end_v—the vertical measurement where the line should end, in the current units.

end_h—the horizontal measurement where the line should end, in the current units.

thickvalue—a value specifying the thickness of the line in the current units. The default thickness may be set using the SET THICKNESS command.

stylename—a keyword indicating a line style. The default line style may be set using the SET LINESTYLE command.

capname—a keyword indicating a line cap. The default line cap may be set using the SET LINECAP command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the line. The default line color can be set using the SET LCOLOR command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the line is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

See also: LCOLOR, LINECAP, LINESTYLE, THICKNESS, UNITS

DRAW OBJECT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Draws a page that was designed in another application, and saved as an Encapsulated PostScript or HP-PCL file, as an image at the specified position

Syntax:
```
DRAW OBJECT AT start_v,start_h USING objfile
[{TO end_v,end_h | HEIGHT height WIDTH width}
[BORDER borderon [THICKNESS thickvalue] [LCOLOR linecolor]]
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]]
[OPAQUE opaqueon] [ROTATION angle] [WHEN condition]
[MACRO "macroname"] [EXTENSION "PARAMETER=value"]
```

Parameters: *start_v*—the vertical measurement where the image should start (the upper left corner), in the current units.

start_h—the horizontal measurement where the image should start (the upper left corner), in the current units.

objfile—a literal string or variable containing the .EPS or .PCL file path and name.

end_v—the vertical measurement where the image bounding box should end, in the current units. The TO point defines the bounding box used to draw a border or shadow. For PostScript printer drivers, the bounding box is also used to scale the image.

end_h—the horizontal measurement where the image bounding box should end, in the current units.

height—the height of the image bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box used to draw a border or shadow. For PostScript printer drivers, the bounding box is also used to scale the image. The default height may be set using the SET HEIGHT command.

width—the width of the image bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box used to draw a border or shadow. For PostScript printer drivers, the bounding box is also used to scale the image. The default width may be set using the SET WIDTH command.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box. The default border setting may be set using the SET BORDER command.

thickvalue—a value specifying the thickness of the border in the current units. The default thickness may be set using the SET THICKNESS command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the border. The default line color can be set using the SET LCOLOR command.

offset—the offset between the image and the shadow, in the current units. An object with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

opaqueon—a Boolean expression that indicates whether the image should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is “Off”. The default may be set using the SET OPAQUE command.

angle—the counter-clockwise angle of rotation. Valid values are 0, 90, 180, and 270. This is available only for PostScript printer drivers. The default angle may be set using the SET ROTATION command.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

macroname—the name of a macro; this indicates that the object is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

See also: BORDER, HEIGHT, LCOLOR, OPAQUE, ROTATION, SCOLOR, SHADOW, THICKNESS, UNITS, WIDTH

DRAW ROW

This command is the same as the command DRAW LINE. See page 1-67.

DRAW TEXT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Graphics command

Purpose: Prints text at the specified position

Syntax:

```
DRAW TEXT AT start_v,start_h USING content
[FONT fontname [POINTS "pointsize"] |
TYPEFACE "typename" POINTS "pointsize" [SYMBOLS "symset"]]
[{HEIGHT boxheight WIDTH boxwidth | TO end_v,end_h}
[ALIGN "alignment"] [WORDWRAP wrapon [JUSTIFY juston]]
[BORDER borderon [THICKNESS thickvalue] [LCOLOR linecolor]]
[PATTERN "patname" [FCOLOR fillpatcolor]]
[SHADOW offset,["shadpat"] [,"corner"] [SCOLOR shadpatcolor]]]
[TCOLOR textcolor] [ROTATION angle]
[SPACING spacevalue] [LINES numlines] [LEADING leadvalue]
[REVERSE revtrue] [OPAQUE opaqueon] [UNDERLINE undtrue]
[WHEN condition] [FIELD fieldname] [MACRO "macroname"]
[EXTENSION "PARAMETER=value" [LINK "url"]]
```

Parameters: *start_v*—the vertical measurement where the text should start (the baseline), in the current units. When printing an array, this is the vertical measurement of the baseline of the first element.

start_h—the horizontal measurement where the text should start, in the current units.

content—the literal string, a variable containing the text string, or an expression that evaluates to the string, that should be printed.

fontname—the name or alias of a font, as specified in a device or fontpack module. The default font may be set using the SET FONT command.

pointsize—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. If used with the FONT clause, this overrides the point size of the font. For PostScript printer drivers, an optional horizontal point size is supported; for example POINTS "12,10" specifies a 12 point height and a 10 point width. The default points may be set using the SET POINTS command.

typename—the name of a typeface that is available on your printer. The default typeface may be set using the SET TYPEFACE command.

symset—the name of a symbol set that has a tagged font metrics (.tfm) file and is supported by your printer. The default symbol set may be set using the SET SYMBOLS command.

boxheight—the height of the text bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box, used to align the text or to draw a border or shadow. The default height may be set using the SET HEIGHT command.

boxwidth—the width of the text bounding box, in the current units. The HEIGHT and WIDTH clauses define a bounding box, used to align the text or to draw a border or shadow. The default width may be set using the SET WIDTH command; otherwise, the default bounding box width is the width of the text.

end_v—the vertical measurement where the text bounding box should end, in the current units. The TO point defines a bounding box, used to align the text or to draw a border or shadow.

end_h—the horizontal measurement where the text bounding box should end, in the current units.

alignment—a keyword indicating the alignment of the text within the bounding box; the default is text left. The default alignment may be set using the SET ALIGN command.

wrapon—a Boolean expression that indicates whether the text should wrap between the left and right edges of the bounding box. The default is "Off". The default may be set using the SET WORDWRAP command.

juston—a Boolean expression that indicates whether the wrapped text should be justified, so that it is aligned to both the left and right edges of the bounding box. The default is "Off". The default may be set using the SET JUSTIFY command.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box. The default border setting may be set using the SET BORDER command.

thickvalue—a value specifying the thickness of the border in the current units. The default thickness may be set using the SET THICKNESS command.

linecolor— a literal string or an expression that evaluates to a string, containing the name of a color for the border. The default line color can be set using the SET LCOLOR command.

patname—a keyword indicating a fill pattern for the bounding box. The default pattern may be set using the SET PATTERN command.

fillpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the fill pattern. The default fill pattern color can be set using the SET FCOLOR command.

offset—the offset between the bounding box and the shadow, in the current units. The shadow is the same size and shape as the bounding box. Text with a shadow is always opaque.

shadpat—a keyword indicating a fill pattern for the shadow.

corner—a keyword indicating the direction of the shadow.

shadpatcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the shadow pattern The default shadow pattern color can be set using the SET SCOLOR command.

textcolor— a literal string or an expression that evaluates to a string, containing the name of a color for the text. The default text color can be set using the SET TCOLOR command.

angle—the counter-clockwise angle of rotation. Valid values are 0, 90, 180, and 270. The default angle may be set using the SET ROTATION command.

spacevalue—a value specifying the horizontal spacing of the characters, in the current units. This forces fixed spacing of any font. The default spacing may be set using the SET SPACING command.

numlines—the number of elements to print from an array, beginning with the first element. The default is all elements of the array. The default may be changed using the SET LINES command.

leadvalue—a value specifying the vertical line spacing for printing an array, in the current units. The default leading may be set using the SET LEADING command.

revtrue—a Boolean expression that indicates whether to print the text in white or reverse, to contrast with a dark background. The default is "False".

opaqueon—a Boolean expression that indicates whether the bounding box should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". The default may be set using the SET OPAQUE command.

undtrue—a Boolean expression that indicates whether to underline the text. The default is "False".

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed.

fieldname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a name for the field. When using a PDF or RTF device, this creates a field (containing the USING text), for use in other applications, such as Optio e.ComPresent and Microsoft Word. The field data is used to index and burst the document in Optio e.ComPresent.

macroname—the name of a macro; this indicates that the text is part of the macro. For more information on macros, see the PCL section of Chapter 2 in this book.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

url—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a destination URL. This creates a hyperlink within the text bounding box for PDF output. For more information, see Chapter 4 of the *Advanced Features Guide*.

See also: ALIGN, BORDER, FCOLOR, FONT, HEIGHT, LCOLOR, LEADING, OPAQUE, PATTERN, REVERSE, ROTATION, SCOLOR, SHADOW, SPACING, TCOLOR, THICKNESS, UNITS, WIDTH

DUPLEX

This clause is used with the commands below. See the entries for those commands.

See: DEVICE, FORMAT

ELIF

This clause is used with the command below. See the entry for that command.

See: IF

ELSE

This clause is used with the command below. See the entry for that command.

See: IF

EMPTY

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if the last page of data read by an input channel contains no data, except for carriage return codes, line feed codes, and form feed codes; returns FALSE if any other data is found.

Syntax: `EMPTY(`*`channelstring`*`)`

Parameters: *channelstring*— a literal string or variable containing the name of an input channel (the predefined input channel is stdin)

Examples: IF EMPTY("stdin") THEN

IF EMPTY(acctgreports) THEN

See also: BREAK, EOF, ISBLANK

END

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Marks the end of a module, conditional statement, or loop.

Syntax: `END {CHANNEL | DATAMAP | DEVICE | DOCUMENT | FONTPACK | FOR | FORMAT | FORM | FUNCTION | IF | PALETTE | PART | REGION | SEGMENT | SQL | SWITCH | TRANSLATE | WHILE}`

See also: CHANNEL, DATAMAP, DEVICE, DOCUMENT, EXIT, FONTPACK, FOR, FORMAT, FUNCTION, IF, PALETTE, PART, REGION, SEGMENT, SQL, SWITCH, TRANSLATE, WHILE

EOF

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns FALSE if an input channel contains more pages of data; returns TRUE if not. If your document does not use the command PROCESS PARTS, use this function to test for the end of the input data, so that your document can tell OptioDCS to stop reading data when it reaches the end.

Syntax: `EOF(channelstring)`

Parameters: *channelstring*— a literal string or variable containing the name of an input channel (the predefined input channel is stdin)

Examples:
```
WHILE NOT EOF("stdin")

IF EOF(acctgreports) THEN
```

See also: BREAK, EMPTY

ESCAPE

This function is the same as the function CANON. See page 1-37.

EXIT

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Causes the processing of a module, conditional statement, or loop to end immediately.

Syntax: `EXIT {DATAMAP | DOCUMENT | FOR | FORMAT | FORM | FUNCTION | PART | REGION | SEGMENT | SWITCH | WHILE}`

See also: DATAMAP, DOCUMENT, END, FOR, FORMAT, FUNCTION, PART, REGION, RETURN, SEGMENT, SWITCH, WHILE

EXPORT

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Assigns a value to a system parameter.

Syntax: `EXPORT(`*`assignstring`*`)`

Parameters: *assignstring*— a literal string which contains a statement assigning a value to a system parameter

Example:
```
LET status = EXPORT("PRINTER=myprn")
```

See also: IMPORT, PARAMETER, TRANSMIT

EXTENSION

This clause is used with the commands below. See the entries for those commands.

See: DOCUMENT, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, FORMAT, PART, SEGMENT

FAMILY

This clause is used with the command below. See the entry for that command.

See: FONT

FATALFILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the name of the fatal file.

Syntax: `FATALFILE()`

Examples:

```
LET filename = FATALFILE()

LET string = "Log: " & FATALFILE()
```

FAX

Doc ■ Par ■ Fun ■ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Sets fax settings to the specified values. This command should be placed after the SET OUTPUT command that selects a fax channel. This command may be used with or without an OptioFAX phonebook. For more information, see Chapter 3 of the *Advanced Features Guide*.

Syntax: **Without a Phonebook**

FAX NUMBER *faxnum* [AREACODE *faxarea*] [SUFFIX *faxsuff*] [COUNTRY *faxcntry*] [PREFIX *faxpref*] [EXTENSION *faxext*]
[COVERSHEET *coveron* [FROM *fromname*] [SUBJECT *subjstring*] [NOTES *notearray*] [COMPANY *coname*]
[NAME *recipname* | FIRSTNAME *recipfname* LASTNAME *reciplname*]]
[OPTIONS *optkeywords*] [NOTIFY *notifyaddr*]
[CLASS *classname* [, *altclassname*]]
[SERVER *servername*] [PORT *portnum*] [[ACCESSKEY *keyname*] |
[USERNAME *faxuser*] [PASSWORD *faxpass*]] [TAG *tagstring*]
[DATE *senddate*] [TIME *sendtime*] [PRIORITY *priorlevel*]
[RETRY *atmpts1*,*intvl1*[,*atmpts2*,*intval2*[*atmpts3*,*intval3*]]]

With a Phonebook

FAX PHONEBOOK *phbookname* [[COMPANY *coname*]
[FIRSTNAME *recipfname*] [LASTNAME *reciplname*] [NUMBER *faxnum*] [AREACODE *faxarea*] [COUNTRY *faxcntry*] [EXTENSION *faxext*]]
[PREFIX *faxpref*] [SUFFIX *faxsuff*]
[COVERSHEET *coveron* [NAME *recipname*] [FROM *fromname*] [SUBJECT *subjstring*] [NOTES *notearray*]]
[OPTIONS *optkeywords*] [NOTIFY *notifyaddr*]
[CLASS *classname* [, *altclassname*]]
[SERVER *servername*] [PORT *portnum*] [[ACCESSKEY *keyname*] |
[USERNAME *faxuser*] [PASSWORD *faxpass*]] [TAG *tagstring*]
[DATE *senddate*] [TIME *sendtime*] [PRIORITY *priorlevel*]
[RETRY *atmpts1*,*intvl1*[,*atmpts2*,*intval2*[*atmpts3*,*intval3*]]]

Parameters: *phbookname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of the phonebook containing the fax recipient information. Any phonebook entry that matches the specified company, first name, last name, fax number, area code, country code, or extension is selected. If none of these are specified, all entries in the phonebook are selected. Phonebooks are created using OptioFAX; for more information, see the OptioFAX documentation.

faxnum—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the (local) fax phone number of the recipient. If using a phonebook, all entries matching this are selected. You may specify a partial number, to select all entries that start with that number (e.g., "31" matches all entries with a number beginning with "31").

faxarea—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the area code of the recipient, if necessary. If using a phonebook, all entries matching this are selected. You may specify a partial number, to select all entries that start with that number (e.g., "31" matches all entries with an area code beginning with "31").

faxsuff—a literal string, a variable containing a string, or an expression that evaluates to a string, containing any suffix required by the sending telephone system, such as a long distance account code; use a comma (,) to indicate a pause.

faxcntry—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the country code of the recipient, if necessary. If using a phonebook, all entries matching this are selected. You may specify a partial number, to select all entries that start with that number (e.g., "31" matches all entries with a country code beginning with "31").

faxpref—a literal string, a variable containing a string, or an expression that evaluates to a string, containing any prefix required by the sending telephone system; use a comma (,) to indicate a pause.

faxext—a literal string, a variable containing a string, or an expression that evaluates to a string, containing any extension required by the receiving telephone system. If using a phonebook, all entries matching this are selected. You may specify a partial number, to select all entries that start with that number (e.g., "31" matches all entries with an extension beginning with "31").

coveron—a Boolean expression that indicates whether to include the standard cover sheet in the fax.

fromname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the sender for the cover sheet.

subjstring—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the subject for the cover sheet.

notearray—an array of literal strings, variables containing strings, or expressions that evaluate to strings, containing notes for the cover sheet; use empty strings ("") for blank lines.

coname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the recipient's company for the cover sheet. If using a phonebook, all entries matching this are selected. You may specify a partial name, to select all entries that start with that string (e.g., "Sm" matches all entries with a company name beginning with "Sm").

recipname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the recipient for the cover sheet. If using a phonebook, this overrides the names retrieved from the phonebook.

recipfname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the first name of the recipient for the cover sheet. If using a phonebook, all entries matching this are selected; you may specify a partial

name, to select all entries that start with that string (e.g., "Da" matches all entries with a first name beginning with "Da").

reciplname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the last name of the recipient for the cover sheet. If using a phonebook, all entries matching this are selected; you may specify a partial name, to select all entries that start with that string (e.g., "Sm" matches all entries with a last name beginning with "Sm").

optkeywords—an array of keywords indicating which send options to turn on, as listed below. Some special options are on by default, as configured in OptioFAX.

Send Option	Keyword
use fine mode regardless of the recipient fax system; this is necessary for faxing Postscript documents	"FORCE_FINE"
use fine mode if the recipient fax system supports it	"FINE_MODE"
print sender information at the top of the page	"ID_STRIP"
issue an e-mail message when the fax request leaves the queue; use with the NOTIFY clause	"NOTIFY_BY_MAIL"
restart interuppted transmissions at the first page	"NO_PARTIAL"
remove the request from the queue after the first attempt to send, regardless of the OptioFAX retry schedule	"EARLY_ABORT"
ignore retransmission requests from the recipient's fax system	"NO_REFEED"
dial the modem silently	"SILENT"

notifyaddr—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the e-mail address for the notification when the fax request leaves the queue; use with the "NOTIFY_BY_MAIL" option.

classname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the modem server class to use for the fax. Modem server classes are defined by the OptioFAX administrator; see the OptioFAX documentation for more information.

altclassname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of an alternate modem server class to use for the fax if the first one is not available.

servername—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the remote OptioFAX API server used to send the fax. (For UNIX systems, this is the DNS name or IP address of the server). This overrides the server specified in the output channel module. For more information, see Chapter 3 of the *Advanced Features Guide*.

portnum—a constant, variable, or expression, that evaluates to a numerical value, indicating the port or socket number assigned to the OptioFAX API software. This overrides the port number specified in the output channel module. The default is 3000. For more information, see Chapter 3 of the *Advanced Features Guide*.

keyname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key used by the client system to log on to the OptioFAX API server. For more information on access keys, see Chapter 2 of the *User's Guide*.

faxuser, *faxpass*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the user name and password used by the client system to log into the OptioFAX API server. This overrides the user name and password specified in the output channel module.

For UNIX systems, if the OptioFAX security database has been implemented, the user name for the security database should be specified.

For NT systems:

- If OptioFAX is running on an NT server, use the NT user account name prefaced with its domain name (i.e., USERNAME "*domain\username*"). The user name must be valid to log in locally to both the OptioFAX server and the OptioDCS server.
- If OptioFAX is running on a UNIX server, the user name must be valid both in OptioFAX security and on the NT server.

tagstring—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a name for the fax queue entry, up to 21 characters including spaces.

senddate—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the date to send the fax, in the form mm/dd/yy, mmddyy, or yymmdd (e.g., 04/30/97, 043097, or 970430). The default is "TODAY".

sendtime—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the time to send the fax, in the form hh:mm or hhmm (e.g., 18:30 or 1830). The default is "NOW".

priorlevel—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the fax priority, from 0 (highest) to 9 (lowest). The default value is 5. Faxes are sent in priority order.

atmpts1, *intvl1*, etc.—constants, variables, or expressions, that evaluate to numerical values, indicating the retransmission schedule if the first attempt is unsuccessful. The *attempts* is the number of attempts, and *interval* is the number of minutes to wait between attempts. You may specify up to three sets of attempts (e.g., "5","4","5","10","5","15"). The default retry schedule is 3 at 10 minutes, 3 at 15 minutes, and 3 at 20 minutes.

See also: BREAK ON, FAX RESET, SERVER, SET OUTPUT

FAX RESET

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ □ □ □ □ □ □ □

Type: Command

Purpose: Resets fax settings to default values (empty strings).

Syntax: `FAX RESET`

See also: BREAK ON, FAX

FCOLOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Selects a color for the fill pattern of an object. This applies to the stripe, crosshatch, and solid ("U") fill patterns. For more information, see Chapter 2 of the *Advanced Features Guide*.

Affects: arcs, boxes, circles, combs, and ellipses

Syntax: `FCOLOR` *`fillpatcolor`*

Default: black

Parameters: *fillpatcolor*— a literal string or an expression that evaluates to a string, containing the name of a color defined in the current palette module. The palette module is selected in the device module. The default fill pattern color can be set using the SET FCOLOR command.

See also: COLOR, DEVICE, DRAW ARC, DRAW BOX, DRAW COMB, DRAW ELLIPSE, LCOLOR, PALETTE, PATTERN, SCOLOR, SET

FEEDER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Defines a keyword for a paper tray or feeder available on a device.

Syntax: `FEEDER "trayword","traycode"`

Parameters: *trayword*—a keyword for the paper tray or feeder.

traycode—the code that indicates the corresponding paper tray or feeder to the printer.

See also: PAPER, STACKER

2

This clause is used in a different mode with the command below. See the entry for that command.

See: FORMAT

FETCH

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Calls an SQL module for executing, logging on to the database with the access key and executing the SQL statement as specified in the module. An SQL module retrieves, inserts, or deletes SQL database data. If the SQL module contains a ROWS clause, the next set of matching records are affected when the module is executed. For more information, see Chapter 5 of the *Advanced Features Guide*.

Syntax: `FETCH sqlmodname`

Parameters: *sqlmodname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of the SQL module.

Example:
```
PART "fileOrders"
   OPEN "getorders"
   LET rc = SQLERROR()
   WHILE rc = 0
      FETCH "getorders"
      LET rc = SQLERROR()
      DRAW "fileCopy"
   END WHILE
   CLOSE "getorders"
END PART
```

Note: Before using this command, the SQL module must be opened as an input channel using the OPEN command. After using this command, the SQL module should be closed as an input channel using the CLOSE command.

See also: CALL, RUN, SET INPUT, SQL, SQLERROR, SQLRUN

FIELD

This clause is used with the command below. See the entry for that command.

See: DRAW TEXT

FIELDS

This clause is used with the command below. See the entry for that command.

See: CHANNEL

FILE
1

This clause is used with the command below. See the entry for that command.

See: CHANNEL

2

This word is part of the commands listed below. See the entries for those commands.

See also: SET LOG FILE, SET TRACE FILE, SET WARNING FILE

FILENAME

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the name of the temporary file which is created to store the data being sent to a smart output channel. A smart channel is used to pass information, in addition to processed documents, to applications. The channel module must have the command SMART "On". *This function is not valid for Windows NT systems.*

Syntax: `FILENAME(channelstring)`

Parameters: *channelstring*— a literal string or variable containing the name of an output channel (predefined output channels are stdout and stderr)

Examples:
```
LET tempname = FILENAME(accntg)

IF FILENAME("stdout") = name2 THEN
```

See also: TRANSMIT

FIND

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Searches a string for a match of an expression; if a match is found, returns the first matching substring; otherwise, returns an empty string. If searching an array of strings, returns an array of the first matching substring in each element. The search expression can be a regular expression, which is a notation for describing patterns of characters (see your Windows NT or UNIX system documentation for more information).

Syntax: `FIND(searchexpr,string)`

Parameters: *searchexpr*— a literal string, a variable containing a string, or an expression or regular expression that evaluates to a string, containing the expression to find

string— a literal string, a variable containing a string or strings, or an expression that evaluates to a string, containing the string to search for the expression

Examples:

```
LET itemcode = FIND("D* ",items[x])

WHILE NOT (FIND(names[i],info[i,1:length]) = "")

IF FIND("PAST DUE",@[1:3,1:80]) = "" THEN
```

See also: LOOKUP, MATCHES, SORT, SPAN, WHERE

FIX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Makes each element in an array of strings a specified length, by truncating excess characters or padding short strings with spaces.

Syntax: `FIX(stringarray, newlength)`

Parameters: *stringarray*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

newlength—a numeric expression indicating the desired number of characters.

Examples:

```
LET initials = FIX(lastnames, 1)

LET shortdesc = FIX(fulldesc, fieldlength)
```

See also: COLUMNS, LENGTH, SPAN

FLATTEN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Determines where the paragraph breaks should be in an array of text strings, and returns a rearranged array of strings, placing each paragraph into a single string (array element). Any series of two or more spaces at the end of a string, following a period,

question mark, exclamation point, or colon, is interpreted as a paragraph break. If the result is assigned to the same array, the extra array elements are replaced with empty strings.

Syntax: FLATTEN(*array*)

Parameters: *array*— an array variable containing text strings

Examples:
```
LET paragraphs = FLATTEN(textarray)

LET messages = FLATTEN(messages)
```

See also: ROWS, SQUISH

FLOATING

This clause is used with the command below. See the entry for that command.

See: REGION

FLUSH

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Flushes the temporary buffer, by copying any data from the buffer, and outputting it as specified in a channel module. FLUSH OUTPUT copies to the current output channel, or another channel may be specified.

Syntax: FLUSH {OUTPUT | "*channelname*"}

Parameters: *channelname*—the name of a channel.

See also: CHANNEL

FOLDER

This clause is used with the command below. See the entry for that command.

See: MAIL

FONT

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

1

Type: Clause

Purpose: Controls which font is used to print text.

Affects: text

Syntax: `FONT "fontname" [POINTS "pointsize"]`

Default: the default font for the output device

Parameters: *fontname*—the name or alias of a font, as specified in a device or fontpack module.

pointsize—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. This overrides the point size of the font. For PostScript printer drivers, an optional horizontal point size is supported; for example POINTS "12,10" specifies a 12 point height and a 10 point width.

See also: DRAW TEXT, FORMAT, POINTS, SEGMENT, SET

2

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ □ □ □ ■ ■

Type: Command

Purpose: Defines a font or font alias in a device or fontpack module, which makes the font or alias available for any document that calls the device or fontpack module.

Syntax:
```
FONT "fontname" {LIKE "oldfontname" |
TYPEFACE "typename" SYMBOLS "symset" [SYMBOLS "symset" ...]
{METRICS "metricstable" | COMMAND "escstring"}}
[STYLE "fontstyle" FAMILY "fontfam" MONOSPACED monotrue]
[POINTS "pointsize"] [GLYPHS "glyphpath"]
```

Parameters: *fontname*—a name for the font, if creating a new font, or an alias for an existing font.

oldfontname—the name of an existing font. The LIKE clause is used to create an alias.

typename—the name of a typeface that is available on your printer. The TYPEFACE, SYMBOLS, and METRICS clauses are used to create a new font.

symset—the name of a symbol set that has a tagged font metrics (.tfm) file and is supported by your printer. Multiple symbol sets may be specified; the first is used as the default. The TYPEFACE, SYMBOLS, and METRICS clauses are used to create a new font.

metricstable—the name of a file containing a font metrics table. The TYPEFACE, SYMBOLS, and METRICS clauses are used to create a new font.

escstring—the font escape string; substitute "\E" for the escape character. The COMMAND clause may be used, instead of the METRICS clause, to create a new font without a metrics file.

fontstyle—the font style as specified in the .ttm file. This is required for .ttm fonts.

fontfam—the font family as specified in the .ttm file. This is required for .ttm fonts.

monotrue—a boolean value indicating whether the font is monospaced, as specified in the .ttm file. This is required for .ttm fonts.

pointsize—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. If used with the LIKE clause to create an alias, this overrides the point size of the existing font. For PostScript printer drivers, an optional horizontal point size is supported; for example POINTS "12,10" specifies a 12 point height and a 10 point width.

glyphpath—the path and name of an Adobe PostScript FontType 1 glyph definition (.pfb) file (IBM type), used to define an auto-downloadable font for PDF output. The first time each auto-downloadable font is used in a document, OptioDCS automatically dowloads the glyph definition file into the PDF file.

See also: POINTS, SYMBOLS, TYPEFACE

FONTPACK

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Module command

Purpose: Defines a fontpack module. A fontpack module contains font specifications describing a group of fonts that are available on a device.

Syntax: `FONTPACK "packname" [LIKE "existpack"]`

Parameters: *packname*—a name for the fontpack.

existpack—the name of an existing fontpack. The LIKE clause copies all of the font specifications in the existing fontpack to the new fontpack.

Note: The fontpack module must end with the END FONTPACK command.

See also: END

2

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Includes the font specifications from a fontpack module in the definition of a device module. This does not download fonts to the device, but makes the fonts that are already available on the device available for use in an OptioDCS document.

Syntax: `FONTPACK "packname"`

Parameters: *packname*—the name of an existing fontpack module.

See also: FONT

FOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Repeats the execution of a set of commands until a counter reaches a specified value. The counter is set to a starting value. Then the commands between the FOR command and the END FOR command are executed, and the counter is incremented by 1. This

repeats until the counter is greater than the ending value or an EXIT FOR command is executed within the set of commands.

Syntax: `FOR counter = startvalue TO endvalue commands END FOR`

Parameters: *counter*—a variable used as the counter for the FOR statement.

startvalue—a numeric expression indicating the starting value of the *counter*.

endvalue—a numeric expression indicating the ending value of the *counter*.

commands—the commands that are executed repeatedly while the *counter* is less than the *endvalue*. The commands must be valid for use in the current module.
Note: The counter is incremented by 1 each time the commands are executed, but the commands may also change its value.

Note: The FOR statement must end with the END FOR command.

See also: END, EXIT, WHILE

FORMAT / FORM

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Module command

Purpose: Defines a format module. A format module specifies how to format the data, and what other text and graphics to place on the document.

Syntax:
```
FORMAT "formatname" [FEEDER "papertray"] [PAPER "papersize"] [DUPLEX
"duplextype" [SURFACE "side"]] [LAYOUT "orientation"] [STACKER
"outtray"] [MACRO "macname"]
[MACROS rangemin[:rangemax]] [ACCEPTS param1 [,param2 ...]]
[UNITS "unittype"] [THICKNESS thickvalue]
[LINESTYLE "stylename"] [LINECAP "capname"]
[HEIGHT height] [WIDTH width] [PATTERN "patname"]
[OPAQUE opaqueon] [SHADOW offset,["shadpat"] [,"corner"]]
[REVERSE revtrue] [ROTATION angle] [BORDER borderon]
[RADIUS radius] [FONT fontname [POINTS "pointsize"] |
TYPEFACE "typename" POINTS "pointsize" [SYMBOLS "symset"]] [ALIGN
"alignment"] [UNDERLINE undtrue] [SPACING spacevalue] [LEADING
leadvalue] [LINES numlines]
[DENSITY density] [RATIO ratio] [STYLE "style"]
[EXTENSION "PARAMETER=value"]
```

Parameters: *formatname*—a name for the format.

papertray—a keyword indicating which paper tray or feeder to use for printing. The feeder keywords are defined in the device module.

papersize—a keyword indicating a paper size. The paper size keywords are defined in the device module.

duplextype—a keyword indicating a type of duplex, as listed in the table below. The default is "Off".

Type	Keyword
no duplexing	Off
normal duplexing/long edge binding	LongEdge
tumbled duplexing/short edge binding	ShortEdge

side—a keyword indicating which side of the paper should be used for printing this format, as listed in the table below.

Side	Keyword
always on the front	Front
always on the back	Back
whichever side is next	Either

orientation—a keyword indicating the orientation of the page, either "Portrait" or "Landscape".

outtray—a keyword indicating which output tray or stacker to use on the printer. The stacker keywords are defined in the device module.

macname—a name for a macro; this creates a macro, including all DRAW statements in the format module, until the first SET MACRO statement. Macros should include only static text and graphic elements. For more information on macros, see the PCL section of Chapter 2 in this book.

rangemin, *rangemax*—constants, variables, or expressions that evaluate to numerical values, indicating the range of memory slots to use for macros from this format. The macros in the format are automatically numbered consecutively, using this range. This range must fall within the range specified in the device module, and must not overlap any ranges specified in other format modules of the same document.

param1, *param2*—variable names for the parameter values that are passed into the format. The values passed in are assigned to these variables, which are used within the format module. A format may have any number of parameters.

unittype—a keyword indicating the units of measurement. This sets the default for the format module.

thickvalue—a value specifying a line thickness in the current units. This sets the default for the format module.

stylename—a keyword indicating a line style. This sets the default for the format module.

capname—a keyword indicating a line cap. This sets the default for the format module.

height—a numeric expression indicating an object height in the current units. This sets the default for the format module.

width—a numeric expression indicating an object width in the current units. This sets the default for the format module.

patname—a keyword indicating a fill pattern. This sets the default for the format module.

opaqueon—a Boolean expression that indicates whether objects should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". This sets the default for the format module.

offset—a numeric expression indicating an offset between objects and their shadows, in the current units. An object with a shadow is always opaque. This sets shadows on, and the default offset for the format module.

shadpat—a keyword indicating a fill pattern for shadows. This sets the default for the format module.

corner—a keyword indicating the direction of shadows. This sets the default for the format module.

revtrue—a Boolean expression that indicates whether to print images and text in reverse, meaning that white pixels become black and black pixels become white. The default is "False". This sets the default for the format module.

angle—the counter-clockwise angle of rotation of objects. Valid values are 0, 90, 180, and 270. This sets the default for the format module.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box of an object. This sets the default for the format module.

radius—the radius of circles, in the current units. This forces the ellipses to be circles, and causes the AT point to be the center. This sets the default for the format module.

fontname—the name or alias of a font, as specified in a device or fontpack module. This sets the default for the format module.

pointsize—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. If used with the FONT clause, this overrides the point size of the font. For PostScript printer drivers, an optional horizontal point size is also supported. This sets the default for the format module.

typename—the name of a typeface that is available on your printer. This sets the default for the format module.

symset—the name of a symbol set that has a tagged font metrics (.tfm) file and is supported by your printer. This sets the default for the format module.

alignment—a keyword indicating the alignment of text within its bounding box; the default is text left. This sets the default for the format module.

undtrue—a Boolean expression that indicates whether to underline text. The default is "False". This sets the default for the format module.

spacevalue—a value specifying the horizontal spacing of text, in the current units. This forces fixed spacing of any font. This sets the default for the format module.

leadvalue—a value specifying vertical line spacing for printing arrays, in the current units. This sets the default for the format module.

numlines—the number of elements to print from arrays, beginning with the first element. The default is all elements of the array. This sets the default for the format module.

density—the width of bar code narrow bars in the current units. This value is ignored for bar code types which do not vary in size. This sets the default for the format module.

ratio—the width of bar code wide bar, in terms of the narrow bar width. Valid values are 2.0 (for a 2.0 to 1 ratio), 2.5 , and 3.0. This value is ignored for bar code types which do not vary in size. This sets the default for the format module.

style—a keyword indicating a bar code style (a variation within the type). This sets the default for the format module.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

Note: The format module must end with the END FORMAT command if it started with FORMAT, or the END FORM command if it started with FORM.

See also: ALIGN, BORDER, DENSITY, DEVICE, DRAW, DRAW TEXT, END, FEEDER, FONT, HEIGHT, LEADING, LINECAP, LINESTYLE, OPAQUE, PAPER, PATTERN, POINTS, RADIUS, RATIO, RESET MACRO, REVERSE, ROTATION, SET MACRO, SHADOW, SPACING, STACKER, STYLE, SYMBOLS, THICKNESS, TYPEFACE, UNDERLINE, UNITS, WIDTH

FORMFEEDS

This clause is used with the command below. See the entry for that command.

See: CHANNEL

FROMHEX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a hexadecimal number to decimal.

Syntax: `FROMHEX(number)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a hexadecimal value

Examples:

```
LET sum = FROMHEX(hexsum)

LET value = FROMHEX(FF75AB3)

LET amts[2:7] = FROMHEX(hexamounts[2:7] * 2)
```

See also: TOHEX

FUNCTION

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Module command

Purpose: Defines a function module. A function module uses DCL commands to perform a specific function, accepting parameters and returning values.

Syntax: `FUNCTION "functionname" [ACCEPTS param1 [,param2 ...]]`

Parameters: *functionname*—a name for the function.

param1, *param2*—variable names for the parameter values that are passed into the function. The values passed in are assigned to these variables, which are used within the function module. A function may have any number of parameters.

Note: The function module must end with the END FUNCTION command.

See also: CALL, END, RETURN

GLOBAL

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Creates variables in the global worksheet.

Syntax: `GLOBAL variable1 = initvalue1 [, variable2 = initvalue2 ...]`

Parameters: *variable1*, *variable2*—names for the new variables. A GLOBAL statement may create any number of variables.

initvalue1, *initvalue2*—expressions indicating initial values for the variables.

See also: AUTO, CLEAN WORKSHEET, DESTROY, PRIVATE, PUBLIC, SET GLOBAL

2

This clause is used with the command below. See the entry for that command.

See: CLEAN, SET GLOBAL

GLYPHS

This clause is used with the command below. See the entry for that command.

See: FONT

HEIGHT

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the height of an object.

Affects: bar codes, boxes, circles, combs, ellipses, image bounding boxes, object bounding boxes, and text bounding boxes

Syntax: `HEIGHT` *`height`*

Default: 0 for bar codes, boxes, combs, ellipses, and circles;
the image height for image bounding boxes and object bounding boxes;
the font height for text bounding boxes

Parameters: *height*—a numeric expression indicating the height of the object, in the current units.

See also: DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, SET, UNITS, WIDTH

HOSTNAME

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Returns the name of the host system.

Syntax: `HOSTNAME()`

Examples:

```
LET host = HOSTNAME()

LET string = "Host: " & HOSTNAME()
```

See also: CPUNAME, OSCPUID, OSNAME, PLATFORM, USERNAME

IF

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Executes a set of commands based on conditional tests. If the condition is true, the set of commands between the THEN clause and the next ELSE clause, ELIF clause, or END IF command are executed in order.

If the condition is false, the commands between the THEN clause and the next ELSE clause, ELIF clause, or END IF command are skipped. If an ELSE clause is next, the commands between the ELSE clause and the next END IF command are executed.

The ELIF clause is a combination of the ELSE clause and the IF command. If the previous IF or ELIF condition is false, the ELIF condition is tested, and processing continues just as it does for an IF condition.

Syntax:

```
IF ifcondition THEN ifcommands
[ELSE elsecommands |
{ELIF elifcondition THEN elifcommands} ... [ELSE elsecommands]]
END IF
```

Parameters: *ifcondition*—a Boolean expression specifying the condition for executing the *ifcommands*.

ifcommands—the commands that are executed if the *ifcondition* is true. The commands must be valid for use in the current module.

elsecommands—the commands that are executed if the previous *ifcondition* or *elifcondition* is false. The commands must be valid for use in the current module.

elifcondition—a Boolean expression specifying the condition for executing the *elifcommands*. An IF statement may have any number of ELIF clauses.

elifcommands—the commands that are executed if the previous *elifcondition* is true. The commands must be valid for use in the current module.

Note: The IF statement must end with the END IF command.

See also: END, SWITCH, WHILE

IGNORE WORKSHEET

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Removes a global, public, or user-created worksheet from the worksheet stack.

Syntax: `IGNORE WORKSHEET "sheetname"`

Parameters: *sheetname*—the name of a global, public, or user-created worksheet. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

See also: CREATE WORKSHEET, DESTROY WORKSHEET, SEARCH WORKSHEET, SET GLOBAL, SET PUBLIC

IMPORT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the value of a system parameter.

Syntax: `IMPORT(sysparameter)`

Parameters: *sysparameter*— a literal string or variable containing the name of a system parameter

Example:
```
LET homedir = IMPORT("HOME")
```

See also: EXPORT, PARAMETER, TRANSMIT

INPUT

This keyword is part of the commands below. See the entries for those commands.

See: ALTER CHANNEL, REWIND, SET INPUT

INVERT

This clause is the same as the clause REVERSE. See page 1-142.

ISALNUM

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns TRUE if a variable or string contains only alphabetic and numeric characters, including periods (.) and commas (,); returns FALSE if it contains any other characters (or spaces). If the variable is an array, returns TRUE if all elements in the array contain only alphabetic and numeric characters; returns FALSE if not.

Syntax: `ISALNUM(`*`expression`*`)`

Parameters: *expression*— a constant, literal string, variable, or expression

Examples:

```
WHILE ISALNUM(itemnumber)

LET done = ISALNUM(itemcodes[1:5])

IF NOT ISALNUM("V" & vendornum & itemnums[n]) THEN
```

See also: ISALPHA, ISDATE, ISDIGITS, ISMONEY, ISNUMERIC

ISALPHA

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if all characters in a string are alphabetic; returns FALSE if it contains any other characters (or spaces). If the string parameter is an array, returns TRUE if all elements in the array contain only alphabetic characters; returns FALSE if not.

Syntax: `ISALPHA(string)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:

```
WHILE ISALPHA(name)

LET done = ISALPHA(itemcodes[1:5])

IF NOT ISALPHA("V" & vendorcodes[n] & "I" & itemcodes[n]) THEN
```

See also: ISALNUM, ISDATE, ISDIGITS, ISMONEY, ISNUMERIC

ISBLANK

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if all characters in a string are spaces; returns FALSE if it contains any other characters. If the string parameter is an array, returns TRUE if all elements in the array contain only spaces; returns FALSE if not.

Syntax: `ISBLANK(string)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET done = ISBLANK(@[3,1:8])

WHILE ISBLANK(itemcodes[1:5])

IF NOT ISBLANK(vendorcodes[n] & itemcodes[n]) THEN
```

See also: EMPTY, ISALNUM, SQUISH

ISDATE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if all characters in a string are valid date characters, specifically digits, hyphens (-), and slashes (/); returns FALSE if it contains any other characters.

Syntax: `ISDATE(`*`string`*`)`

Parameters: *string*— a literal string, a variable containing a string, or an expression that evaluates to a string

Examples:
```
LET done = ISDATE(@[3,1:8])

WHILE ISDATE(msgblock[4])

IF NOT ISDATE(a & "/" & b & "/96") THEN
```

See also: ISALNUM, ISALPHA, ISDIGITS, ISMONEY, ISNUMERIC

ISDIGITS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if a variable or string contains only numeric digits; returns FALSE if it contains any other characters (or spaces). If the variable is an array, returns TRUE if all elements in the array contain only numeric digits; returns FALSE if not.

Syntax: `ISDIGITS(`*`expression`*`)`

Parameters: *expression*— a constant, a literal string, a variable, or an expression

Examples:

```
LET done = ISDIGITS(@[3,1:8])

WHILE ISDIGITS(itemcodes[1:5])

IF NOT ISDIGITS(vendornum[n] & itemnum[n]) THEN
```

See also: ISALNUM, ISALPHA, ISDATE, ISMONEY, ISNUMERIC

ISMONEY

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if all characters in a string are valid characters for monetary values, specifically digits, hyphens (-), commas(,), and dollar signs ($); returns FALSE if it contains any other characters. If the string parameter is an array, returns TRUE if all elements in the array contain only valid characters for monetary values; returns FALSE if not.

Syntax: ISMONEY(*string*)

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:

```
LET done = ISMONEY(@[11:55,40:49])

WHILE ISMONEY(prices[1:max])

IF NOT ISMONEY("$ " & total) THEN
```

See also: ISALNUM, ISALPHA, ISDATE, ISDIGITS, ISNUMERIC

ISNUMERIC

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if a variable or string contains only numeric characters, including periods (.) and commas (,); returns FALSE if it contains any other characters (or

spaces). If the variable is an array, returns TRUE if all elements in the array contain only numeric characters; returns FALSE if not.

Syntax: `ISNUMERIC(`*`expression`*`)`

Parameters: *expression*— a constant, literal string, variable, or expression

Examples:
```
LET done = ISNUMERIC(@[1:5,10:55])

WHILE ISNUMERIC(quantities[1:5])

IF NOT ISNUMERIC(vendornum & itemnums[n]) THEN
```

See also: ISALNUM, ISALPHA, ISDATE, ISDIGITS, ISMONEY

JUSTIFY

This clause is used with the command below. See the entry for that command.

See: DRAW TEXT

LANGUAGE

This clause is used with the command below. See the entry for that command.

See: DEVICE

LAYOUT

This clause is used with the command below. See the entry for that command.

See: FORMAT

LCOLOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Selects a color for the lines or border of an object. For more information, see Chapter 2 of the *Advanced Features Guide*.

Affects: arcs, bar code bounding boxes, boxes, circles, combs, curves, ellipses, image bounding boxes, lines, object bounding boxes, and text bounding boxes

Syntax: `LCOLOR linecolor`

Default: black

Parameters: *linecolor*— a literal string or an expression that evaluates to a string, containing the name of a color defined in the current palette module. The palette module is selected in the device module. The default line color can be set using the SET LCOLOR command.

See also: BORDER, COLOR, DEVICE, DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, FCOLOR, PALETTE, SCOLOR, SET

LEADING

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the line spacing for printing multiple lines of text from an array.

Affects: text

Syntax: `LEADING leadvalue`

Default: the standard line spacing value of the current font

Parameters: *leadvalue*—a leading value, which is the distance between the baselines of consecutive lines of text, in the current units. To set the leading value to the default for the font, specify a value of -1.

See also: DRAW TEXT, SET, SPACING, UNITS

LENGTH

1

This clause is the same as the LINES clause, used with the commands below. See the entries for those commands.

See: DATAMAP, REGION

2

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Finds the length of each element in an array of strings, returns an array of the length values.

Syntax: `LENGTH(array)`

Parameters: *array*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:

```
LET lastnames = LENGTH(lastnames)

FOR i = 1 TO LENGTH(@[1:5])

LET itemlengths = LENGTH(itemdesc[1:numitems])
```

See also: COLUMNS, FIX, ROWS, SPAN

LET

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Assigns a value to a variable. This may be used to assign input data to variables, to manipulate strings of data, and to perform calculations.

The LET command is also used with DCL functions, which cannot stand alone, but must be used within another command. The LET command assigns the return value of the function to a variable.

Syntax: LET *varname* = *value*

Parameters: *varname*—the name of a variable.

value—an expression indicating the value to assign to the variable.

LIKE

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, DEVICE, FONT, FONTPACK

LINE

This clause is used with the command below. See the entry for that command.

See: REGION

See also: ADVANCE LINE, DRAW LINE, LINES, NEXT LINE, REPEAT LINE

LINECAP

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the appearance of the cap on the ends of lines. A butted cap ends the line exactly at the specified coordinates, with a square shape. The squared, rounded, or triangular cap extends the line 1/2 the line thickness past the coordinates, with the indicated shape.

Affects: simple arcs, curves, and lines

Syntax: LINECAP "*capname*"

Default: "BUTTED"

Parameters: *capname*—a keyword indicating a line cap style, as listed in the table below.

Linecap	Keywords
default	D DEFAULT
butted	B BUTTED
squared	S SQUARED
rounded	R ROUNDED
triangular	T TRIANGULAR

See also: DRAW ARC, DRAW CURVE, DRAW LINE, SET, THICKNESS

LINEFEEDS

This clause is the same as the clause NEWLINES, used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

LINES

1

This clause is used with the commands listed below. See the entries for those commands.

See: DATAMAP, DRAW COMB, DRAW TEXT, REGION, SET

2

This clause is the same as the clause ROWS, used with the commands listed below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL, SQL

3

This function is the same as the function ROWS. See page 1-145.

LINESTYLE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the appearance of a line.

Affects: arcs, boxes, circles, comb borders, curves, ellipses, and lines

Syntax: `LINESTYLE "stylename"`

Default: "SOLID"

Parameters: *stylename*—a keyword indicating a line style, as listed in the table below. The fixed 7, fixed 8, and scaled styles consist of dashes of various lengths. With a scaled style, the lengths of the dashes and dots are scaled according to the line thickness.

Linestyle	Keywords		
default	D	DEFAULT	
solid	S	SOLID	
dash	F1	FIXED1	Dash
dot	F2	FIXED2	Dot
dash-dot	F3	FIXED3	DashDot
dash-dot-dot	F4	FIXED4	DashDotDot
long dash	F5	FIXED5	LongDash
dot-dot	F6	FIXED6	DotDot
fixed 7	F7	FIXED7	
fixed 8	F8	FIXED8	
scaled 1	SCALED1	A1	ADAPT1
scaled 2	SCALED2	A2	ADAPT2
scaled 3	SCALED3	A3	ADAPT3
scaled 4	SCALED4	A4	ADAPT4
scaled 5	SCALED5	A5	ADAPT5
scaled 6	SCALED6	A6	ADAPT6
scaled 7	SCALED7	A7	ADAPT7
scaled 8	SCALED8	A8	ADAPT8

See also: DRAW ARC, DRAW BOX, DRAW CURVE, DRAW ELLIPSE, DRAW LINE, SET

LINK

This clause is used with the command below. See the entry for that command.

See: DRAW TEXT

LOAD

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Copies a DCL file into memory. This is useful when a document uses a module that is in a separate file, so that the module can be found quickly.

Syntax: `LOAD "filename"`

Parameters: *filename*—the name of the DCL file.

LOG

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

1

Type: Command

Purpose: Writes a message to the current log file.

Syntax: `LOG "message1" [, "message2"]`

Parameters: *message1*, *message2*—literal strings containing messages to be logged. A LOG FILE statement may have any number of messages.

See also: SET LOG FILE

2

This clause is used with the command below. See the entry for that command.

See: SET LOG FILE

LOGFILE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the name of the log file.

Syntax: `LOGFILE()`

Examples:
```
LET filename = LOGFILE()

LET string = "Log: " & LOGFILE()
```

See also: SET LOG FILE, LOG

LOOKUP

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Finds the position of the key value in the key array. Returns an array of the values that are in the corresponding positions in the target arrays. If a target array does not include a corresponding position, an empty string, 0 value, or FALSE is returned.

Syntax: `LOOKUP(`*`keyvalue`*`, `*`keyarray`*`, `*`target1`*` [, `*`target2`*`…])`

Parameters: *keyvalue*—a literal string or value, a variable, or an expression, containing the value to be found in the key array. The value may be a string, a numeric value, or a boolean value.

keyarray—an array containing the key values. The values may be strings, numeric values, or boolean values.

target1, *target2*—arrays containing the target values. One value is returned from each of these arrays, in the order that the arrays are specified. You may specify any number of target arrays.

Examples:
```
LET townname = LOOKUP("30096",zipcodes,townnames)

LET emplinfo = LOOKUP(num,empnums,empnames,titles,hiredates,exts)
```

See also: FIND, IF, IMPORT, PARAMETER, SORT, SWITCH, WHERE

MACRO

This clause is used with the commands below. See the entries for those commands.

See: FORMAT, RESET MACRO, SEGMENT, SET MACRO

MACROS

This clause is used with the commands below. See the entries for those commands.

See: DEVICE, FORMAT

MAIL

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ □ □ □ □ □ □ □

Type: Command

Purpose: Sets e-mail message settings to the specified values. This command must be placed after the SET OUTPUT command that selects an e-mail channel.

Syntax:

For SMTP E-mail

```
MAIL TO tostring1 [,tostring2...] FROM fromstring
[REPLY replystring] [CC ccstring1 [,ccstring2...]]
[BCC bccstring1 [,bccstring2...]] [SUBJECT subjstring]
[NOTES notestring1 [,notestring2...]]
[SIGNATURE sigstring1 [,sigstring2...]]
[FILE filename] [MIME datatype]
[ATTACH attfile1 [,attfile2...]]
[SERVER srvrname PORT portnum]
```

For MAPI E-mail (for Windows NT systems only)

```
MAIL TO tostring1 [,tostring2...] PROFILE profilename
[REPLY replystring] [CC ccstring1 [,ccstring2...]]
[BCC bccstring1 [,bccstring2...]] [SUBJECT subjstring] [NOTES
notestring1 [,notestring2...]]
[FILE filename] [FOLDER folderpath FROM fromstring] [SENSITIVITY
```

"*senslevel*"] [PRIORITY "*priorlevel*"]
[EXPIRES *expdate*] [ACCESSKEY *keyname*]

Parameters: *tostring1*, *tostring2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the e-mail addresses of the message recipients. For MAPI mail, you can specify the beginning of an address to have all available address books searched for a match; the first matching address is used; if multiple matches exist in a single address book, the message is not sent and a warning message is generated. MAPI mail is valid only for Windows NT systems.

fromstring—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the e-mail address of the message sender. For MAPI mail, this can be a name or an e-mail address. MAPI mail is valid only for Windows NT systems.

profilename—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of a MAPI profile that exists on the computer running OptioDCS. You create profiles using a MAPI e-mail program, such as Microsoft Outlook. This is valid only for Windows NT systems.

replystring—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the e-mail address that should be used for replies to the message.

ccstring1, *ccstring2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the e-mail addresses of the message copy recipients. For MAPI mail, you can specify the beginning of an address to have all available address books searched for a match; the first matching address is used; if multiple matches exist in a single address book, the partial address is ignored and a warning message is generated. MAPI mail is valid only for Windows NT systems.

bccstring1, *bccstring2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the e-mail addresses of the message blind copy recipients. The addresses of blind copy recipients are not included in the message to other recipients. For MAPI mail, you can specify the beginning of an address to have all available address books searched for a match; the first matching address is used; if multiple matches exist in a single address book, the partial address is ignored and a warning message is generated. MAPI mail is valid only for Windows NT systems.

subjstring—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the message subject.

notestring1, *notestring2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the message.

sigstring1, *sigstring2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the sender's signature lines.

filename—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name for the attached document that is displayed in the message.

folderpath—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path of a public folder such as "\Accounting\Reports" A copy of the message is saved in this public folder on your MAPI mail server. This is valid only for Windows NT systems.

senslevel—a keyword representing the message sensitivity. Valid values are Normal (default), Personal, Private, and Confidential. This may affect the message controls, appear in the message header, or neither, depending on the recipient's mail program. This is valid only for Windows NT systems.

priorlevel—a keyword representing the message priority. Valid values are Low, Normal (default), and High. This may affect the message controls, appear in the message header, or neither, depending on the recipient's mail program. This is valid only for Windows NT systems.

expdate—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the message expiration date and time, in the form *yyyy*/*mm*/*dd*. The expiration date may affect the message controls, appear in the message header, or neither, depending on the recipient's mail program. This is valid only for Windows NT systems.

datatype—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the data type of the attachment, such as "application/rtf" or "application/postscript". This is used by the recipient's mail program to determine what viewer to use for the attachment.

attfile1, *attfile2*—literal strings, variables containing strings, or expressions that evaluate to strings, containing the paths and names of attachment files other than the OptioDCS document. Each file is MIME-encoded and attached to the SMTP mail message to be sent.

srvrname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the SMTP mail server used to send the message. This overrides the server specified in the output channel module.

portnum—a constant, variable, or expression, that evaluates to a numerical value, indicating the port of the SMTP mail server used to send the message. This overrides the port number specified in the output channel module.

keyname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key used by the client system to log on to the OptioDCS server. This overrides the access key specified in the output channel module. For more information on access keys, see Chapter 2 of the *User's Guide*. This is valid only for Windows NT systems.

See also: BREAK ON, MAIL RESET, SERVER, SET OUTPUT, SHOWDATE

MAIL RESET

Doc ■ Par ■ Fun ■ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Resets e-mail message settings to default values (empty strings).

Syntax: `MAIL RESET`

See also: BREAK ON, MAIL

MAP

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Calls a datamap module or region module for processing. A data map module or region module specifies to OptioDCS how to read the input data and assign it to variables.

Syntax: `MAP "`*`mapname`*`" [WHEN `*`condition`*`]`
`[WITH `*`param1`* `[,`*`param2...`*`]] [RETURNING `*`retval1`* `[,`*`retval2...`*`]]`

Parameters: *mapname*—the name of the datamap module or region module; if the module is in a separate file, include the path of the file.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed. The WHEN clause must be before any WITH or RETURNING clauses.

param1, *param2*—expressions to be passed into the data map or region as parameter values. The number of WITH values should match the number of parameters the module accepts, and they should be in the same order as defined by the ACCEPTS clause of the DATAMAP or REGION statement.

retval1, *retval2*—variable names for the return values that are received from the data map or region. The number of RETURNING values should match the number of values the module returns, and they should be in the same order as defined by the RETURN statement within the data map or region module.

See also: DATAMAP, REGION

MATCHES

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns TRUE if two strings match exactly; returns FALSE if not.

Syntax: `MATCHES(string1,string2)`

Parameters: *string1*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

string2— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:

```
LET more = MATCHES(@[55,1:5],moreflag)

IF MATCHES(@[2,70:75],"Page " & pagenum) THEN

WHILE MATCHES(names[i],info[i,1:length])
```

See also: FIND, LENGTH, SPAN, PRUNE, TOLOWER, TOUPPER

MAX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Finds the maximum value of the values in an array.

Syntax: `MAX(array)`

Parameters: *array*— an array variable containing numeric values, or strings of only numeric characters

Examples:
```
LET most = MAX(quantites)

IF MAX(prices[2:7]) > 250.00 THEN
```

See also: AVG, LENGTH, MIN

MESSAGEFILE

This function is the same as the function LOGFILE. See page 1-111.

METRICS

This clause is used with the command below. See the entry for that command.

See: FONT

MIN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Finds the minimum value of the values in an array.

Syntax: `MIN(array)`

Parameters: *array*— an array variable containing numeric values, or strings of only numeric characters

Examples:
```
LET least = MIN(quantites)

IF MIN(prices[2:7]) < 7.99 THEN
```

See also: AVG, LENGTH, MAX

MODE

This clause is used with the command below. See the entry for that command.

See: CHANNEL

MONOSPACED

This clause is used with the command below. See the entry for that command.

See: FONT

NEWLINES

This clause is used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

NEXT LINE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Causes region or datamap processing to start again at the first command, and moves the current scan line down one line. The current scan line is a marker which indicates a certain line in the input data; it is used to refer to locations of data on the input page. When the current scan line is moved outside the region (or datamap), the module is exited.

Syntax: `NEXT LINE`

See also: ADVANCE LINE, REPEAT LINE

NEXT ROW

This command is the same as the command NEXT LINE. See page 1-117.

NOW

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the current time, as the number of seconds since 00:00:00 Greenwich mean time (GMT), January 1, 1970.

Syntax: `NOW()`

Examples:
```
LET printdate = NOW()

IF NOW() < 870000000 THEN
```

See also: DATE, SHOWDATE, TIME

NUM

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Converts a number to a string, formatted with the specified decimal separator character, thousands separator character, leading or trailing currency symbol, and number of digits following the decimal.

Syntax: `NUM(`*`number, places, thousands, decimal, currency, prefix`*`)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

places— the number of digits that should follow the decimal string. The number is rounded if necessary. The default is 0.

thousands— a character string used to separate each set of three digits before the decimal string. For example, in North America this is usually a comma. The default is an empty string.

decimal— a variable containing a string or strings, or an expression that evaluates to a string, containing the characters to place between the whole number portion of the string from the fraction portion. For example, in North America this is usually a period; in Europe, it is usually a comma. The default is a period.

currency— a variable containing a string or strings, or an expression that evaluates to a string, containing the characters that should begin or end the resulting string, including any spaces. The default is an empty string.

prefix— a Boolean expression that indicates whether the currency string should begin the resulting string. If this is FALSE, the currency string ends the resulting string. The default is TRUE.

Examples:

```
LET amdollars = NUM(number, 2, ",", ".", "$")

LET francstring = NUM(number, 2, " ", ",", " FF", FALSE)
```

See also: ISALNUM, ISDIGITS, ISNUMERIC, PICTURE, ROUND, TONUMBER, TOSTRING, TOWORDS

OPAQUE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Determines whether an object is opaque or transparent.

Note: An object with a shadow is always opaque. See the SHADOW entry.

Affects: arcs, bar codes, boxes, circles, combs, ellipses, images, objects, and text

Syntax: `OPAQUE` *opaqueon*

Default: Off

Parameters: *opaqueon*—a Boolean expression that indicates whether the object should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent.

See also: DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, SET, SHADOW

OPEN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Opens a channel. OPEN INPUT opens the current input channel, OPEN OUTPUT opens the current output channel, or a channel or SQL module can be specified.

An input channel is opened automatically by the MAP command, and an output channel is opened automatically by the DRAW command. All channels are closed automatically when document processing is complete, and any channel may be closed explicitly using the CLOSE command.

Syntax: `OPEN {INPUT | OUTPUT | "channelname"}`

Parameters: *channelname*—the name of a channel or SQL module.

See also: CHANNEL, CLOSE, FETCH, SET INPUT, SET OUTPUT, SQL, SQLRUN

OSCPUID

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the CPU ID of the operating system.

Syntax: `OSCPUID()`

Examples:
```
LET opsys = OSCPUID()

LET string = "Op sys: " & OSCPUID()
```

See also: CPUNAME, HOSTNAME, OSNAME, PLATFORM, USERNAME

OSNAME

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the name of the operating system.

Syntax: `OSNAME()`

Examples: `LET os = OSNAME()`

`LET string = "Op sys: " & OSNAME()`

See also: CPUNAME, HOSTNAME, OSCPUID, PLATFORM, USERNAME

OTHERWISE

This clause is used with the command below. See the entry for that command.

See: SWITCH

OUTPUT

This keyword is part of the commands below. See the entries for those commands.

See: ALTER CHANNEL, SET OUTPUT

OVERLAY

This clause is used with the command below. See the entry for that command.

See: REGION

PAGE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the page number of the current page of data in an input channel. The current page is the page that was read most recently.

Syntax: PAGE("*channelname*")

Parameters: *channelname*—the name of an input channel.

Examples:
```
LET pagenum = PAGE("stdin")

IF PAGE("stdin") > 8 THEN
```

See also: PAGES, READ PAGE, REWIND

PAGES

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the number of pages of data in an input channel, and advances the channel to the end of the data. After using this function, use the REWIND command to move to the beginning of the file, or the READ PAGE command to move to a specific page.

If the input data is being piped directly to OptioDCS for UNIX from your application as a data stream, pipeline, or socket, you cannot rewind the data unless the input channel includes the clause SEARCHABLE TRUE.

Syntax: PAGES("*channelname*")

Parameters: *channelname*—the name of an input channel.

Examples:
```
LET numpages = PAGES("stdin")

IF PAGES("stdin") > 30 THEN
```

See also: PAGE, READ PAGE, REWIND

PALETTE

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ □ □ □ □ □

1

Type: Module command

Purpose: Defines a palette module. A palette module contains color definitions describing a group of colors for use with a device. For more information, see Chapter 2 of the *Advanced Features Guide*.

Syntax: PALETTE "*palettename*" TYPE "RGB"

Parameters: *palettename*—a name for the palette.

TYPE "RGB"—a keyword indicating the type of palette.
RGB palette colors are defined by red, green, and blue light color values.

Note: The palette module must end with the END PALETTE command.

See also: END

2

This clause is used with the command below. See the entry for that command.

See: DEVICE

PAPER

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ □ □ □ ■ □

1

Type: Command

Purpose: Defines a keyword for a paper size available on a device.

Syntax: PAPER "*paperword*","*papercode*",*length*,*width*,
offset_v,*offset_h*,*margin_v*,*margin_h*

Parameters: *paperword*—a keyword for the paper size.

papercode—the code that indicates the corresponding paper size to the printer.

length—the length of the paper size in the current units.

width—the width of the paper size in the current units.

offset_v—the vertical offset of the logical origin (0,0) from physical top edge of the paper.

offset_h—the horizontal offset of the logical origin (0,0) from physical left edge of the paper.

margin_v—the width of the vertical unprintable region, in the current units.

margin_h—the width of the horizontal unprintable region, in the current units.

See also: FORMAT

2

This clause is used in a different mode with the command below. See the entry for that command.

See: FORMAT

PARAMETER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the value of a variable from the odcs.ini file, the value of a variable from the command line, the name of a file from the command line, or the number of files on the command line.

To return the value of a variable from the odcs.ini file, use the section name as the first parameter and the variable name as the second. For more information on the odcs.ini file, see the file or the *User's Guid*e (Chapter 7 for Unix, Chapter 5 for Windows NT).

To return the value of a variable from the command line, use "-" as the first parameter and the variable name as the second. Variables are created using the -p option with the ocserver command.

To return the name of a file from the command line, use "*" as the first parameter and the sequential number of the file as the second. For example, using 1 as the second parameter returns the name of the first file on the command line.

To return the number of files on the command line, use "*" as the first parameter and "0" as the second.

Syntax: PARAMETER("*section*","*inivariable*")
PARAMETER("-","*variable*")
PARAMETER("*",*number*)
PARAMETER("*","0")

Parameters: *section*— the name of a section in the odcs.ini file

inivariable— the name of a variable in the odcs.ini file

variable— the name of a variable for which a value is passed in from the command line

number— the sequential number of a file name on the command line, i.e., 1 for the first, 2 for the second, etc.

Examples:
```
LET swversion = PARAMETER("User Defined Values","Version")

FOR i = 1 TO PARAMETER("-","COPIES")

LET thirdfile = PARAMETER("*",3)

LET numfiles = PARAMETER("*","0")

FOR i = 1 TO numfiles
   LET files[i] = PARAMETER("*",i)
END FOR
```

See also: EXPORT, IMPORT, SYSTEM, TRANSMIT

PART

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Module command

Purpose: Defines a part module. A part module controls the processing of one part of a document.

Syntax: PART "*partname*" [ACCEPTS *param1* [,*param2* ...]]
[EXTENSION *"PARAMETER=value"*]

Parameters: *partname*—a name for the part module.

param1, *param2*—variable names for the parameter values that are passed into the part. The values passed in are assigned to these variables, which are used within the part module. A part may have any number of parameters.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

Note: The part module must end with the END PART command.

See also: END, PROCESS, PROCESS PARTS, SET SEQUENCE

PARTS

This clause is used with the command below. See the entry for that command.

See: PROCESS PARTS

PASSWORD

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, FAX

PATCH

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the patch number of the OptioDCS software. For example, if your software is version 2.11.05, the version number is 2, the release number is 11, and the patch number is 05.

Syntax: `PATCH()`

Examples:

```
LET patchnum = PATCH()

IF PATCH() > 8 THEN
```

See also: RELEASE, VERSION, SERIAL

PATTERN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the fill pattern of an object.

Affects: chord and pie arcs, boxes, circles, combs, ellipses, text bounding boxes, and object shadows

Syntax: `PATTERN "patname"`

Default: "CLEAR"

Parameters: *patname*—a keyword indicating a fill pattern, as listed in the table below.

Pattern	Keywords			
default	D	DEFAULT		
clear	C	CLEAR	NONE	
solid (user defined)	U	USER	USER DEFINED	USERDEFINED
crosshatch	HC	CROSS HATCH 1	HCROSSHATCH	HORIZONTALCROSSHATCH
diagonal crosshatch	DC	CROSS HATCH 2	DCROSSHATCH	DIAGONALCROSSHATCH TRELLIS TR
horizontal stripes	HS	STRIPES 1	HSTRIPES	HORIZONTALSTRIPES
vertical stripes	VS	STRIPES 2	VSTRIPES	VERTICALSTRIPES
top-to-bottom diagonal	TS	STRIPES 3	TSTRIPES	TOPSTRIPES
bottom-to-top diagonal	BS	STRIPES 4	BSTRIPES	BOTTOMSTRIPES
white	W	WHITE	G0	GREY0
1% - 2% grey	G1	GREY1		
3% - 10% grey	G2	GREY2		
11% - 20% grey	G3	GREY3		
21% - 35% grey	G4	GREY4		
36% - 55% grey	G5	GREY5		
56% - 80% grey	G6	GREY6		
81% - 99% grey	G7	GREY7		
black	B	BLACK	G8	GREY8

See also: DRAW ARC, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, FCOLOR, SHADOW, SET

PICTURE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a number to a string, formatted according to a specified pattern, using the specified thousands separator character, decimal separator character, currency symbol, and fill character. If the number does not fit the pattern, the string contains a number sign (#) for each digit. For more information and examples, see Chapter 2 of the *Advanced Features Guide*.

Syntax: `PICTURE(pattern, number, thousands, decimal, currency, fill)`

Parameters: *pattern*— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the pattern for the resulting string. The pattern can contain any valid ASCII character. The characters listed below indicate how to format the number; any other characters are copied literally to the resulting string:

Character		Pattern Usage
number sign	#	for a digit of the number If the number has too few digits after the decimal, it is padded with zeros; if it has too few digits before decimal, it is padded with spaces. If the number has too many digits after the decimal, it is rounded to the number of number signs; if the number has too many digits before decimal, all digits are replaced with number signs to indicate an overflow.
dollar sign	$	The first dollar sign is for the currency symbol. Subsequent dollar signs are for digits of the number, and the currency symbol is placed immediately to the left of the first digit. If the number has too few digits after the decimal, it is padded with zeros; if it has too few digits before decimal, it is padded with spaces before the currency symbol. If there is no whole number, it is padded with a zero between the currency symbol and the decimal. If the number has too many digits after the decimal, it is rounded to the number of dollar signs; if the number has too many digits before decimal, all digits are replaced with number signsto indicate an overflow.
asterisk	*	for a digit of the number If the number has too few digits after the decimal, it is padded with zeros; if it has too few digits before decimal, it is padded

		with the fill character. If the number has too many digits after the decimal, it is rounded to the number of number signs; if the number has too many digits before decimal, all digits are replaced with number signsto indicate an overflow.
comma	,	for the thousands separator string If the character to the left of the comma is replaced by a currency symbol or sign character, the thousands separator is omitted and the currency symbol or sign character is moved to its place. If the comma follows an asterisk that is replaced by a fill character, the comma is also replaced by a fill character.
period	.	for the decimal separator string
plus sign	+	for the sign of the number; a minus sign if negative, a plus sign if positive If the plus sign is before the decimal, the sign character is placed before the first digit or currency symbol (after any padding spaces). If the plus sign is after the decimal, any sign character before the decimal is omitted.
minus sign	-	for the sign of the number; a minus sign if negative, a space if positive If the minus sign is before the decimal, the sign character is placed before the first digit or currency symbol (after any padding spaces). If the minus sign is after the decimal, any sign character before the decimal is omitted.
opening parenthesis	(	for the sign of the number; an opening parenthesis for a negative number, a space for a positive number
closing parenthesis	)	for the sign of the number; a closing parenthesis for a negative number, a space for a positive number

number— a constant, variable, or expression, that evaluates to a numerical value

thousands— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the character used to separate each set of three digits before the decimal; used to replace the comma in the pattern string, if the character to the left of the comma is replaced by a digit or number sign. The default is a comma.

decimal— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the characters to place between the whole number portion of the string from the fraction portion; used to replace the period in the pattern string. The default is a period.

currency— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the characters that should replace the first dollar sign in the pattern string. The default is a dollar sign.

fill— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the character that should be used to pad the string if the number has too few digits to the left of the decimal, used to replace the asterisk in the pattern string. If a comma follows an asterisk that is replaced by a fill character, the comma is also replaced by a fill character. The default is an asterisk.

Examples:

```
LET amdollars = PICTURE("$$,$$$.##-", number)

LET francstring =
    PICTURE("+**,***.##$", number, " ", ",", " FF", "0")
```

See also: ISALNUM, ISDIGITS, ISNUMERIC, NUM, TONUMBER, TOSTRING, TOWORDS

PIVOT

This clause is used with the command below. See the entry for that command.

See: DRAW ARC

PLACES

This function is the same as the function NUM. See page 1-119.

PLATFORM

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the name of the platform on which you are running the OptioDCS software.

Syntax:

```
PLATFORM()
```

Examples: `LET runplat = PLATFORM()`

`LET string = "Platform: " & PLATFORM()`

See also: CPUNAME, HOSTNAME, OSCPUID, OSNAME, USERNAME

POINTS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the size of a font

Affects: text

Syntax: `POINTS "pointsize"`

Default: none

Parameters: *pointsize*—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. For PostScript printer drivers, an optional horizontal point size is supported; for example POINTS "12,10" specifies a 12 point height and a 10 point width.

See also: DRAW TEXT, FONT, SET

PORT

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, FAX, MAIL

PRINTER

This clause is used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

PRIVATE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

1

Type: Command

Purpose: Creates variables in the private worksheet for the current module.

Syntax:
```
PRIVATE variable1 = initvalue1
[, variable2 = initvalue2 ...]
```

Parameters: *variable1*, *variable2*—names for the new variables. A PRIVATE statement may create any number of variables.

initvalue1, *initvalue2*—expressions indicating initial values for the variables, which are assigned to the variables the first time the module is executed.

See also: AUTO, CLEAN WORKSHEET, DESTROY, GLOBAL, PUBLIC

2

This clause is used with the command below. See the entry for that command.

See: CLEAN

PROCESS

Doc ■ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Calls a part module, executing the commands in the module in order. A part module controls the processing of one part of a document.

Syntax: PROCESS "*partname*" [WHEN *condition*]
[WITH *param1* [,*param2*...]] [RETURNING *retval1* [,*retval2*...]]

Parameters: *partname*—the name of the part module; if the module is in a separate file, include the path of the file.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed. The WHEN clause must be before any WITH or RETURNING clauses.

param1, *param2*—expressions to be passed into the part as parameter values. The number of WITH values should match the number of parameters the part accepts, and they should be in the same order as defined by the ACCEPTS clause of the PART statement.

retval1, *retval2*—variable names for the return values that are received from the part. The number of RETURNING values should match the number of values the part returns, and they should be in the same order as defined by the RETURN statement within the part module.

See also: PART, PROCESS PARTS, READ

PROCESS PARTS

Doc ■ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Processes the part modules of a document in the order specified by the SET SEQUENCE command. Part processing consists of a loop of reading a page of input data, then executing the commands in each part module in order. The loop is repeated until all of the input data has been read and processed.

Syntax: PROCESS PARTS

Note: This command must be placed after the SET SEQUENCE command.

See also: PART, PROCESS, SET SEQUENCE

PROFILE

This clause is used with the command below. See the entry for that command.

See: MAIL

PROTECT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Protects the specified variables from being deleted by the CLEAN command. The CLEAN command can delete variables from global, public, or user-created worksheets; it is automatically executed on a public worksheet each time the processing completes on the corresponding part module.

Syntax:
```
PROTECT [worksheet1::]variable1
[, [worksheet2::]variable2 ...]
```

Parameters: *worksheet1*, *worksheet2*—the names of the worksheets containing the variables. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

variable1, *variable2*—the names of the variables to be protected. If no worksheet is specified, the first occurrence of the variable is protected; any private worksheet is searched first, then the worksheets on the stack are searched in order.

See also: CLEAN, CREATE WORKSHEET, GLOBAL, PUBLIC, UNPROTECT

PRUNE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Removes spaces from both ends of a string. If the string parameter is an array, removes spaces from both ends of each element (string).

Syntax:
```
PRUNE(string)
```

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET name = PRUNE(name)

LET itemcodes[2:7] = PRUNE(itemcodes[2:7])

LET code = PRUNE(vendorcodes[n] & "I" & itemcodes[n])
```

See also: CLIP, TRIM, LENGTH

PUBLIC

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

1

Type: Command

Purpose: Creates variables in the public worksheet for the current part module. Variables created with this command are protected by default.

Syntax: `PUBLIC variable1 = initvalue1 [, variable2 = initvalue2 ...]`

Parameters: *variable1*, *variable2*—names for the new variables. A PUBLIC statement may create any number of variables.

initvalue1, *initvalue2*—expressions indicating initial values for the variables, which are assigned to the variables the first time the module is executed.

See also: AUTO, CLEAN WORKSHEET, DESTROY, GLOBAL, PRIVATE, UNPROTECT, SET PUBLIC

2

This clause is used with the command below. See the entry for that command.

See: CLEAN, SET PUBLIC

QUEUE

This clause is the same as the clause PRINTER, used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

RADIUS

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the radius (half the width) of a circle or arc.

Affects: arcs, circles, and ellipses

Syntax: `RADIUS` *`radius`*

Default: 0

Parameters: *radius*—the radius of the circle or arc, in the current units. The radius is the distance between the center or pivot point and each point on the circle or arc. For an ellipse, this forces it to be a circle, and causes the AT point to be the center.

See also: DRAW ARC, DRAW ELLIPSE, SET, UNITS

RATIO

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the ratio of wide bar width to narrow bar width within a bar code.

Affects: bar codes

Syntax: `RATIO` *`ratio`*

Default: 3.0

Parameters: *ratio*—the width of the wide bar, in terms of the narrow bar width. Valid values are 2.0 (for a 2.0 to 1 ratio), 2.5 , and 3.0. This value is ignored for bar code types which do not vary in size.

See also: DENSITY, DRAW BARCODE, SET

READ

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Reads the next page of data (or set of records) from the specified input channel into the input buffer (usually @). READ INPUT reads from the current input channel, or a channel may be specified.

Syntax: `READ {INPUT | "channelname"}`

Parameters: *channelname*—the name of a channel.

See also: CHANNEL, PAGE, READ PAGE, REDIRECT, REWIND, WRITE

READ PAGE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Reads a page of data, specified by page number, from the current input channel into the input buffer (usually @). The specified page becomes the current page.

If the input data is being piped directly to OptioDCS for UNIX from your application as a data stream, pipeline, or socket, this command can be used to skip to a later page; it cannot be used to move back to an earlier page unless the input channel includes the clause SEARCHABLE TRUE.

Syntax: `READ PAGE pagenum`

Parameters: *pagenum*— a constant, variable, or expression, that evaluates to a numerical value

See also: CHANNEL, PAGE, PAGES, READ, REWIND

REDIRECT

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg □ Cha □ Dev □ Fon

Type: Command

Purpose: Sends data from an input channel to an output channel without processing or modifying it.

Syntax: `REDIRECT "inchannel" TO "outchannel"`

Parameters: *inchannel*—the name of the input channel.

outchannel—the name of the output channel.

Example:
```
REWIND INPUT
REDIRECT "stdin" TO "stdout"
```

See also: CHANNEL, READ, REWIND, PAGE, PAGES

REGION

Doc □ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon

Type: Module command

Purpose: Defines a region module. A region module specifies to OptioDCS how to read a section of input data and assign it to variables. This is used to process the section of input data separately from other input data. A region module may be standard, floating, or overlay.

A standard region covers a specific section of the page, starting with the specified line number and extending the specified length. The absence of the FLOATING and OVERLAY keywords indicate this type of module.

A floating region covers a specific number of lines, starting with the current scan line. The FLOATING keyword indicates this type of module.

An overlay region covers a specific section of the page, starting with the specified line number and extending the specified length, but if the current scan line is lower than the specified starting line, region processing begins at the current scan line. The OVERLAY keyword indicates this type of module.

Syntax:
```
REGION "regionname" {FLOATING | [OVERLAY] LINE line#}
[LINES maxlines | BUFFER inputbuf] [ACCEPTS param1 [,param2 ...]]
```

Parameters: *regionname*—a name for the region.

line#—for a standard region, the number of the line where the region starts in the input data; for an overlay region, the minimum starting line number.

maxlines—for a standard or floating region, a numeric expression indicating the number of lines in the region; for an overlay region, a numeric expression indicating the maximum number of lines in the region. When this number of lines has been processed within the region, the region processing is ended. The default is a value that covers the remainder of the input data page.

inputbuf—the input buffer; this sets the maximum number of lines allowed per page of input data to the length of the input buffer. The default input buffer is **@**.

param1, *param2*—variable names for the parameter values that are passed into the region. The values passed in are assigned to these variables, which are used within the region module. A region may have any number of parameters.

Note: The region module must end with the END REGION command.

See also: END, MAP

RELEASE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the release number of the OptioDCS software. For example, if your software is version 2.11.05, the version number is 2, the release number is 11, and the patch number is 05.

Syntax: `RELEASE()`

Examples:
```
LET relnum = RELEASE()

IF RELEASE() > 15 THEN
```

See also: PATCH, VERSION, SERIAL

REPEAT LINE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Causes region or datamap processing to start again at the first command, without affecting the current scan line. The current scan line is a marker which indicates a certain line in the input data; it is used to refer to locations of data on the input page.

Syntax: `REPEAT LINE`

See also: ADVANCE LINE, NEXT LINE

REPEAT ROW

This command is the same as the command REPEAT LINE. See page 1-140.

RESET MACRO

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Resets a macro, so that the next time it is executed, it is downloaded to printer memory again. When OptioDCS processes a document in automatic macro mode (the default), it downloads each macro to printer memory the first time it is executed. Whenever OptioDCS encounters the same macro again, it sends only the execute command to the printer. When a macro is reset, OptioDCS forgets that it has been downloaded, so it is downloaded the next time it is executed. This command is only for processing in automatic mode.

Syntax: `RESET MACRO "`*`macroname`*`"`

Parameter: *macroname*—the name of the macro.

Example: `RESET MACRO "statictext"`

See also: SET MACRO, SET MACROMODE

RESOLUTION

This clause is used with the command below. See the entry for that command.

See: DEVICE

RETURN

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Returns values from a function, document, part, data map, region, format, or segment module, and ends processing of the module.

Syntax: `RETURN retval1 [,retval2 ...]`

Parameters: *retval1*, *retval2*—values that are returned from the module. These may be literal values, variable names used within the module, or expressions. A module may have any number of return values.

See also: CALL, DATAMAP, DOCUMENT, DRAW, EXIT, FORMAT, FUNCTION, MAP, PART, PROCESS, REGION, RUN, SEGMENT

RETURNING

This clause is used with the commands below. See the entries for those commands.

See: CALL, RUN

RETURNS

This clause is used with the command below. See the entry for that command.

See: CHANNEL

REVERSE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Determines whether an object is drawn in reverse. For a reverse image, white pixels become black and black pixels become white; reverse text is printed in white, to contrast with a dark background.

Affects: images and text

Syntax: `REVERSE revtrue`

Default: "False"

Parameters: *revtrue*—a Boolean expression that indicates whether to print the object in reverse.

See also: DRAW IMAGE, DRAW TEXT, SET

REWIND

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: "Rewinds" the file associated with the input channel, so that the next input processing starts at the first byte of the file. REWIND INPUT rewinds the current input channel, or a channel may be specified.

If the input data is being piped directly to OptioDCS for UNIX from your application as a data stream, pipeline, or socket, this command does not work unless the input channel includes the clause SEARCHABLE TRUE.

Syntax: `REWIND {INPUT | "channelname"}`

Parameters: *channelname*—the name of a channel.

See also: CHANNEL, PAGE, READ, READ PAGE

ROLLBACK WORK

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon

Type: Command

Purpose: Undoes any changes made by SQL modules that have not yet been committed to the SQL database. When SQL transaction control is off, each change made by an SQL module is automatically committed to the database, and cannot be rolled back. If autorollback is on and an SQL error occurs, all uncommitted changes are automatically rolled back. All uncommitted SQL statements are automatically rolled back when either the SQL module is closed or when the document ends.

Syntax: `ROLLBACK WORK`

See also: COMMIT WORK, SET SQL AUTOROLLBACK, SET SQL TRANSACTIONS, SQL

ROTATION

Doc □ Par □ Fun □ Dat □ Reg □ For ■ Seg ■ Cha □ Dev □ Fon

1

Type: Clause

Purpose: Controls the rotation of an object, relative to the current page orientation.

Affects: bar codes, images, objects, and text

Syntax: `ROTATION` *`angle`*

Default: 0

Parameters: *angle*—the counter-clockwise angle of rotation. Valid values are 0, 90, 180, and 270.

See also: DRAW BARCODE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, SET

2

This clause is used in a different mode with the command listed below. See the entry for that command.

See: DRAW ARC

ROUND

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Rounds a value up or down to the nearest number having the specified number of places after the decimal point. A fraction of .5 or higher rounds up. For example, ROUND(3.475, 2) evaluates to 3.48.

Syntax: `ROUND(number, decplaces)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

decplaces— a constant, variable, or expression, that evaluates to a numerical value indicating the number of places after the decimal point

Examples:
```
LET difference = ROUND(difference, 3)

IF ROUND(values[x] - values[y] * .33, 0) > 6.0 THEN
```

See also: NUM, SUM, TODOLLARS, TOWORDS

ROW

This clause is the same as the clause LINE, used with the command below. See the entry for that command.

See: REGION

ROWS

1

This clause is used with the commands listed below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL, SET, SQL

2

This clause is the same as the clause LINES, used with the commands listed below. See the entries for those commands.

See: DATAMAP, DRAW COMB, DRAW TEXT, REGION

3

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Counts the rows (elements) of an array. If the specified array does not exist, creates the array and returns the value 1.

Syntax: `ROWS(`*`array`*`)`

Parameters: *array*— an array variable

Examples:

```
LET lines = ROWS(messages)

IF ROWS(items) > limit THEN

LET boxheight = 0.3 * ROWS(warning)
```

See also: COLUMNS, LENGTH, ROWS2, SQUISH, FLATTEN

ROWS2

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Counts the rows (elements) of an array. If the specified array does not exist, returns the value 0.

Syntax: `ROWS2("`*`array`*`")`

Parameters: *array*— an array variable

Examples:

```
LET numlines = ROWS2("messages")

IF ROWS2("items") > limit THEN

LET boxheight = 0.3 * ROWS2("warning")
```

See also: COLUMNS, LENGTH, ROWS, SQUISH, FLATTEN

RUN

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Calls a document module for processing, passing in any parameters needed and receiving any return values. A document module controls the processing of a document.

Syntax: RUN *docname* [WHEN *condition*]
[WITH *param1* [, *param2*…]] [RETURNING *retval1* [, *retval2*…]]

Parameters: *docname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of the document module.

condition—a Boolean expression specifying a condition for executing the command. If the condition is true, the command is executed; if the condition is false, the command is not executed. The WHEN clause must be before any WITH or RETURNING clauses.

param1, *param2*—expressions to be passed into the document as parameter values. The number of WITH values should match the number of parameters the document accepts, and they should be in the same order as defined by the ACCEPTS clause of the DOCUMENT statement.

retval1, *retval2*—variable names for the return values that are received from the document. The number of RETURNING values should match the number of values the document returns, and they should be in the same order as defined by the RETURN statement within the document module.

See also: CALL, DOCUMENT, READ, RETURN, REWIND

SCOLOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Selects a color for the pattern of an object shadow. For more information, see Chapter 2 of the *Advanced Features Guide*.

Affects: chord or pie arcs, boxes, circles, combs, ellipses, image bounding boxes, object bounding boxes, and text bounding boxes

Syntax: SCOLOR *shadowpatcolor*

Default: black

Parameters: *shadowpatcolor*— a literal string or an expression that evaluates to a string, containing the name of a color defined in the current palette module. The palette module is selected in the device module. The default shadow pattern color can be set using the SET SCOLOR command.

See also: COLOR, DEVICE, DRAW ARC, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, FCOLOR, LCOLOR, PALETTE, PATTERN, SHADOW, SET

SEARCH WORKSHEET

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Places the specified global, public, or user-created worksheet on top of the worksheet stack.

Syntax: SEARCH WORKSHEET "*sheetname*"

Parameters: *sheetname*—the name of a global, public, or user-created worksheet. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

See also: CREATE WORKSHEET, IGNORE WORKSHEET, SET GLOBAL, SET PUBLIC

SEARCHABLE

This clause is used with the commands below. See the entries for those commands.

See: ALTER CHANNEL, CHANNEL

SEGMENT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Module command

Purpose: Defines a segment module. A segment module specifies how to format a set of data, text, and graphics that may be used repeatedly within the document or in other documents.

Syntax:
```
SEGMENT "segmentname" [ACCEPTS param1 [,param2 ...]]
[MACRO "macname"] [UNITS "unittype"] [THICKNESS thickvalue]
[LINESTYLE "stylename"] [LINECAP "capname"]
[HEIGHT height] [WIDTH width] [PATTERN "patname"]
[OPAQUE opaqueon] [SHADOW offset,["shadpat"] [,"corner"]]
[REVERSE revtrue] [ROTATION angle] [BORDER borderon]
[RADIUS radius] [FONT fontname [POINTS "pointsize"] |
TYPEFACE "typename" POINTS "pointsize" [SYMBOLS "symset"]] [ALIGN
"alignment"] [UNDERLINE undtrue] [SPACING spacevalue] [LEADING
leadvalue] [LINES numlines]
[DENSITY density] [RATIO ratio] [STYLE "style"]
[EXTENSION "PARAMETER=value"]
```

Parameters: *segmentname*—a name for the segment module.

param1, *param2*—variable names for the parameter values that are passed into the segment. The values passed in are assigned to these variables, which are used within the segment module. A segment may have any number of parameters.

macname—a name for a macro; this creates a macro, including all DRAW statements in the format module, until the first SET MACRO OFF statement. Macros should include only static text and graphic elements. When macros are enabled, OptioDCS sends the macro elements to the printer as a macro. If macros are disabled, the macro elements are sent to the printer as normal elements of the segment module.

unittype—a keyword indicating the units of measurement. This sets the default for the segment module.

thickvalue—a value specifying a line thickness in the current units. This sets the default for the segment module.

stylename—a keyword indicating a line style. This sets the default for the segment module.

capname—a keyword indicating a line cap. This sets the default for the segment module.

height—a numeric expression indicating an object height in the current units. This sets the default for the segment module.

width—a numeric expression indicating an object width in the current units. This sets the default for the segment module.

patname—a keyword indicating a fill pattern. This sets the default for the segment module.

opaqueon—a Boolean expression that indicates whether objects should be opaque, so that any white areas are opaque, or transparent, so that any white areas are transparent. The default is "Off". This sets the default for the segment module.

offset—a numeric expression indicating an offset between objects and their shadows, in the current units. An object with a shadow is always opaque. This sets shadows on, and the default offset for the segment module.

shadpat—a keyword indicating a fill pattern for shadows. This sets the default for the segment module.

corner—a keyword indicating the direction of shadows. This sets the default for the segment module.

revtrue—a Boolean expression that indicates whether to print images and text in reverse, meaning that white pixels become black and black pixels become white. The default is "False". This sets the default for the segment module.

angle—the counter-clockwise angle of rotation of objects. Valid values are 0, 90, 180, and 270. This sets the default for the segment module.

borderon—a Boolean expression that indicates whether to draw a border around the bounding box of an object. This sets the default for the segment module.

radius—the radius of circles, in the current units. This forces the ellipses to be circles, and causes the AT point to be the center. This sets the default for the segment module.

fontname—the name or alias of a font, as specified in a device or fontpack module. This sets the default for the segment module.

pointsize—the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. If used with the FONT clause, this overrides the point size of the font. For PostScript printer drivers, an optional horizontal point size is also supported. This sets the default for the segment module.

typename—the name of a typeface that is available on your printer. This sets the default for the segment module.

symset—the name of a symbol set that has a tagged font metrics (.tfm) file and is supported by your printer. This sets the default for the segment module.

alignment—a keyword indicating the alignment of text within its bounding box; the default is text left. This sets the default for the segment module.

undtrue—a Boolean expression that indicates whether to underline text. The default is "False". This sets the default for the segment module.

spacevalue—a value specifying the horizontal spacing of text, in the current units. This forces fixed spacing of any font. This sets the default for the segment module.

leadvalue—a value specifying vertical line spacing for printing arrays, in the current units. This sets the default for the segment module.

numlines—the number of elements to print from arrays, beginning with the first element. The default is all elements of the array. This sets the default for the segment module.

density—the width of bar code narrow bars in the current units. This value is ignored for bar code types which do not vary in size. This sets the default for the segment module.

ratio—the width of bar code wide bar, in terms of the narrow bar width. Valid values are 2.0 (for a 2.0 to 1 ratio), 2.5 , and 3.0. This value is ignored for bar code types which do not vary in size. This sets the default for the segment module.

style—a keyword indicating a bar code style (a variation within the type). This sets the default for the segment module.

PARAMETER=value—the name of a printing extension parameter and the value to assign to the parameter. Extensions to the DCL language vary according to printer driver; see Chapter 2 or 3 in this book for a list of extensions for your printer driver.

Note: The segment module must end with the END SEGMENT command.

See also: ALIGN, BORDER, DENSITY, DRAW, DRAW TEXT, END, FONT, HEIGHT, LEADING, LINECAP, LINESTYLE, OPAQUE, PATTERN, POINTS, RADIUS, RATIO, REVERSE, ROTATION, SHADOW, SPACING, STYLE, SYMBOLS, THICKNESS, TYPEFACE, UNDERLINE, UNITS, WIDTH

SEPARATOR

This clause is used with the command below. See the entry for that command.

See: CHANNEL

SEQUENCE

This clause is used with the command below. See the entry for that command.

See: SET SEQUENCE

SERIAL

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the serial number of the OptioDCS software.

Syntax: `SERIAL()`

Examples:
```
LET serialnum = SERIAL()

LET string = "OC Serial #: " & SERIAL()
```

See also: PATCH, RELEASE, VERSION

SERVER

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, FAX, MAIL

SET

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ ■ □ □ ■ ■ □ □ □

1

Type: Command

Purpose: Sets a default value.

Syntax: SET *clause*

Parameters: *clause*—a DCL clause, as listed below, along with its parameter(s).

ADVANCE	LEADING	REVERSE	TCOLOR
ALIGN	LINECAP	ROTATION	THICKNESS
BORDER	LINES	ROWS	TYPE
DENSITY	LINESTYLE	SCOLOR	TYPEFACE
FCOLOR	OPAQUE	SHADOW	UNDERLINE
FONT	PATTERN	SPACING	UNITS
HEIGHT	POINTS	STYLE	WIDTH
INVERT	RADIUS	SYMBOLS	WORKSHEET
LCOLOR	RATIO		

Note: SET LINES specifies the number of elements to print from arrays.

Note: SET FONT, SET POINTS, SET SYMBOLS, and SET TYPEFACE specify defaults for printing text.

Note: SET TYPE specifies a bar code type.

See also: ALIGN, BORDER, DENSITY, DRAW BARCODE, DRAW TEXT, FCOLOR, FONT, HEIGHT, LCOLOR, LEADING, LINECAP, LINESTYLE, OPAQUE, PATTERN, POINTS, RADIUS, RATIO, REVERSE, ROTATION, SCOLOR, SHADOW, SPACING, STYLE, SYMBOLS, TCOLOR, THICKNESS, TYPEFACE, UNDERLINE, UNITS, WIDTH, WORKSHEET

2

Type: Command

Purpose: Selects a module or setting. These commands are documented separately.

See: SET COLLATE, SET COPIES, SET DEVICE, SET GLOBAL, SET INPUT, SET LOG FILE, SET MACRO, SET MACROMODE, SET MESSAGE FILE, SET OUTPUT, SET SEQUENCE, SET SQL ACCESSKEY, SET SQL AUTOROLLBACK, SET SQL DATABASE, SET SQL TRANSACTIONS, SET SQL VENDOR, SET TRACE FILE, SET TRACE MASK, SET WARNING

SET COLLATE

Doc ■ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Determines whether the parts of a multi-part document are collated. With collating, all parts are processed with the first page of data, then with the second page, and so on. Without collating, the first part is procesed with all pages of data, then the second part, and so on.

Syntax: `SET COLLATE "collateon"`

Default: "On"

Parameters: *collateon*—a Boolean expression that indicates whether to collate the parts.

See also: SET SEQUENCE

SET COPIES

Doc ■ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: If the printer supports multiple copies, this controls how many copies of each page of a document are printed (for information, see your printer documentation). All copies of the first page are printed, then all copies of the second page, and so on.

Syntax: `SET COPIES numcopies`

Default: 1

Parameters: *numcopies*—a numeric expression that indicates the number of copies. The value can be any positive integer.

SET DEVICE

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Selects a device module for use. A device module specifies the attributes of a type of output device, such as a printer type or a fax application.

Syntax: `SET DEVICE "devicename"`

Parameters: *devicename*—the name of a device module defined in OptioDCS.

See also: DEVICE

SET GLOBAL

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Causes the specified user-created worksheet to act as the global worksheet.

Syntax: `SET GLOBAL "sheetname"`

Parameters: *sheetname*—the name of a user-created worksheet.

See also: CREATE WORKSHEET, IGNORE WORKSHEET, SEARCH WORKSHEET, SET WORKSHEET

SET INPUT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Selects an input channel module or an SQL module for use as an input channel. An input channel module specifies a pipe or path through which OptioDCS receives input data. An SQL module retrieves SQL database data. For more information on SQL modules, see Chapter 5 of the *Advanced Features Guide*.

Syntax: `SET INPUT "inchannelname"`

Default: "stdin"

Parameters: *inchannelname*—the name of an input channel or SQL module, either defined in OptioDCS or the predefined input channel **stdin**. If using an SQL module, it must include a SELECT command and the clause ROWS 1 so that it retrieves one record each time the part modules are processed.

See also: ALTER CHANNEL, CHANNEL, REDIRECT, SQL

SET LOG FILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Specifies the name of the log file for the document. If the file does not already exist, it is created when a LOG command is executed.

Syntax: `SET LOG FILE "filename"`

Default: stdout

Parameters: *filename*—a path and name for the file.

See also: LOG, LOGFILE

SET MACRO

Doc □ Par □ Fun □ Dat □ Reg □ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Starts or ends a macro definition. If the name of a macro is specified, this creates a macro, including all DRAW statements following the command, until the next SET MACRO statement. Macros should include only static text and graphic elements. Macros are numbered according to the ranges specified in the format and device commands. For more information on macros, see the PCL section of Chapter 2 in this book.

Syntax: `SET MACRO {"macroname" | OFF | ""}`

Parameter: *macroname*—a name for the macro.

Examples:

```
SET MACRO "statictext"

SET MACRO OFF

SET MACRO ""
```

See also: DEVICE, FORMAT, RESET MACRO, SET MACROMODE

SET MACROMODE

Doc ■ Par ■ Fun □ Dat □ Reg □ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Sets the mode for macro processing, as listed below. It is best to set the mode in the document module before the PROCESS PARTS or PROCESS command; the mode cannot be changed after the output channel is opened (when the first DRAW statement is executed in a format or segment). For more information on macros, see the PCL section of Chapter 2 in this book.

Automatic – As OptioDCS processes a document, it downloads each macro to printer memory the first time it is executed. Whenever OptioDCS encounters the same macro again, it sends only the execute command to the printer. After document processing, OptioDCS removes the macros from printer memory.

Download - OptioDCS searches the format modules for macros, and downloads any macros (without executing them), numbering them in order of occurrence. All formats (in all documents) that send macros to a particular device must have unique macro ranges. No other format commands are executed. If the output channel sends data to a printer, the macros are sent to printer memory, and remain until removed by command or the printer is turned off. If the output channel sends data to a file, the macros are written to the file, which can be used for creating a macro catridge, or for manually downloading to a printer.

Preloaded - OptioDCS assumes that all macros have already been downloaded, so when a macro is encountered in a format module, a command is sent to the printer to execute it. It is assumed that the macros are numbered in order of execution.

Syntax: `SET MACROMODE` *`mode`*

Parameter: *mode*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a keyword indicating the mode as listed below. The default is automatic.

Mode	Keywords		
automatic	AUTOMATIC	AUTO	A
download	DOWNLOAD	D	
preloaded	PRELOADED	P	

Example:

```
SET MACROMODE "PRELOADED"
```

See also: RESET MACRO, SET MACRO

SET MESSAGE FILE

This command is the same as the command SET LOG FILE. See page 1-156.

SET OUTPUT

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Selects an output channel module for use. An output channel module specifies a pipe or path through which OptioDCS sends a processed document to the output device.

Syntax: `SET OUTPUT "outchannelname"`

Default: "stdout"

Parameters: *outchannelname*—the name of an output channel, either defined in OptioDCS or one of the predefined output channels **stdout**, **stderr**, **smtp** (SMTP e-mail channel), **mapi** (MAPI e-mail channel, for Windows NT systems only), or **faxfx** (standard OptioFAX channel).

See also: ALTER CHANNEL, CHANNEL, MAIL

SET PUBLIC

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Causes the specified user-created worksheet to act as the public worksheet for the current part module.

Syntax: `SET PUBLIC "sheetname"`

Parameters: *sheetname*—the name of a user-created worksheet.

See also: CREATE WORKSHEET, IGNORE WORKSHEET, PROTECT, SEARCH WORKSHEET, SET WORKSHEET

SET SEQUENCE

Doc ■ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Specifies the order of processing for the part modules of a document. Each part module controls the processing of one part of a document.

Syntax: `SET SEQUENCE "part1" [,"part2"...]`

Parameters: *part1, part2*—the names of the part modules, in the order they should be processed. A document may have any number or parts.

Note: This command must be placed before the PROCESS PARTS command.

See also: PART, PROCESS PARTS

SET SQL ACCESSKEY

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Selects an access key for OptioDCS to use to log on to an SQL database. This is necessary when using SQL statements within an OptioDCS document. For more information on using SQL statements, see Chapter 5 of the *Advanced Features Guide*. For information on access keys, see Chapter 2 of the *User's Guide*.

Syntax: `SET SQL ACCESSKEY keyname`

Parameters: *keyname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key that OptioDCS uses to log on to the SQL database selected by the SET SQL DATABASE command.

Example: `SET SQL ACCESSKEY "sqluser"`

See also: SET SQL DATABASE, SET SQL VENDOR

SET SQL AUTOROLLBACK

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon

Type: Command

Purpose: Determines whether changes made by SQL modules that have not been committed to the SQL database should be automatically undone when an SQL error occurs. When SQL transaction control is off, each change made by an SQL module is automatically committed to the database, and cannot be rolled back.

Syntax: `SET SQL AUTOROLLBACK "rollbackon"`

Default: "On"

Parameters: *rollbackon*—a Boolean expression that indicates whether uncommitted SQL changes should be automatically rolled back when an SQL error occurs.

See also: COMMIT WORK, ROLLBACK WORK, SET SQL TRANSACTIONS, SQL

SET SQL DATABASE

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon

Type: Command

Purpose: Selects an SQL database. This is necessary when using SQL statements within an OptioDCS document. For more information on using SQL statements, see Chapter 5 of the *Advanced Features Guide*.

Syntax: `SET SQL DATABASE dbname`

Parameters: *dbname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the SQL database.

For an ODBC-compliant database, use the name of the system DSN defined on the OptioDCS server. For more information on system DSN names, see your ODBC documentation.

For an Oracle database, use the database alias as defined in the Oracle client software on the OptioDCS server.

Example:
```
SET SQL DATABASE "Personnel"
```

See also: SET SQL ACCESSKEY, SET SQL VENDOR

SET SQL TRANSACTIONS

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Determines whether changes made by SQL modules are automatically committed to the SQL database. When SQL transaction control is off, each change made by an SQL module is automatically committed to the database, and cannot be rolled back. When SQL transaction control is on, changes made by SQL statements are not commited to the database until the COMMIT WORK command is executed.

Syntax: `SET SQL TRANSACTIONS "transacton"`

Default: "Off"

Parameters: *transacton*—a Boolean expression that indicates whether SQL changes should not be automatically committed to the SQL database.

See also: COMMIT WORK, ROLLBACK WORK, SET SQL AUTOROLLBACK, SQL

SET SQL VENDOR

Doc ■ Par ■ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Command

Purpose: Indicates the type of SQL database. This is necessary when using SQL statements within an OptioDCS document. For more information on using SQL statements, see Chapter 5 of the *Advanced Features Guide*. *This command is not valid for UNIX systems.*

Syntax: `SET SQL VENDOR ["ODBC" | "OCI"]`

Parameters: VENDOR—a keyword indicating the type of the database selected by the SET SQL DATABASE command; "ODBC" (the default) indicates an ODBC-compliant database; "OCI" indicates an Oracle database.

Example:
```
SET SQL VENDOR "OCI"
```

See also: SET SQL ACCESSKEY, SET SQL DATABASE

SET TRACE FILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Specifies the name of the trace file for the document. If the file does not already exist, it is created as needed to store trace messages. The TRACE command generates trace messages, which are filtered by the trace mask.

Syntax: `SET TRACE FILE "filename"`

Default: stdout

Parameters: *filename*—a path and name for the file.

See also: SET TRACE MASK, TRACE, TRACEFILE

SET TRACE MASK

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Controls which mask pattern is used for the trace log file.

Syntax: `SET TRACE MASK masknum`

Default: allows all messages to be logged

Parameters: *masknum*—a numeric expression that evaluates to the decimal equivalent of a 32-bit integer. Each bit of the integer represents one type of trace message. If a bit is 1, the corresponding message is logged; if it is 0, the message is ignored. For a list of message types, see the TRACEMASK entry.

See also: TRACE, TRACEMASK

SET WARNING FILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Specifies the name of the warning file for the document. If the file does not already exist, it is created as needed to store warning messages. Warning messages can be generated by OptioMedForms or by the WARNING command. All warning messages are filtered by the warning mask.

Syntax: `SET WARNING FILE "filename"`

Default: stdout

Parameters: *filename*—a path and name for the file.

See also: SET WARNING MASK, WARNING, WARNINGFILE

SET WARNING MASK

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Purpose: Controls which mask pattern is used for the warning log file.

Syntax: `SET WARNING MASK masknum`

Default: allows all messages to be logged

Parameters: *masknum*—a numeric expression that evaluates to the decimal equivalent of a 32-bit integer. Each bit of the integer represents one type of warning message. If a bit is 1, the corresponding message is logged; if it is 0, the message is ignored. For a list of message types, see the WARNINGMASK entry.

See also: WARNING, WARNINGMASK

SET WORKSHEET

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Sets the default worksheet to the specified global, public, or user-created worksheet. Any new variables created in the current module using the LET command (after the SET WORKSHEET command) are placed in the default worksheet.

Syntax: `SET WORKSHEET "sheetname"`

Parameters: *sheetname*—the name of a global, public, or user-created worksheet. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

See also: CREATE WORKSHEET, DESTROY WORKSHEET, SEARCH WORKSHEET, SET GLOBAL, SET PUBLIC

SHADOW

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Determines whether a shadow is drawn behind an object. An object with a shadow is always opaque.

Note: If the SHADOW clause is used with the SET command (i.e., SET SHADOW) all subsequent objects in the module are drawn with shadows.

Affects: chord or pie arcs, boxes, circles, combs, ellipses, image bounding boxes, object bounding boxes, and text bounding boxes

Syntax: `SHADOW offset,["shadpat"] [,"corner"]`

Default: no shadow
with a shadow, *shadpat* default is the current default pattern and *corner* default is "BR"

Parameters: *offset*—the offset between the object and the shadow, in the current units.

shadpat—a keyword indicating a fill pattern for the shadow. All pattern keywords are valid; see the PATTERN entry, page 1-127.

corner—a keyword indicating the direction of the shadow, as listed in the table below.

Corner	Keyword
top left	TL
top right	TR
bottom left	BL
bottom right	BR

See also: DRAW ARC, DRAW BOX, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, OPAQUE, SET, UNITS

SHOWDATE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a date and time from a number of seconds to a string in the specified format. The date and time must be in the form of the number of seconds since 00:00:00 Greenwich mean time (GMT), January 1, 1970.

Syntax: `SHOWDATE(`*`formstring`*`, `*`numseconds`*`)`

Parameters: *formstring*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the format for the date and time. The most commonly used formatting symbols are listed below. (To print a % sign, use "%%".)

	Symbol	Produces
Date	%a	abbreviated weekday name
	%A	full weekday name
	%b	abbreviated month name
	%B	full month name
	%d	day of month (01 - 31)
	%D	date as %m/%d/%y
	%e	day of month (1-31; single digits preceded by a blank)
	%h	abbreviated month name
	%m	month number (01 - 12)
	%y	year within century (00 - 99)
	%Y	year as ccyy (for example, 1986)
Time	%H	hour (00 - 23)
	%I	hour (01 - 12)
	%M	minute (00 - 59)
	%p	equivalent of either AM or PM
	%R	same as %H:%M
Both	%c	basic date and time representation

numseconds—a constant, variable, or expression, that evaluates to a numerical value

Examples:

```
LET currtime = SHOWDATE("Current time: %H:%M", NOW())

LET duedate = SHOWDATE("%m/%d/%y", NOW() + 31*60*60)
```

See also: DATE, NOW, TIME

SIN

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Calculates the sine of an angle in radians.

Syntax: `SIN(`*`angle`*`)`

Parameters: *angle*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
IF SIN(firstangle) < 0.4 THEN

LET answer = SIN((ang1 - ang2) * 3)
```

See also: ASIN, COS, TAN

SMART

This clause is used with the command below. See the entry for that command.

See: CHANNEL

SORT

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Sorts the elements of a group of arrays according to the values of elements in the key arrays. All arrays are sorted by the elements in the first key array; if any of those elements are the same, those elements are sorted by the elements in the second key array; and so on. This is similar to sorting a table by certain columns: complete rows of data are sorted, so that the items in one column are in order, and the data on each row is kept together. Returns TRUE if the sort is completed; returns FALSE if not.

You can use this function to sort pages of input from **stdin**, by first having OptioDCS read in all input data, then sorting by key values that are on every page.

Note: If a key array contains numeric values in strings, it is sorted alphabetically

instead of numerically, e.g., 1, 200, 30, 4. To force a numeric sort, convert the array to numeric values using the TONUMBER function.

Syntax: SORT(*numkeys*, *sortorders*,
keyarray1 [, *keyarray2…*] [, *alsosort1* [, *alsosort2…*]])

Parameters: *numkeys*—a literal value, a variable, or an expression that evaluates to a numeric value, containing the number of key arrays.

sortorders—an array of boolean values, indicating the sort order for each key array. A value of TRUE indicates ascending sort order, FALSE indicates descending order. This array must have one element for each key array. The first element is applied to the first key array, the second element to the second key array, and so on.

keyarray1, *keyarray2*—arrays containing the key values. All key arrays must have the same number of elements. The values may be strings, numeric values, or boolean values. All arrays are sorted by the values in the first key array; if any elements are the same in the first array, those elements are sorted by the elements in the second key array; and so on. You can specify any number of key arrays. Larger numbers of key arrays may slow document processing.

alsosort1, *alsosort2*—other arrays that are sorted in the same order as the key arrays, regardless of the element values. You can sort any number of arrays.

Examples:
```
LET sortworked = SORT(2, orders, lastnames, firstnames, exts, titles)

LET sorted = SORT(2, <<FALSE, TRUE>>, prices, itemnums, itemdescs,
quantities)

LET sorted = SORT(1, <<TRUE>>, zipcodes, townnames, states)
```

See also: FIND, LOOKUP, READ, TONUMBER, WHERE

SPACING

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the horizontal spacing of characters. This forces fixed spacing of any font.

Affects: text

Syntax: SPACING *spacevalue*

Default: the standard spacing values of the current font

Parameters: *spacevalue*—a value specifying the horizontal spacing of the characters, in the current units. This forces fixed spacing of any font. To set the spacing values to the default for the font, specify a value of -1.

See also: DRAW TEXT, LEADING, SET, UNITS

SPAN

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Searches a string (or array of strings) for a match of an expression; if a match is found, returns the length of the first matching substring; otherwise, returns 0. If searching an array of strings, returns an array of the lengths of the first matching substring in each element. The search expression may be a regular expression, which is a notation for describing patterns of characters (see your Windows NT or UNIX system documentation for more information).

Syntax: `SPAN(searchexpr,string2)`

Parameters: *searchexpr*— a literal string, a variable containing a string, or an expression or regular expression that evaluates to a string

string2— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET morelength = SPAN(moreflag & "*",@[55,1:8])

WHILE SPAN(names[i],info[i,1:length]) > 12
```

See also: FIND, LENGTH, MATCHES, WHERE

SQL

Doc □ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

1

Type: Module command

Purpose: Defines an SQL module. An SQL module retrieves, inserts, or deletes SQL database data, and can be used as an input channel. For more information on SQL modules, see Chapter 5 of the *Advanced Features Guide*.

Syntax:
```
SQL modulename [VENDOR {"ODBC" | "OCI"}]
DATABASE "dbname" ACCESSKEY "keyname"
[ROWS numrecords] SQLstatement;
```

Parameters: *modulename*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing a name for the SQL module.

VENDOR—a keyword indicating the type of the SQL database ; "ODBC" (the default) indicates an ODBC-compliant database; "OCI" indicates an Oracle database.

dbname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the SQL database.

For an ODBC-compliant database, use the name of the system DSN defined on the OptioDCS server. For more information on system DSN names, see your ODBC documentation.

For an Oracle database, use the database alias as defined in the Oracle client software on the OptioDCS server.

keyname—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the name of the access key that OptioDCS uses to log on to the SQL database. For more information on access keys, see Chapter 2 of the *User's Guide*.

numrecords—a numeric expression indicating the number of records to retrieve, delete, or insert at one time. The default is all records.

SQLstatement—an SQL statement, in the SQL syntax supported by your database, specifying the action to perform on the database. The statement must end with a semicolon, and must be the last clause in the SQL module.

If a SELECT command retrieves data of an unsupported type, OptioDCS creates a variable for the data, but assigns an empty string to it. If a SELECT command retrieves long numeric data, OptioDCS truncates the value to 15 significant digits.

If a DELETE or INSERT command includes a DCL variable, the data type of the variable must match the data type of the database column. By default, DCL variables are treated as strings; use the TONUMBER function to convert the variable to a number if needed.

Examples:
```
SQL "GetEmpNames"
   VENDOR "OCI"
   DATABASE "Personnel"
   ACCESSKEY "sqluser"
   ROWS 1
   SELECT lastname FROM Employees WHERE Age > 30;
END SQL
```

Note: The SQL module must end with the END SQL command.

See also: CHANNEL, END, FETCH, SET INPUT, SQLERROR, SQLRUN, TONUMBER

2

This keyword is part of the commands below. See the entries for those commands.

See: SET SQL ACESSKEY, SET SQL DATABASE, SET SQL VENDOR

SQLERRMSG

Type: Function

Purpose: Returns the ten most recent error messages that the SQL database driver provides to OptioDCS.

Syntax: `SQLERRMSG("modname")`

Parameters: *modname*—the SQL module name

Example:
```
SQLRUN "GetNumbers"
LET rc = SQLERROR()
IF rc != 0 THEN
   LET messages = SQLERRMSG("GetCustomers")
   LET nRows = ROWS(messages)
   WARNING "last error message: " & messages[nRows]
END IF
```

See also: FETCH, SQL, SQLERROR, SQLRUN

SQLERROR

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns any error code generated by processing an SQL module; if the SQL module executed successfully, returns 0.

Syntax: `SQLERROR()`

Example:
```
SQLRUN "GetNumbers"
LET rc = SQLERROR()
IF rc != 0 THEN
```

```
      WARNING "SQL error" & rc & "detected; aborting job"
      EXIT DOCUMENT
END IF
```

See also: FETCH, SQL, SQLERRMSG, SQLRUN

SQLRUN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Calls an SQL module for executing, logging on to the database with the access key and executing the SQL statement as specified in the module. An SQL module retrieves, inserts, or deletes SQL database data. If the SQL module contains a ROWS clause, only the first set of matching records are affected when the module is executed. For more information, see Chapter 5 of the *Advanced Features Guide*.

Syntax: SQLRUN *sqlmodname*

Parameters: *sqlmodname*—a literal string, a variable containing a string, or an expression that evaluates to a string, containing the path and name of the SQL module.

Example:

Doc Par Fun Dat Reg For Seg Cha Dev Fon

```
PART "File PO"
   MAP "PurchOrd"
   SQLRUN "GetVendorName"
   DRAW "FileForm"
END PART
```

See also: CALL, FETCH, RUN, SET INPUT, SQL, SQLERROR

SQUISH

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Determines which elements in an array are blank, and returns an array of only the non-blank elements (so that the non-blank elements are numbered consecutively). If the result is assigned to the same array, the extra array elements are replaced with empty strings. This can be used to remove blank lines from an array of text strings.

Syntax: SQUISH(*array*)

Parameters: *array*— an array variable

Examples: `LET list = SQUISH(results)`

`LET messages = SQUISH(messages)`

See also: FLATTEN, ISBLANK, ROWS

STACKER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Defines a keyword for an output tray or stacker available on a device.

Syntax: `STACKER "trayword","traycode"`

Parameters: *trayword*—a keyword for the output tray or stacker.

traycode—the code that indicates the corresponding output tray or stacker to the printer.

See also: FORMAT

2

This clause is used in a different mode with the command below. See the entry for that command.

See: FORMAT

START

This clause is used with the command below. See the entry for that command.

See: DRAW ARC

STOP

This clause is used with the command below. See the entry for that command.

See: DRAW ARC

STYLE

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

1

Type: Clause

Purpose: Controls the style of a bar code, which is a variation within the bar code type.

Affects: bar codes

Syntax: `STYLE "style"`

Default: none

Parameters: *style*—a keyword indicating the bar code style (a variation within the type). See Chapter 2 or 3 in this book for a list of styles for each bar code type for your printer driver.

See also: DRAW BARCODE, SET

2

This clause is used in a different sense with the command below. See the entry for that command.

See: FONT

SUM

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Calculates the sum of the values in an array.

Syntax: `SUM(array)`

Parameters: *array*— an array variable containing numeric values, or strings of only numeric characters

Examples:
```
LET total = SUM(quantities)

IF SUM(prices[2:7]) > 250.00 THEN
```

See also: ABS, AVG, MAX, MIN, ROUND, ROWS

SURFACE

This clause is used with the command below. See the entry for that command.

See: FORMAT

SWITCH

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Executes various sets of commands depending on the value of an expression. The expression is evaluated, and then compared to specified case values in order. When a matching value is found, the commands between the matching CASE clause and the first EXIT SWITCH or END SWITCH command are executed. If none of the case values match the expression value, the commands following the OTHERWISE clause are executed.

Syntax:
```
SWITCH expression CASE case_1 1_commands [EXIT SWITCH]
[CASE case_2 2_commands [EXIT SWITCH]...]
OTHERWISE othercommands END SWITCH
```

Parameters: *expression*—the expression that is evaluated and compared to the case values.

case_1, case_2—expressions specifying possible values of the *expression*. A SWITCH statement may have any number of CASE clauses.

1_commands, 2_commands—commands that are executed if the expression matches the case value.

Note: The EXIT SWITCH command is used to end the processing of the case commands. If the case commands do not include an EXIT SWITCH command, processing continues with the commands of the next case.

othercommands—commands that are executed if *expression* equals none of the case values.

Note: The SWITCH statement must end with the END SWITCH command.

See also: END, EXIT, IF, WHILE

SYMBOLS

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ ■ ■

Type: Clause

Purpose: Controls the symbol set used for printing text.

Affects: text

Syntax: `SYMBOLS "symset"`

Default: the default symbol set for the current typeface

Parameters: *symset*—the name of a symbol set that has a tagged font metrics (.tfm) file and is supported by your printer.

See also: DRAW TEXT, FONT, SET, TYPEFACE

SYSTEM

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: For UNIX systems, starts a system shell and executes a UNIX command.

For Windows NT systems, starts a system shell and executes a shell command. Document processing pauses while the shell is running. This function always returns a blank string.

Syntax: `SYSTEM(command)`

Parameters: *command*— a literal command text string or a variable containing a command string

Examples: for UNIX systems

```
LET sys = SYSTEM("cat systemid")

LET sys = SYSTEM(commands[4])

LET username = SYSTEM("grep $LOGNAME /etc/passwd | awk -F:
                       '{ print $5 }'")
```

for Windows NT systems

```
LET sys = SYSTEM("copy special.fgl special2.fgl")

LET sys = SYSTEM(commands[4])
```

See also: CPUNAME, EXPORT, HOSTNAME, IMPORT, OSCPUID, OSNAME, PARAMETER, PLATFORM, TRANSMIT, USERNAME

TABS

This clause is used with the command below. See the entry for that command.

See: CHANNEL

TAN

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Calculates the tangent of an angle in radians.

Syntax: `TAN(`*`angle`*`)`

Parameters: *angle*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
IF TAN(firstangle) < 0.4 THEN

LET answer = TAN((ang1 - ang2) * 3)
```

See also: ATAN, COS, SIN

TCOLOR

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Selects a color for text. For more information, see Chapter 2 of the *Advanced Features Guide*.

Affects: text

Syntax: TCOLOR *textcolor*

Default: black

Parameters: *textcolor*— a literal string or an expression that evaluates to a string, containing the name of a color defined in the current palette module. The palette module is selected in the device module. The default text color can be set using the SET TCOLOR command.

See also: COLOR, DEVICE, DRAW BARCODE, DRAW TEXT, LCOLOR, PALETTE, SET

TEXT

This keyword is part of the command below. See the entry for that command.

See: DRAW TEXT

THEN

This clause is used with the command below. See the entry for that command.

See: IF

THICKNESS

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ □ □

Type: Clause

Purpose: Controls the thickness of an object or its outline.

Affects: arcs, bar code borders, boxes, circles, comb borders, curves, ellipses, image borders, lines, object borders, and text borders

Syntax: `THICKNESS` *`thickvalue`*

Default: 0

Parameters: *thickvalue*—a value specifying the thickness of the object or its outline, in the current units.

See also: BORDER, DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, SET, UNITS

TIME

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Returns the system time, in the form *hh*:*mm*:*ss*, where *hh* are hours, *mm* are minutes, and *ss* are seconds. This function uses a 24-hour clock, i.e., 1:00 p.m. is represented as 13:00:00, and so on.

Syntax: `TIME()`

Examples: `LET printtime = TIME()`

`LET string = "Time processed: " & TIME()`

See also: DATE, NOW, SHOWDATE

TO

This clause is used with the commands listed below. See the entries for those commands.

See: DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, FOR

TOBOOLEAN

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns a boolean value (true or false) according to whether a string is TRUE or FALSE. The boolean value can then be used to test for conditions in IF and WHILE commands.

For a numerical value, if it is 0, the boolean value is FALSE; otherwise the boolean value is TRUE.

Certain strings are converted to boolean values as listed below.

TRUE	FALSE
"TRUE"	"FALSE"
"T"	"F"
".T."	".F."
"YES"	"NO"
"Y"	"N"
"ON"	"OFF"
"1"	"0"
"+"	"-"

Syntax: `TOBOOLEAN(`*`expression`*`)`

Parameters: *expression*— a constant, literal string, variable, or expression

Examples: `LET done = TOBOOLEAN("false")`

```
IF TOBOOLEAN(lasttime) THEN

WHILE TOBOOLEAN(answers[3])
```

See also: TOLOWER, TOUPPER, TOSTRING

TODOLLARS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a number to a string representing a dollar value, using words instead of numerical characters. Any fractions of cents are rounded to the nearest whole cent; a fraction of .005 or higher rounds up. For example, 84.107 becomes "eighty four dollars and eleven cents". If the number parameter is an array, each element is converted to a string.

Syntax: `TODOLLARS(`*`number`*`)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

Examples:

```
LET fourgrand = TODOLLARS(4000)

LET balance = TODOLLARS(balance)

LET strings[2:7] = TODOLLARS(amount[2:7])

LET string = TODOLLARS(amounts[1] - amounts[2] * .33)
```

See also: ISMONEY, ISNUMERIC, ROUND, TOWORDS, TONUMBER

TOHEX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a decimal number to hexadecimal.

Syntax: `TOHEX(`*`number`*`)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
LET hexsum = TOHEX(sum)

LET hexmask = TOHEX(TRACEMASK())

LET hexamts[2:7] = TOHEX(amounts[2:7] * 2)
```

See also: FROMHEX, ISNUMERIC, TRACEMASK, WARNINGMASK

TOLOWER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a string to all lower case characters.

Syntax: `TOLOWER(`*`string`*`)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET message[3] = TOLOWER(message[3])

LET nameline = TOLOWER(lastname & ", " & firstname)
```

See also: TOUPPER

TONUMBER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a string of numeric characters to a number. If the string contains a leading or trailing minus sign, it is converted to a negative number. All non-numeric characters are removed (including spaces). For example, "-123a 4.5", "-12 +34.5", and "1,234.5-" would each become -1234.5. This function correctly handles numbers with up to seven places after the decimal point.

Syntax: `TONUMBER(`*`string`*`, "`*`decimalchar`*`")`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

decimalchar— the single character that separates the whole number portion of the string from the fraction portion. For example, in North America this is usually a period; in Europe, it is usually a comma. The default is a period.

Examples:
```
IF TONUMBER(amount) > 200 THEN

LET costs[1:6] = TONUMBER(prices[2:7])

LET codenum = TONUMBER(vendornum & "000" & itemnum)
```

See also: ISALNUM, ISDIGITS, ISNUMERIC, NUM, PICTURE, TOSTRING, TOWORDS

TOSTRING

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a number to a string of characters.

Syntax: `TOSTRING(`*`number`*`)`

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
LET fourgrand = TOSTRING(4000)

LET balance = TOSTRING(balance)

LET strings[2:7] = TOSTRING(amount[2:7])

LET string = TOSTRING(amounts[1] - amounts[2] * .33)
```

See also: ISDIGITS, ISNUMERIC, TONUMBER, TOWORDS

TOUPPER

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a string to all upper case characters.

Syntax: TOUPPER(*string*)

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET message[3] = TOUPPER(message[3])

LET nameline = TOUPPER(lastname & ", " & firstname)
```

See also: TOLOWER

TOWORDS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a number to a string, using words instead of numerical characters. Any fractions smaller than 1/1000 are rounded to the nearest thousandth; a fraction of .0005 or higher rounds up. For example, 84.1037 becomes “eighty four point one zero four”. For whole numbers, “point zero zero zero” is added to the end of the string. For example, 23 becomes “twenty-three point zero zero zero”.

Syntax: TOWORDS(*number*)

Parameters: *number*— a constant, variable, or expression, that evaluates to a numerical value

Examples:
```
LET fourgrand = TOWORDS(4000)

LET balance = TOWORDS(balance)

LET strings[2:7] = TOWORDS(amount[2:7])

LET string = TOWORDS(amounts[1] - amounts[2] * .33)
```

See also: ISNUMERIC, ISMONEY, ROUND, TODOLLARS, TOSTRING

TRACE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Writes a variable value or messages to a trace log file.

TRACE CACHES writes the contents of the caches.

TRACE DEFAULTS writes the default values.

If a worksheet is specified, the contents of that worksheet is written to the file; TRACE WORKSHEETS writes the contents of all worksheets; TRACE WORKSPACE writes the contents of the workspace, such as environment settings.

Specific messages may also be specified. A newline is appended to the end of each line of data in the message; to suppress this, place a semicolon at the end of the TRACE statement.

Syntax: `TRACE {CACHES | DEFAULTS | WORKSHEET "`*`sheetname`*`" | WORKSHEETS | WORKSPACE | `*`message1`*` [, `*`message2`*`...][;]}`

Parameters: *sheetname*—the name of a worksheet.

message1, *message2*—variables or literal strings containing messages. A TRACE statement may have any number of messages.

See also: SET TRACE FILE, SET TRACE MASK

2

This clause is used with the commands listed below. See the entries for those commands.

See: SET TRACE FILE, SET TRACE MASK

TRACEFILE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the name of the trace file.

Syntax: `TRACEFILE()`

Examples:
```
LET filename = TRACEFILE()

LET string = "Log: " & TRACEFILE()
```

See also: SET TRACE FILE, TRACE

TRACEMASK

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the current value of the trace mask, or converts a string of keywords to a valid mask value.

To return the current mask value, do not use a parameter. The value is returned as a (usually large) decimal number.

To convert keywords to a mask value, use a string of keywords as the parameter. If the string parameter is an array, the strings are concatenated and treated as one string.

Syntax:
```
TRACEMASK()
TRACEMASK(string)
```

Parameters: *string*— a literal string of trace mask keywords and plus or minus signs, a variable containing a string or strings, or an expression that evaluates to a string; keywords are listed below.

Keyword	Message Types
All	all trace messages
None	no trace messages
Assignments	traces LET statements
BlobCache	traces loading or unloading of blob files from disk
Breaks	shows the values of the variables specified in the BREAK ON command
Channels	shows when channels are opened and closed
Elements	traces the drawing of text and graphics
EnvData	shows system parameters as they are loaded
Files	traces file and directory activity
FontData	shows what fonts and spacing tables are loaded
ImageCache	traces loading or unloading of image files from disk
ImageData	shows information on image files that are loaded, such as the size and file type
Keys	shows the authorization keys as they are read
Misc	reserved for future use

Modules	traces the execution of modules
Sets	traces SET statements
SpoolData	shows each page of input data as it is read in
Tests	traces IF statements

Examples:

```
LET lastmask = TOHEX(TRACEMASK())

SET TRACE MASK TRACEMASK("None +Tests +Breaks")

SET TRACE MASK TRACEMASK("All -Sets -Keys")
```

See also: SET TRACE MASK, TOHEX, TRACEFILE, TRACENAMES

TRACENAMES

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Converts a trace mask value to an array of keywords. See page 1-185 for a list of mask keywords.

Syntax: `TRACENAMES(maskvalue)`

Parameters: *maskvalue*— a constant, variable, or expression, that evaluates to a numerical value that is valid as a trace mask

Examples:

```
LET lastmask = TRACENAMES(TRACEMASK())

SET TRACE MASK TRACEMASK(TRACENAMES(newvalue))
```

See also: SET TRACE MASK, TRACEMASK

TRANSLATE

Doc □ Par □ Fun □ Dat □ Reg □ For □ Seg □ Cha □ Dev □ Fon □

Type: Module command

Purpose: Defines a translate module, which is a translation table for input data. A translation table consists of sets of character substitutions, which are made in the input data before the data is processed. For more information, see Chapter 4 of the *Advanced Features Guide*.

Syntax: TRANSLATE "*tablename*" [*fromchar1* = *tochar1* [*fromchar2* = *tochar2* ...]]

Parameters: *tablename*—a name for the translate module.

fromchar1, *fromchar2*—input data characters that are replaced. A character may be indicated by a string containing the character (or its escape sequence), or by the character's ASCII number.

tochar1, *tochar2*—characters that replace the original input data characters. A character may be indicated by a string containing the character (or its escape sequence), or by the character's ASCII number. To remove a character from the input data, replace it with 0 (the ASCII value of the NUL character).

Note: The translate module must end with the END TRANSLATE command.

See also: CHANNEL, END

TRANSMIT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Passes a string to a smart channel, usually a string containing a parameter value for use by a UNIX shell script. The string is passed to the currently selected output channel. A smart channel is used to pass information, in addition to processed documents, to applications. The channel module must have the command SMART "On". *This command is not valid for Windows NT systems.*

Syntax: TRANSMIT *string1* [, *string2*...]

Parameters: *string1*, *string2*— a series of literal strings, variables containing strings, or expressions that evaluate to strings

See also: EXPORT, FILENAME, IMPORT, PARAMETER, SMART, SYSTEM

2

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Passes a string to a smart channel, usually a string containing a parameter value for use by a UNIX shell script. A smart channel is used to pass information, in addition to

processed documents, to applications. The channel module must have the command SMART "On". *This function is not valid for Windows NT systems.*

Syntax: `TRANSMIT(channelstring,string)`

Parameters: *channelstring*— a literal string or variable containing the name of a channel (predefined channels are stdin, stdout, and stderr)

string— a literal string, a variable containing a string, or an expression that evaluates to a string

Example:
```
LET status = TRANSMIT("faxfx","FAXDIR=/OptioFAX/spool")
```

See also: EXPORT, FILENAME, IMPORT, PARAMETER, SMART, SYSTEM

TRIM

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Removes spaces from the end of a string. If the string parameter is an array, removes spaces from both ends of each element (string).

Syntax: `TRIM(string)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Examples:
```
LET name = TRIM(name)

LET itemcodes[2:7] = TRIM(itemcodes[2:7])

LET code = TRIM(vendorcodes[n] & "I" & itemcodes[n])
```

See also: CLIP, PRUNE, LENGTH

TYPE

This clause is used with the commands listed below. See the entries for those commands.

See: DRAW ARC, DRAW BARCODE, PALETTE, SET

TYPEFACE

Doc Par Fun Dat Reg For Seg Cha Dev Fon
□ □ □ □ □ ■ ■ □ ■ ■

Type: Clause

Purpose: Controls the typeface of a font.

Affects: text

Syntax: `TYPEFACE "typename"`

Default: none

Parameters: *typename*—the name of a typeface that is available on your printer.

See also: DRAW TEXT, FONT, SET

UNCANON

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Function

Purpose: Detects any special (non-ASCII) characters in a string. Returns the same string with each special character converted to an escape code character sequence (beginning with a carat or backslash).

Syntax: `UNCANON(string)`

Parameters: *string*— a variable containing a string, or an expression that forms a string

Examples: `LET readstring = UNCANON(printstring)`

`LET mystring = UNCANON(mystring & escapecode)`

See also: CANON, VISIBLE, VISIBLEX

UNDERLINE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Determines whether text is printed with an underline.

Affects: text

Syntax: `UNDERLINE` *`undtrue`*

Default: "False"

Parameters: *undtrue*—a Boolean expression that indicates whether to underline the text.

See also: DRAW TEXT, SET

UNESCAPE

This function is the same as the function UNCANON. See page 1-189.

UNITS

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Clause

Purpose: Controls the units of measurement used for positioning and measuring graphics, text, and margins.

Affects: arcs, bar codes, borders, boxes, circles, combs, curves, ellipses, images, lines, margins, objects, shadows, and text

Syntax: UNITS "*unittype*"

Default: "INCHES"

Parameters: *unittype*—a keyword indicating the units of measurement, as listed in the table below. When producing ASCII output, use the keyword "Characters".

Units	Keywords		
points	DPI72	POINTS	
300 dpi	DPI300	DOTS	
decipoints	DPI720	DECIPOINTS	
microinches	DPI1000	MI	MICROINCHES
micropoints	DPI7200	MP	MICROPOINTS
inches	INCHES	IN	
centimeters	CENTIMETERS	CM	CENTIMETRES
rows and columns	Characters		

See also: DENSITY, DEVICE, DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, HEIGHT, LEADING, RADIUS, SET, SHADOW, SPACING, THICKNESS, WIDTH

UNPROTECT

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Command

Purpose: Removes protection from the specified variables.

Syntax: UNPROTECT [*worksheet1*::]*variable1* [, [*worksheet2*::]*variable2* ...]

Parameters: *worksheet1*, *worksheet2*—the names of the worksheets containing the variables. The name of the default global worksheet is **globals**, and a public worksheet has the same name as the corresponding part module.

variable1, *variable2*—the names of the protected variables. If no worksheet is specified, protection is removed from the first occurrence of the variable; any private worksheet is searched first, then the worksheets on the stack are searched in order.

See also: CLEAN, CREATE WORKSHEET, DESTROY, DESTROY WORKSHEET, GLOBAL, PUBLIC, PROTECT

USERNAME

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Function

Purpose: Returns the current user name.

Syntax: `USERNAME()`

Examples:
```
LET curruser = USERNAME()

LET string = "User: " & USERNAME()
```

See also: CPUNAME, HOSTNAME, OSCPUID, OSNAME, PLATFORM

2

This clause is used with the commands below. See the entries for those commands.

See: CHANNEL, FAX

USING

This clause is used with the commands listed below. See the entries for those commands.

See: DRAW BARCODE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT

VENDOR

1

This clause is used with the command below. See the entry for that command.

See: SQL

2

This keyword is part of the command below. See the entry for that command.

See: SET SQL VENDOR

VERSION

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Returns the version number of the OptioDCS software. For example, if your software is version 2.11.05, the version number is 2, the release number is 11, and the patch number is 05.

Syntax: `VERSION()`

Examples:
```
LET vernum = VERSION()

IF VERSION() > 2 THEN
```

See also: PATCH, RELEASE, SERIAL

VIA

This clause is used with the command below. See the entry for that command.

See: DRAW CURVE

VISIBLE

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Detects any non-printable characters in a string. Returns the same string with each non-printable character converted to an underbar, a backspace, and a printable character, so that the character is visible but appears underlined. For example, if the string contains a Ctrl-A character, it is converted to underbar-backspace-A, which prints as A. This function converts only the lower 32 ASCII characters.

Syntax: `VISIBLE(`*`string`*`)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Example: `LET inputdata = VISIBLE(@)`

See also: CANON, UNCANON, VISIBLEX

VISIBLEX

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Detects any non-printable characters in a string. Returns the same string with each non-printable character converted to an underbar, a backspace, and a printable character, so that the character is visible but appears underlined. For example, if the string contains a Ctrl-A character, it is converted to underbar-backspace-A, which prints as A. This function is extended to work with all 256 ASCII characters.

Syntax: `VISIBLEX(`*`string`*`)`

Parameters: *string*— a literal string, a variable containing a string or strings, or an expression that evaluates to a string

Example: `LET inputdata = VISIBLEX(@)`

See also: CANON, UNCANON, VISIBLE

WARNING

Doc Par Fun Dat Reg For Seg Cha Dev Fon

1

Type: Command

Purpose: Writes warning messages to a warning log file.

Syntax: `WARNING` *`message1`* `[,` *`message2`*`...]`

Parameters: *message1*, *message2*—variables or literal strings containing messages. A WARNING statement may have any number of messages.

See also: SET WARNING FILE, SET WARNING MASK

2

This clause is used with the commands listed below. See the entries for those commands.

See: SET WARNING FILE, SET WARNING MASK

WARNINGFILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the name of the warning file.

Syntax: `WARNINGFILE()`

Examples:
```
LET filename = WARNINGFILE()

LET string = "Log: " & WARNINGFILE()
```

See also: SET WARNING FILE, WARNING

WARNINGMASK

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Returns the current value of the warning mask, or converts a string of keywords to a valid mask value.

To return the current mask value, do not use a parameter. The value is returned as a (usually large) decimal number.

To convert keywords to a mask value, use a string of keywords as the parameter. If the string parameter is an array, the strings are concatenated and treated as one string.

Syntax:
```
WARNINGMASK()
WARNINGMASK(string)
```

Parameters: *string*— a literal string of warning mask keywords and plus or minus signs, a variable containing a string or strings, or an expression that evaluates to a string; keywords are listed below.

Keyword	Message Types
All	all warning messages
None	no warning messages
NotSupported	warns that a feature is not supported on the selected device
NotAuthorized	warns that the activity attempted requires an authorization key
CantDoThat	warns that a command is illegal
FileNotFound	warns that a referenced file cannot be found
VarNotFound	warns that a variable that is used does not exist
NonFatalErrors	warns that a miscellaneous error, that did not terminate the job, occurred
ParameterGone	warns that a parameter is missing from the .INI file, or the PARAMETER function was used with a non-existent parameter name
FileIOErrors	warns that an error occurred in reading or writing to a file
ChannelErrors	reserved for future use
FunctionErrors	warns that the wrong number of parameters are passed to a function
FontErrors	warns that a font that is used does not exist
ContainerErrors	warns that an error occurred in an internal list, stack, queue, or cache object
BMPErrors	warns that an error occurred when loading a .BMP file
GIFErrors	warns that an error occurred when loading a .GIF file
PCXErrors	warns that an error occurred when loading a .PCX file
TIFErrors	warns that an error occurred when loading a .TIF file
TIFWarnings	warns that an error occurred when loading a .TIF file, but it was corrected
InvalidBarcodes	reserved for future use

Examples:

```
LET lastmask = TOHEX(WARNINGMASK())

SET WARNING MASK WARNINGMASK("None +FontErrors +CantDoThat")

SET WARNING MASK WARNINGMASK("All -GIFErrors -TIFErrors")
```

See also: SET WARNING MASK, TOHEX, WARNINGFILE, WARNINGNAMES

WARNINGNAMES

Doc Par Fun Dat Reg For Seg Cha Dev Fon

Type: Function

Purpose: Converts a warning mask value to an array of keywords. See page 1-196 for a list of mask keywords.

Syntax: WARNINGNAMES(*maskvalue*)

Parameters: *maskvalue*— a constant, variable, or expression, that evaluates to a numerical value that is valid as a warning mask

Examples:
```
LET lastmask = WARNINGNAMES(WARNINGMASK())

SET WARNING MASK WARNINGMASK(WARNINGNAMES(newvalue))
```

See also: SET WARNING MASK, WARNINGMASK

WHEN

This clause is used with the commands listed below. See the entries for those commands.

See: CALL, DRAW, DRAW ARC, DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW CURVE, DRAW ELLIPSE, DRAW IMAGE, DRAW LINE, DRAW OBJECT, DRAW TEXT, MAP, PROCESS, RUN

WHERE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Function

Purpose: Searches a string (or array of strings) for a match of an expression; if a match is found, returns the starting column number (or index) of the first matching substring; otherwise, returns 0. If searching an array of strings, returns an array of the starting column numbers of the first matching substring in each element. The search expression can be a regular expression, which is a notation for describing patterns of characters (see your Windows NT or UNIX system documentation for more information).

Syntax: WHERE(*searchexpr*,*string2*)

Parameters: *searchexpr*— a literal string, a variable containing a string, or an expression that evaluates to a string, containing the expression to find

string— a literal string, a variable containing a string or strings, or an expression that evaluates to a string, containing the string to search for the expression

Examples:
```
LET descrcol = WHERE("Desc*",items[1])

WHILE WHERE(names[i],info[i,1:length]) = 4
```

See also: FIND, LOOKUP, MATCHES, SPAN

WHILE

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Command

Purpose: Executes a set of commands while a specified condition is true. The condition is tested, and if it is true, the commands between the WHILE command and the next END WHILE command are executed. This repeats until the condition is false or an EXIT WHILE command is executed within the set of commands.

Syntax: `WHILE condition commands END WHILE`

Parameters: *condition*— a Boolean expression specifying the condition for executing the *commands*.

commands— the commands that are executed repeatedly while the *condition* is true. The commands must be valid for use in the current module.
Note: Unless the condition is affected by the system time or date, the commands must affect the condition, or the WHILE execution will continue to repeat indefinitely.

Note: The WHILE statement must end with the END WHILE command.

See also: END, EXIT, FOR, IF, SWITCH

WIDTH

Doc □ Par □ Fun □ Dat □ Reg □ For ■ Seg ■ Cha □ Dev □ Fon □

Type: Clause

Purpose: Controls the width of an object.

Affects: bar code bounding boxes, boxes, circles, combs, ellipses, image bounding boxes, object bounding boxes, and text bounding boxes

Syntax: `WIDTH width`

Default: the bar code width for bar code bounding boxes;
0 for boxes, combs, ellipses, and circles;
the image width for image and object bounding boxes;
the text width for text bounding boxes

Parameters: *width*—a numeric expression indicating the width of the object, in the current units.

See also: DRAW BARCODE, DRAW BOX, DRAW COMB, DRAW ELLIPSE, DRAW IMAGE, DRAW OBJECT, DRAW TEXT, HEIGHT, SET, UNITS

WITH

This clause is used with the commands below. See the entries for those commands.

See: CALL, RUN

WORDWRAP

This clause is used with the command below. See the entry for that command.

See: DRAW TEXT

WORKSHEET

Doc ■ Par ■ Fun ■ Dat ■ Reg ■ For ■ Seg ■ Cha □ Dev □ Fon □

This clause is used with the commands listed below. See the entries for those commands.

See: CLEAN, CREATE WORKSHEET, DESTROY WORKSHEET, IGNORE WORKSHEET, SEARCH WORKSHEET, SET WORKSHEET

WRITE

Doc Par Fun Dat Reg For Seg Cha Dev Fon
■ ■ ■ ■ ■ ■ ■ □ □ □

Type: Command

Purpose: Writes the current set of data records (or page) to the specified output channel, usually a file. WRITE OUTPUT writes to the current output channel, or a channel may be specified.

Syntax: `WRITE {OUTPUT | "channelname"}`

Parameters: *channelname*—the name of a channel.

See also: CHANNEL, READ

Chapter 2

Printer Drivers

The principal printer drivers provided with OptioDCS are the PCL printer driver, the PostScript printer driver, and the ASCII text driver. These drivers were designed for printing a hard copy of your documents on a PCL, PostScript, or line printer. They are also useful for creating an HP-PCL file, an Encapsulated PostScript file, or an ASCII text file. These drivers offer the highest degree of output control, because they actually generate the printer commands for the final printed output.

OptioDCS also includes a PDF driver that produces Adobe Portable Document Format files, and an RTF driver that produces Rich Text Format files. PDF and RTF files can be used as E-mail attachments, or with other applications such as Optio e.ComPresent, Adobe Acrobat, or Microsoft Word.

To use a Windows NT print driver other than PCL, PostScript, or RTF, use a Graphics Device Interface, or GDI, device module. OptioDCS for Windows NT includes a program that creates GDI device modules. When using a GDI device module, OptioDCS does not generate the printer commands for the final printed output, so this method does not offer the same level of output control as the principal printer drivers.

This chapter describes the unique features of each of these drivers.

OptioDCS also provides drivers for several of the most popular label printers. See Chapter 3 in this book in this book for information on these drivers.

Using the PCL Printer Driver

OptioDCS provides a PCL printer driver that supports the PCL 4, PCL 5 and PCL 5 GL2 printer languages. This section contains information on the types of files to use for the DRAW OBJECT command, how to define and use PCL printer macros, and the barcodes that are supported by OptioDCS for PCL printers.

DRAW OBJECT Files

The DRAW OBJECT command allows you to include a page that was designed in another application in your OptioDCS document as an image. The PCL printer driver supports HP-PCL or text files as objects.

Using PCL Printer Macros

If your PCL printer supports macros, you may create macros of static text and graphic elements in your OptioDCS document. When macros are enabled, OptioDCS sends the macro elements to the printer as a macro, before processing the rest of the document. The macro is stored in the printer memory, which can reduce the time it takes to process your document. If macros are disabled, the macro elements are sent to the printer as normal elements of the format module.

Note: When using macros with an older PCL printer, you should see your printer documentation to verify that the printer supports embedding GL2 in a macro.

Enabling Macros

To enable macros in OptioDCS, use the following clause in the device module:

```
MACROS firstslot:lastslot
```

where *firstslot* and *lastslot* are the first and last printer memory slots available for OptioDCS macros. When a macro is downloaded into a memory slot, any information previously existing in that slot is deleted. For example, the clause below specifies that the memory slots numbered 51 through 60 are available for OptioDCS macros:

```
MACROS 51:60
```

This indicates that the device supports the use of macros, which automatically enables macros in OptioDCS.

Setting Macro Numbers

Set a range of memory slot numbers for the macros in each format using the MACROS header clause in the format module. For example, the clause below specifies that the memory slots numbered 57 through 59 may be used for the macros in the format:

```
MACROS 57:59
```

The macros in the format are automatically numbered consecutively, using this range. This range must fall within the range specified in the device module, and must not overlap any ranges specified in other format modules of the same document. If your format only contains one macro, you may specify a range of one number. For example, the header clause below specifies that the macro in the format should be number 58:

```
MACROS 58
```

When a macro is downloaded into a memory slot, any information previously existing in that slot is deleted.

Defining a Macro

To define a macro, use the following command in the format module:

```
SET MACRO "macroname"
```

where *macroname* is a name for the macro. To end macro definition, use the following command:

```
SET MACRO ""
```

These commands create a macro, including all DRAW statements between the two commands. Only text and graphic elements that should appear the same on every page should be placed in the macro.

During document processing, when OptioDCS encounters a particular macro for the first time (the first time it executes the format), it downloads the macro to the printer memory, and sends a command to the printer to execute the macro. Each time OptioDCS encounters the same macro again, it sends the execute command to the printer again; no downloading is necessary.

For example, the macro below prints a border made up of two boxes, and prints a header, made up of a line beneath the company name and address.

```
SET MACRO "staticstuff"
   DRAW BOX  AT .28,.28 TO 10.72,8.22 THICKNESS .015
   DRAW BOX  AT .30,.30 TO 10.70,8.20 THICKNESS .015
   DRAW LINE AT 1.0,1.0 TO 1.0,7.5 THICKNESS .02
   DRAW TEXT AT 0.9,1.0 USING "Optio Software" FONT "HELV14"
   DRAW TEXT AT 0.7,1.0 TO 0.7,7.5 FONT "HELV08"
      USING "4800 River Green Parkway" ALIGN "Right"
   DRAW TEXT AT 0.9,1.0 TO 0.9,7.5 "HELV08"
      USING "Duluth, Georgia  30096" ALIGN "Right"
SET MACRO ""
```

You may also specify that an individual text or graphic element is part of a macro by using the clause

```
MACRO "macroname"
```

in the DRAW statement. For example, the statements below define a macro called **header** that includes a line and text.

```
DRAW LINE AT 1.0,1.0 TO 1.0,7.5 THICKNESS .02 MACRO "header"
DRAW TEXT AT 0.9,1.0 USING "Optio" FONT "HELV14"
   MACRO "header"
```

This allows you to create a macro from statements located throughout a format module.

Macro Modes

You may use macros in any of the following modes:

- **Automatic** - The default mode is automatic. In automatic mode, when OptioDCS encounters a macro for the first time, it downloads the macro to the printer memory, and sends a command to the printer to execute the macro. Whenever OptioDCS encounters the same macro again, it sends the execute command to the printer again, so it executes the macro in memory. When document processing is complete, OptioDCS removes the macros from printer memory.

- **Download** - In download mode, OptioDCS searches the format modules for macros, and downloads any macros (without executing them). The macros are numbered in order of occurrence, using the range specified in the format module. It is important that all formats (in all documents) that send macros to a particular device have macro ranges that do not overlap. No other format commands are executed, so no output is generated in this mode.

  - If the output channel sends data to a printer, the macros are sent to printer memory. The macros remain in memory until they are removed by a command or the printer is turned off.

- If the output channel sends data to a file, the macros are written to the file, which can be used for creating a macro catridge. You may also manually download macros from the file to printer memory, flash memory, or hard disk.

- **Preloaded** - In preloaded mode, OptioDCS assumes that all macros have already been downloaded, so whenever a macro is encountered in a format module, it sends a command to the printer to execute the macro in memory. OptioDCS assumes that the macros are numbered in the order that they are executed, using the range specified in the format module.

If you are using the *ocserver* command from the command line to run OptioDCS, you may set the macro mode on the command line. To do this, add the switch below to the command.

```
-x MACROMODE=mode
```

where *mode* is a string indicating the mode, as listed below.

Mode	**Keywords**
automatic	"AUTOMATIC" "AUTO" "A"
download	"DOWNLOAD" "D"
preloaded	"PRELOADED" "P"

For example, the command below processes the document purchord.fgl in download mode.

```
ocserver -f ~\doc\purchord.dcl -x MACROMODE="DOWNLOAD"
```

You may also set and change the mode within the document using the command

```
SET MACROMODE mode
```

where *mode* is a string indicating the mode, as listed above. For example, the command below changes the macro mode to preloaded.

```
SET MACROMODE "P"
```

It is best to set the mode in the document module before the PROCESS PARTS or PROCESS command; the mode cannot be changed after the output channel is opened (when the first DRAW statement is executed in a format or segment).

Using DCL Language Extensions

By using the PCL extensions to the DCL language in your documents, you can take advantage of the unique features of the PCL driver. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of PCL output, such as barcode settings. Each extension command is described in the sections below.

Maxicode Extensions

For more information on Maxicode settings and requirements, see your printer documentation and the Maxicode specifications from UPS.

POSTAL_CODE

A clause for the DRAW BARCODE command, used to set the destination postal code. This clause is required for printing a Maxicode. The code must be 5 or 9 digits for delivery within the USA, and 6 digits for an international destination. The following example sets the postal code to the USA zip+4 code 30005-1234.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=300051234"
```

SERVICE_CLASS

A clause for the DRAW BARCODE command, used to set the class of service. Valid values are 001-999. The default value is 001. The following example sets the class of service to 002.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=300051234","SERVICE_CLASS=002"
```

COUNTRY_CODE

A clause for the DRAW BARCODE command, used to set the destination country by a country code. Valid values are 001-999. The default value is 840 (USA). The following example sets the country to 276.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=351234","MODE=3",
   "COUNTRY_CODE=276"
```

MODE

A clause for the DRAW BARCODE command, used to set the shipping mode. Valid values are: 2 for a destination within the USA and 3 for an international destination. The

default value is 2 (domestic shipping). The following example sets the mode to 3, indicating a destination outside the USA.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=351234”,"MODE=3",
   "COUNTRY_CODE=276"
```

SYMBOLS

A clause for the DRAW BARCODE command, used to specify the number of Maxicodes in a set. Valid values are 1-8. The default value is 1. The following example specifies a set of 4 Maxicodes.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SYMBOLS=4","SYMBOL=3"
```

SYMBOL

A clause for the DRAW BARCODE command, used to number the Maxicodes within a set. Valid values are 1-8. The default value is 1. The following example specifies the third Maxicode in a set of 4.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SYMBOLS=4","SYMBOL=3"
```

TRACE

A clause for the DRAW BARCODE command, used to enable tracing. This produces a trace file containing detailed information on how OptioDCS is generating the Maxicode symbol. To enable tracing, use the clause

```
EXTENSION “TRACE=TRUE”
```

Barcode Types Reference Table

The table on the following page lists the available barcode types in alphabetical order. For each type, the following information is provided:

- **Keyword** - the valid DCL keyword used to specify the barcode type
- **Characters** - the types of characters that are valid as data for the barcode type, indicated as follows:

  N numbers (0-9)
  U upper case letters (A-Z)
  L lower case letters (a-z)
  P punctuation
  C control characters (below the space character)

- **Length** - the number of characters or digits for the barcode data

- **Ratios** - the valid ratios of narrow bar width to wide bar width

For more information on a particular barcode, see your printer documentation.

Type	Keyword	Characters	Length	Ratios
Codabar	CODABAR	N [1]	2 to 30	3
Codabar + checksum	CODABAR+			
Code 2 of 5, interleaved	CODE25	N	2 to 30 [2]	2, 2.5, 3
Code 2 of 5, interleaved + checksum	CODE25+			
Code 3 of 9	CODE39	N, U, P	2 to 30	2, 2.5, 3
Code 3 of 9 + checksum	CODE39+			
Code 3 of 9 extended	CODE39X	N, U, L, P, C	2 to 30	2, 2.5, 3
Code 3 of 9 extended + checksum	CODE39X+			
Code 128 mode A + checksum	CODE128A+	N, U, P, C [3]	2 to 30	3
Code 128 mode B + checksum	CODE128B+	N, U, L, P [4]	2 to 30	3
Code 128 mode C + checksum	CODE128C+	N [5]	2 to 30	3
Code 128 Automatic[6] + checksum	CODE128+	N, U, L, P, C	2 to 30	3
Compressed code 3 of 9	CODE93	N, U, P [7]	2 to 30	2, 2.5, 3
Compressed code 3 of 9 + checksum	CODE93+			
Compressed code 3 of 9 extended	CODE93X	N, U, L, P, C	2 to 30	2, 2.5, 3
Compressed code 3 of 9 extended + checksum	CODE93X+			
EAN/JAN-8	EAN8	N	7, 9, or 12	3
EAN/JAN-13	EAN13	N	12, 14, or 17	3
MSI Plessey	MSIPLESSEY	N	2 to 15	2, 2.5, 3
MSI Plessey + checksum	MSIPLESSEY+			
POSTNET (ZIP+4) postal code[8]	POSTNET	N	5, 9, or 11	3
UCC-128 (Code128C)	UCC128	N	19	3
UCC-128 (Code128C) + checksum	UCC128+			
UPC Type A	UPCA	N	11, 13, or 16	3
UPC Type E	UPCE	N	11, 13, or 16	3

Special Instructions

[1]**Codabar** Valid characters are numerals and - $: / +
Start and end characters: A B C D

[2]**Code 2 of 5** The number of data characters must be even.

[3]**Code 128 mode A** Valid characters are the standard keyboard characters, and the following special characters:

NUL	BS	DLE	CAN	SHIFT	FNC1
SOH	HT	DC1	EM	BEL	FNC2
STX	LF	DC2	SI	SUB	FNC3
ETX	VT	DC3	US	ETB	FNC4
EOT	FF	DC4	FS	ESC	
ENQ	CR	NAK	GS	CODEB	
ACK	SO	SYN	RS	CODEC	

[4]**Code 128 mode B** Valid characters are the standard keyboard characters, and the following special characters:

CODEA	FNC2
CODEC	FNC3
FNC1	FNC4

[5]**Code 128 mode C** Valid characters are the set of 100 digit pairs from 00 to 99, and the following special characters:

CODEA
CODEB
FNC1

[6]**Code 128 automatic** Automatically selects mode A, B, or C for optimal barcode size.

[7]**Compressed Code 3 of 9** Valid characters are numerals, capital letters, the space character, and - . $ / + %

[8]**Postnet** These barcodes do not have human readable text.

Using the PostScript Printer Driver

OptioDCS provides a PostScript printer driver that supports the PostScript level 1 and level 2 printer languages. This section contains information on the types of files to use for the DRAW OBJECT command, how to use PostScript ISO Latin1 Encoding, and the barcodes that are supported by OptioDCS for PostScript printers.

DRAW OBJECT Files

The DRAW OBJECT command allows you to include a page that was designed in another application in your OptioDCS document as an image. The PostScript printer driver supports Encapsulated PostScript (EPS) files as objects.

Using PostScript ISO Latin1 Encoding

If your printer supports PostScript level 1 or level 2 ISO Latin1 Encoding, you can print text using this encoding. ISO Latin1 Encoding allows you to use the characters used in Latin-based languages, such as Spanish and French. To use this encoding in your OptioDCS documents, you must use a modified Postscript device file and set the encoding for the DRAW TEXT statements in the format or segment modules.

Modifying the Device

To use ISO Latin1 Encoding in an OptioDCS document, you must use a Postscript device file that is modified to allow encoding. To modify a device file for encoding, add the clauses

```
SYMBOLS "StandardEncoding"
SYMBOLS "ISOLatin1Encoding"
```

to the end of each FONT statement. These clauses make encoding available for the font. For example, the font statement below makes ISO Latin1 encoding available for the **Helvetica Bold** font.

```
FONT "Helvetica Bold" TYPEFACE "Helvetica-Bold"
   METRICS "~afm/hvb_____.afm"
   SYMBOLS "AdobeStandardEncoding"
   SYMBOLS "StandardEncoding"
   SYMBOLS "ISOLatin1Encoding"
```

Encoding Text

To encode text in your format or segment modules, you must set the symbol set to ISO Latin1 Encoding and select the font for the text by the typeface and point size. To do this for a single text string, add the following clauses to each DRAW TEXT statement.

```
TYPEFACE "typeface" POINTS points
SYMBOLS "ISOLatin1Encoding"
```

where typeface and points are the desired typeface and points. Do not use the FONT clause to set the font. For example, the statement below prints the contents of **itemname** using the Helvetica-Bold typeface at 12 points, using the ISO Latin1 encoding.

```
DRAW TEXT AT 2.2,1.1 USING itemname
   TYPEFACE "Helvetica-Bold" POINTS 12
   SYMBOLS "ISOLatin1Encoding"
```

To print mutiple text strings using the encoding, set the points, typeface, and symbols using SET statements. For example, the statements below print the contents of **itemname**, **itemnum**, and **itemcost** in the Helvetica-Bold typeface at 12 points, using the ISO Latin1 encoding.

```
SET TYPEFACE "Helvetica-Bold"
SET POINTS 12
SET SYMBOLS "ISOLatin1Encoding"

DRAW TEXT AT 2.2,1.1 USING itemname
DRAW TEXT AT 2.5,1.1 USING itemnum
DRAW TEXT AT 2.8,1.1 USING itemcost
```

To input a character that is not on your keyboard, use the CANON function with the ASCII number of the character. For example, to print "Café", use

```
CANON("Caf\233")
```

because the ASCII number of "é" is 233.

Auto-downloadable Fonts

If your PostScript level 2 device supports auto-downloadable fonts, you can add them to the to the PostScript level 2 device module. The first time a document uses an auto-downloadable font, OptioDCS automatically downloads the glyph definition file into PDF file. This slows document processing and creates a larger PDF file.

To add a font, add a new FONT command to the device module to specify the characteristics of the new font. For each auto-downloadable font, you must have an Adobe font metrics (.afm) file and an Adobe PostScript FontType 1 glyph definition (.pfb) file (IBM type). Specify the glyph definition file using the clause

```
GLYPHS "glyphFilePath"
```

where *glyphFilePath* is the path and file name of the glyph definition file. For example, the following statement defines the DAmaze-Bold font, using the ~afm/Amaze.afm font metrics file, and the ~pfb/Amaze.pfb glyph definition file.

```
FONT "Damaze Bold" TYPEFACE "DAmaze-Bold"
   SYMBOLS "AdobeStandardEncoding"
   METRICS "~afm/Amaze.afm"
   GLYPHS "~pfb/Amaze.pfb"
```

For more information on the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Using DCL Language Extensions

By using the PostScript extensions to the DCL language in your documents, you can take advantage of the unique features of the PostScript driver. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of PostScript output, such as barcode settings. Each extension command is described in the sections below.

Maxicode Extensions

For more information on Maxicode settings and requirements, see your printer documentation and the Maxicode specifications from UPS.

POSTAL_CODE

A clause for the DRAW BARCODE command, used to set the destination postal code. This clause is required for printing a Maxicode. The code must be 5 or 9 digits for delivery within the USA, and 6 digits for an international destination. The following example sets the postal code to the USA zip+4 code 30005-1234.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”
```

SERVICE_CLASS

A clause for the DRAW BARCODE command, used to set the class of service. Valid values are 001-999. The default value is 001. The following example sets the class of service to 002.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SERVICE_CLASS=002"
```

COUNTRY_CODE

A clause for the DRAW BARCODE command, used to set the destination country by a country code. Valid values are 001-999. The default value is 840 (USA). The following example sets the country to 276.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=351234”,"MODE=3",
   "COUNTRY_CODE=276"
```

MODE

A clause for the DRAW BARCODE command, used to set the shipping mode. Valid values are: 2 for a destination within the USA and 3 for an international destination. The default value is 2 (domestic shipping). The following example sets the mode to 3, indicating a destination outside the USA.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=351234”,"MODE=3",
   "COUNTRY_CODE=276"
```

SYMBOLS

A clause for the DRAW BARCODE command, used to specify the number of Maxicodes in a set. Valid values are 1-8. The default value is 1. The following example specifies a set of 4 Maxicodes.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SYMBOLS=4","SYMBOL=3"
```

SYMBOL

A clause for the DRAW BARCODE command, used to number the Maxicodes within a set. Valid values are 1-8. The default value is 1. The following example specifies the third Maxicode in a set of 4.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=300051234","SYMBOLS=4","SYMBOL=3"
```

TRACE

A clause for the DRAW BARCODE command, used to enable tracing. This produces a trace file containing detailed information on how OptioDCS is generating the Maxicode symbol. To enable tracing, use the clause

```
EXTENSION "TRACE=TRUE"
```

Barcode Types Reference Table

The table on the following page lists the available barcode types in alphabetical order. For each type, the following information is provided:

- **Keyword** - the valid DCL keyword used to specify the barcode type
- **Characters** - the types of characters that are valid as data for the barcode type, indicated as follows:

  N numbers (0-9)
  U upper case letters (A-Z)
  L lower case letters (a-z)
  P punctuation
  C control characters (below the space character)

- **Length** - the number of characters or digits for the barcode data
- **Ratios** - the valid ratios of narrow bar width to wide bar width

For more information on a particular barcode, see your printer documentation.

Type	Keyword	Characters	Length	Ratios
Codabar	CODABAR	N [1]	2 to 30	3
Codabar + checksum	CODABAR+			
Code 2 of 5, interleaved	CODE25	N	2 to 30 [2]	2, 2.5, 3
Code 2 of 5, interleaved + checksum	CODE25+			
Code 3 of 9	CODE39	N, U, P	2 to 30	2, 2.5, 3
Code 3 of 9 + checksum	CODE39+			
Code 3 of 9 extended	CODE39X	N, U, L, P, C	2 to 30	2, 2.5, 3
Code 3 of 9 extended + checksum	CODE39X+			
Code 128 mode A + checksum	CODE128A+	N, U, P, C [3]	2 to 30	3
Code 128 mode B + checksum	CODE128B+	N, U, L, P [4]	2 to 30	3
Code 128 mode C + checksum	CODE128C+	N [5]	2 to 30	3
Code 128 Automatic[6] + checksum	CODE128+	N, U, L, P, C	2 to 30	3
Compressed code 3 of 9	CODE93	N, U, P [7]	2 to 30	2, 2.5, 3
Compressed code 3 of 9 + checksum	CODE93+			
Compressed code 3 of 9 extended	CODE93X	N, U, L, P, C	2 to 30	2, 2.5, 3
Compressed code 3 of 9 extended + checksum	CODE93X+			
EAN/JAN-8	EAN8	N	7, 9, or 12	3
EAN/JAN-13	EAN13	N	12, 14, or 17	3
MSI Plessey	MSIPLESSEY	N	2 to 15	2, 2.5, 3
MSI Plessey + checksum	MSIPLESSEY+			
POSTNET (ZIP+4) postal code[8]	POSTNET	N	5, 9, or 11	3
UCC-128 (Code128C)	UCC128	N	19	3
UCC-128 (Code128C) + checksum	UCC128+			
UPC Type A	UPCA	N	11, 13, or 16	3
UPC Type E	UPCE	N	11, 13, or 16	3

Special Instructions

[1]**Codabar** Valid characters are numerals and - $: / +
Start and end characters: A B C D

[2]**Code 2 of 5** The number of data characters must be even.

[3]**Code 128 mode A** Valid characters are the standard keyboard characters, and the following special characters:

NUL	BS	DLE	CAN	SHIFT	FNC1
SOH	HT	DC1	EM	BEL	FNC2
STX	LF	DC2	SI	SUB	FNC3
ETX	VT	DC3	US	ETB	FNC4
EOT	FF	DC4	FS	ESC	
ENQ	CR	NAK	GS	CODEB	
ACK	SO	SYN	RS	CODEC	

[4]**Code 128 mode B** Valid characters are the standard keyboard characters, and the following special characters:

CODEA	FNC2
CODEC	FNC3
FNC1	FNC4

[5]**Code 128 mode C** Valid characters are the set of 100 digit pairs from 00 to 99, and the following special characters:

CODEA
CODEB
FNC1

[6]**Code 128 automatic** Automatically selects mode A, B, or C for optimal barcode size.

[7]**Compressed Code 3 of 9** Valid characters are numerals, capital letters, the space character, and - . $ / + %

[8]**Postnet** These barcodes do not have human readable text.

Using the ASCII Text Driver

OptioDCS includes an ASCII text driver and device module, allowing you to print to a text file or line printer. This section explains how to use this driver, and contains guidelines to consider when designing a document for ASCII output.

Setting Up the ASCII Text Device Module

An ASCII text device file named **demotext** is provided with OptioDCS. You can use this device, or create your own text device file by copying this file and customizing the copy.

Paper Sizes

The ASCII text device module contains commands similar to the following:

```
LANGUAGE "Text"
UNITS "Characters"

PAPER "Letter", "", 66, 132, 0,0,0,0
PAPER "Legal", "", 88, 132, 0,0,0,0
```

Do not change the LANGUAGE and UNITS statements.

The PAPER commands define names for various paper sizes, such as "Legal". The sizes are measured in rows and columns. If you want to use a paper size other than the sizes already specified, you may add a new PAPER command, defining a name for the paper size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width, 0,0,0,0
```

where *name* is the name for the new paper size, *height* is the height in rows, and *width* is the width in columns.

Fonts

The ASCII text device module does not support fonts. This device produces plain text in rows and columns. Bold, italicized, and underlined text are not supported.

Document Guidelines

The Document or Part Module

To produce ASCII text from your OptioDCS document, you must select the ASCII text device in your document or part module. You must also select an appropriate output channel as follows:

- If you want to send the document to a text file, select an output channel module that sends to a text file using the FILE and MODE clauses.
- If you want to send the document to a line printer, select an output channel module that sends to a printer queue using the COMMAND clause (for UNIX) or the PRINTER clause (for Window NT).

For more information on creating channels for ASCII text, see Chapter 4 of the *Advanced Features Guide*.

The Format Module

There are limitations for certain commands and settings when producing ASCII text output. When designing your format for ASCII output, use the following guidelines:

- The units for the format module must be "Characters", and all text is placed by rows and columns. For example, the statement below places text with the first character at the third row and the tenth column.

```
DRAW TEXT USING inv_num AT 3, 10
```

- All text is produced in a fixed-width format. Fonts, bold text, italicized text, and underlined text are not supported.
- Any linefeeds or carriage returns indicated by the output channel are added to the end of each line of text. Any areas of the page without data are filled with spaces.
- No DRAW commands are supported except DRAW TEXT, DRAW LINE, and DRAW BOX.
  DRAW BOX can be used to print a rectangle, and DRAW LINE can be used to print a horizontal or vertical line, as supported by your printer; the ASCII driver forms basic rectangles and lines from ASCII characters, such as hyphens; no line styles, line thicknesses, or fill patterns are supported.

Using the PDF Driver

OptioDCS includes a PDF driver and device module, allowing you to print to an Adobe Portable Document Format file. This file can be used as an E-mail attachment, publishing on the World Wide Web, or with other applications such as Optio e.ComPresent. These files can be viewed and printed using Adobe Acrobat Reader version 2.0 or higher, or Optio e.ComPresent.

This section explains how to use this driver, and contains guidelines to consider when designing a document for PDF output. It explains how to take advantage of the unique features of PDF files using the PDF extensions to the DCL language in your documents. It also includes a reference of barcodes supported by OptioDCS for PDF output. For more information on creating PDF files, see Chapter 4 of the *Advanced Features Guide*.

Setting Up the PDF Device Module

OptioDCS provides a PDF device module that you can select in your OptioDCS documents in order to produce PDF output. You can create your own PDF device module by copying the provided device file and modifying your copy.

Color Palette

The PDF device module contains a palette module that defines some basic colors for use in PDF documents. You can change the available colors by editing the palette module. For more information on using color in OptioDCS documents, see Chapter 2 of the *Advanced Features Guide*.

Standard Fonts

The PDF device module includes the following standard PDF fonts:

Courier	Courier-Bold	Courier-Oblique	Courier-BoldOblique
Helvetica	Helvetica-Bold	Helvetica-Oblique	Helvetica-BoldOblique
Times-Roman	Times-Bold	Times-Italic	Times-BoldItalic
Symbols			
ZapfDingbats			

It is recommended that you use these fonts in your document formats. For example, the statement below prints text using the Helvetica-Bold font.

```
DRAW TEXT AT 3.5, 1.0  USING "Order Form"
   TYPEFACE "Helvetica-Bold"
```

You must have an Adobe font metrics (.afm) file for each font that you use in your document. For more information, see Chapter 8 of the *Advanced Features Guide*.

Auto-downloadable Fonts

You can add auto-downloadable fonts to the device module. The first time a document uses an auto-downloadable font, OptioDCS automatically downloads the glyph definition file into PDF file. This slows document processing and creates a larger PDF file.

To add a font, add a new FONT command to specify the characteristics of the new font. For each auto-downloadable font, you must have an Adobe font metrics (.afm) file and an Adobe PostScript FontType 1 glyph definition (.pfb) file (IBM type). Specify the glyph definition file using the clause

```
GLYPHS "glyphFilePath"
```

where *glyphFilePath* is the path and file name of the glyph definition file. For example, the following statement defines the DAmaze-Bold font, using the ~afm/Amaze.afm font metrics file, and the ~pfb/Amaze.pfb glyph definition file.

```
FONT "Damaze Bold" TYPEFACE "DAmaze-Bold"
   SYMBOLS "AdobeStandardEncoding"
   METRICS "~afm/Amaze.afm"
   GLYPHS "~pfb/Amaze.pfb"
```

For more information on the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Document Guidelines

The Document or Part Module

To print your OptioDCS document to a PDF file, you must select the PDF device in your document or part module. You must also select an appropriate output channel as follows:

- If you want to send the document as an attachment to an E-mail message, select an output channel module that sends data to a MAPI or SMTP server. For information on creating this type of channel, see Chapter 3 of the *Advanced Features Guide*.

- If you want to produce a PDF file for use with other applications such as Optio e.ComPresent or Adobe Acrobat Reader, select an output channel that sends data to a file. Use the extension **.pdf** for the file name. For information on creating this type of channel, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

The Format Module

There are limitations for certain commands and settings when producing PDF output, as described in the following sections.

DUPLEX
The DUPLEX clause is not supported.

FEEDER
The FEEDER clause is not supported.

LINESTYLE
All linestyles are supported and print correctly. However, when viewing PDF files at 100% in Adobe Acrobat or another viewer, most linestyles, such as DashDot, appear very similar. The scaled linestyles are more distinct than the others, and the styles of lines with a thickness of .04 inches or greater are more distinct.

PATTERN
The stripe and crosshatch patterns are not supported. The PATTERN clause setting affects boxes, circles, combs, ellipses, text bounding boxes, and object shadows.

STACKER
The STACKER clause is not supported.

SURFACE
The SURFACE clause is not supported.

DRAW BARCODE
The ALIGN clause is not supported.

DRAW BLOB
The DRAW BLOB command is not supported.

DRAW TEXT
You must select a standard PDF font or an auto-downloadable font that is specified in the PDF device module or in a fontpack called by that module. You must have an Adobe font metrics file for each font. For more information, see page 2-19.

The FIELD clause is supported. Using this clause creates a PDF field (containing the text specified by the USING clause), for use in other applications, such as Optio

e.ComPresent and Adobe Acrobat. For example, the statement below creates a field called **CustNumber**, which contains the value of the variable **custnum**.

```
DRAW TEXT AT 4.2,1.1 USING custnum FIELD "CustNumber"
```

Using DCL Language Extensions

By using the PDF extensions to the DCL language in your documents, you can take advantage of the unique features of PDF files. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of viewing PDF output, such as hyperlinks and annotations. Each extension command is described in the sections below.

Bar Code Extensions

NOTE

A clause for the DRAW BARCODE command, used to place a PDF annotation at the position specified by the AT clause. Annotations appear only when viewing PDF files; they are not printed with the file. For example, the following command places an annotation containing the text "includes tax" at 1 inch down and 2 inches across.

```
DRAW BARCODE AT 1,2 EXTENSION "NOTE=includes tax" USING pr
```

By default, each annotation is initially closed and displayed as an icon when a user views the PDF file. To have an annotation initially open, use the extension "OPEN=TRUE", as shown in the following example.

```
DRAW BARCODE AT 1,2 EXTENSION "NOTE=includes tax",
"OPEN=TRUE" USING pr
```

MAP

A clause for the DRAW BARCODE command LINK clause, used to include the location of the mouse cursor when a user clicks a mouse button within the hyperlink area. The coordinates of the cursor are the distance from the top left corner of the hyperlink area in 72 dpi. The horizontal value is listed first. The vertical value is always negative.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site that will include the cursor coordinates when a user clicks within the bar code bounding box.

```
DRAW BARCODE AT 1,2 USING itemcode
   LINK “http://www.optiosoftware.com/” EXTENSION "MAP=TRUE"
```

If the user clicks this hyperlink 32/72 inch from the left edge of the bar code bounding box, and 15/72 inch from the top edge, the PDF viewer sends the URL “http://www.optiosoftware.com/?32,-15” to the user’s default browser.

The default value is FALSE.

BORDER

A clause for the DRAW BARCODE command LINK clause, used to display a border around the hyperlink area. The border thickness is specified in 72 dpi. Hyperlink borders appear only when viewing PDF files; they are not printed with the file.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site with a border that is 1/12 of an inch thick.

```
DRAW BARCODE AT 1,2 USING itemcode
   LINK “http://www.optiosoftware.com/” EXTENSION "BORDER=6"
```

The default value is 0 (no border).

Image Extensions

NOTE

A clause for the DRAW IMAGE command, used to place a PDF annotation at the position specified by the AT clause. Annotations appear only when viewing PDF files; they are not printed with the file. For example, the following command places an annotation containing the text “includes tax” at 1 inch down and 2 inches across.

```
DRAW IMAGE AT 1,2 EXTENSION "NOTE=includes tax" USING logo
```

By default, each annotation is initially closed and displayed as an icon when a user views the PDF file. To have an annotation initially open, use the extension “OPEN=TRUE”, as shown in the following example.

```
DRAW IMAGE AT 1,2 EXTENSION "NOTE=includes tax", “OPEN=TRUE”
   USING logo
```

MAP

A clause for the DRAW IMAGE command LINK clause, used to include the location of the mouse cursor when a user clicks a mouse button within the hyperlink area. The coordinates of the cursor are the distance from the top left corner of the hyperlink area in 72 dpi. The horizontal value is listed first. The vertical value is always negative.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site that will include the cursor coordinates when a user clicks within the image bounding box.

```
DRAW IMAGE AT 1,2 USING logo
   LINK “http://www.optiosoftware.com/” EXTENSION "MAP=TRUE"
```

If the user clicks this hyperlink 32/72 inch from the left edge of the image bounding box, and 15/72 inch from the top edge, the PDF viewer sends the URL “http://www.optiosoftware.com/?32,-15” to the user’s default browser.

The default value is FALSE.

BORDER

A clause for the DRAW IMAGE command LINK clause, used to display a border around the hyperlink area. The border thickness is specified in 72 dpi. Hyperlink borders appear only when viewing PDF files; they are not printed with the file.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site with a border that is 1/12 of an inch thick.

```
DRAW IMAGE AT 1,2 USING logo
   LINK “http://www.optiosoftware.com/” EXTENSION "BORDER=6"
```

The default value is 0 (no border).

Text Extensions

NOTE

A clause for the DRAW TEXT command, used to place a PDF annotation at the position specified by the AT clause. Annotations appear only when viewing PDF files; they are not printed with the file. For example, the following command places an annotation containing the text “includes tax” at 1 inch down and 2 inches across.

```
DRAW TEXT AT 1,2 EXTENSION "NOTE=includes tax" USING ""
```

By default, each annotation is initially closed and displayed as an icon when a user views the PDF file. To have an annotation initially open, use the extension “OPEN=TRUE”, as shown in the following example.

```
DRAW TEXT AT 1,2 EXTENSION "NOTE=includes tax", “OPEN=TRUE”
   USING ""
```

MAP

A clause for the DRAW TEXT command LINK clause, used to include the location of the mouse cursor when a user clicks a mouse button within the hyperlink area. The coordinates of the cursor are the distance from the top left corner of the hyperlink area in 72 dpi. The horizontal value is listed first. The vertical value is always negative.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site that will include the cursor coordinates when a user clicks within the text bounding box.

```
DRAW TEXT AT 1,2 USING "click here"
   LINK “http://www.optiosoftware.com/” EXTENSION "MAP=TRUE"
```

If the user clicks this hyperlink 32/72 inch from the left edge of the text bounding box, and 15/72 inch from the top edge, the PDF viewer sends the URL “http://www.optiosoftware.com/?32,-15” to the user’s default browser.

The default value is FALSE.

BORDER

A clause for the DRAW TEXT command LINK clause, used to display a border around the hyperlink area. The border thickness is specified in 72 dpi. Hyperlink borders appear only when viewing PDF files; they are not printed with the file.

For example, the command below creates a hyperlink to the Optio Software World Wide Web site with a border that is 1/12 of an inch thick.

```
DRAW TEXT AT 1,2 USING "click here"
   LINK “http://www.optiosoftware.com/” EXTENSION "BORDER=6"
```

The default value is 0 (no border).

Barcode Types Reference Table

The table on the following page lists the available barcode types in alphabetical order. For each type, the following information is provided:

- **Keyword** - the valid DCL keyword used to specify the barcode type
- **Characters** - the types of characters that are valid as data for the barcode type, indicated as follows:

  N numbers (0-9)
  U upper case letters (A-Z)
  L lower case letters (a-z)
  P punctuation
  C control characters (below the space character)

- **Length** - the number of characters or digits for the barcode data
- **Ratios** - the valid ratios of narrow bar width to wide bar width

For more information on a particular barcode, see your printer documentation.

Type	Keywords	Characters	Length	Ratios
Codabar	CODABAR	N [1]	2 to 30	3
Codabar + checksum	CODABAR+			
Code 2 of 5, interleaved	CODE25	N	2 to 30 [2]	2, 2.5, 3
Code 2 of 5, interleaved + checksum	CODE25+			
Code 3 of 9	CODE39	N, U, P	2 to 30	2, 2.5, 3
Code 3 of 9 + checksum	CODE39+			
Code 3 of 9 extended	EXTEND39	N, U, L, P, C	2 to 30	2, 2.5, 3
Code 3 of 9 extended + checksum	CODE39X+			
Code 128 mode A + checksum	CODE128A+	N, U, P, C [3]	2 to 30	3
Code 128 mode B + checksum	CODE128B+	N, U, L, P [4]	2 to 30	3
Code 128 mode C + checksum	CODE128C+	N [5]	2 to 30	3
Code 128 Automatic[6] + checksum	CODE128+	N, U, L, P, C	2 to 30	3
Compressed code 3 of 9	CODE93	N, U, P [7]	2 to 30	2, 2.5, 3
Compressed code 3 of 9 + checksum	CODE93+			
Compressed code 3 of 9 extended	EXTEND93	N, U, L, P, C	2 to 30	2, 2.5, 3
Compressed code 3 of 9 extended + checksum	EXTEND93+			
EAN/JAN-8	EAN8	N	7, 9, or 12	3
EAN/JAN-13	EAN13	N	12, 14, or 17	3
MSI Plessey	MSIPLESSEY	N	2 to 15	2, 2.5, 3
MSI Plessey + checksum	MSIPLESSEY+			
POSTNET (ZIP+4) postal code[8]	POSTNET	N	5, 9, or 11	3
UCC-128 (Code128C)	UCC128	N	19	3
UCC-128 (Code128C) + checksum	UCC128+			
UPC Type A	UPCA	N	11, 13, or 16	3
UPC Type E	UPCE	N	11, 13, or 16	3

Special Instructions

[1]**Codabar** Valid characters are numerals and - $: / +
Start and end characters: A B C D

[2]**Code 2 of 5** The number of data characters must be even.

[3]**Code 128 mode A** Valid characters are the standard keyboard characters, and the following special characters:

NUL	BS	DLE	CAN	SHIFT	FNC1
SOH	HT	DC1	EM	BEL	FNC2
STX	LF	DC2	SI	SUB	FNC3
ETX	VT	DC3	US	ETB	FNC4
EOT	FF	DC4	FS	ESC	
ENQ	CR	NAK	GS	CODEB	
ACK	SO	SYN	RS	CODEC	

[4]**Code 128 mode B** Valid characters are the standard keyboard characters, and the following special characters:

CODEA	FNC2
CODEC	FNC3
FNC1	FNC4

[5]**Code 128 mode C** Valid characters are the set of 100 digit pairs from 00 to 99, and the following special characters:

CODEA
CODEB
FNC1

[6]**Code 128 automatic** Automatically selects mode A, B, or C for optimal barcode size.

[7]**Compressed Code 3 of 9** Valid characters are numerals, capital letters, the space character, and - . $ / + %

[8]**Postnet** This barcode does not have human readable text.

Using the RTF Driver

OptioDCS includes an RTF driver and device module, allowing you to print to a Rich Text Format file. This file may be used as an E-mail attachment, or with other applications such as Optio e.ComPresent or Microsoft Word. Output from the RTF driver is based on version 1.5 of RTF specifications published by Microsoft. RTF 1.5 files can be read by Microsoft Word 8.0/97 and Microsoft Word Viewer 97, in addition to Optio e.ComPresent.

This section explains how to use this driver, and contains guidelines to consider when designing a document for RTF output. It also includes a reference of barcodes supported by OptioDCS for RTF output.

Setting Up the RTF Device Module

Fonts

The RTF device module provides access to many popular TrueType fonts. To add a new TrueType font, add a new FONT command to specify the characteristics of the new font. You may also use this command to specify a new font size for an existing font. For instructions on using the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Paper Sizes

The RTF device module contains commands similar to the following:

```
LANGUAGE "RTF"
UNITS "DPI300"

PAPER ".Default", "", 3300, 2550, 0, 60, 0,0
PAPER "Executive", "", 3150, 2175, 0, 60, 0,0
PAPER "Letter", "", 3300, 2550, 0, 60, 0,0
PAPER "Legal", "", 4200, 2550, 0, 60, 0,0
PAPER "A4", "", 3507, 2480, 0, 59, 0,0
PAPER "Monarch", "", 2250, 1162, 0, 60, 0,0
PAPER "Com10", "", 2850, 1237, 0, 60, 0,0
PAPER "DL", "", 2598, 1299, 0, 59, 0,0
PAPER "C5", "", 2704, 1913, 0, 59, 0,0
PAPER "B5", "", 2952, 2078, 0, 59, 0,0
```

The PAPER commands define names for various paper sizes, such as "Legal". The sizes are measured in the units specified by the UNITS command. If you want to use a paper size other than the sizes already specified, you may add a new PAPER command,

defining a name for the paper size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width, 0, offset_h, 0,0
```

where *name* is the name for the new paper size, *height* is the height in units, *width* is the width in units, and *offset_h* is the horizontal offset of the logical origin (0,0) from physical left edge of the paper in units.

Document Guidelines

The Document or Part Module

To print your OptioDCS document to an RTF file, you must select the RTF device in your document or part module. You must also select an appropriate output channel as follows:

- If you want to send the document as an attachment to an E-mail message, select an output channel module that sends data to an SMTP server. For information on creating this type of channel, see Chapter 3 of the *Advanced Features Guide*.

- If you want to produce an RTF file for use with other applications such as Optio e.ComPresent or Microsoft Word, select an output channel that sends data to a file. For information on creating this type of channel, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

The Format Module

When designing your format for RTF output, use the following guidelines:

- All filled objects are opaque, regardless of the OPAQUE setting. This means that the order of the DRAW statements is important; if a filled object overlaps other objects, it must be drawn before them or it will conceal them.

- It is recommended that you sort your DRAW statements by type, so that all of the DRAW LINE statements are together, all of the DRAW BOX statements are together, and so on. Place the DRAW TEXT commands at the end of the format module.

- If you save your document to an RTF file, you may need to adjust certain aspects, such as margins, when you open the RTF file in other applications. Consider the finished margins when designing the format.

- The appearance of the RTF file may vary when opened in different Windows applications. If an application does not support RTF graphics, only the text of your document will appear. Also, some applications do not support RTF text formatting.

There are limitations for certain commands and settings when producing RTF output, as described in the following sections.

FEEDER
The FEEDER clause is not supported.

STACKER
The STACKER clause is not supported.

DRAW ARC
The DRAW ARC command is not supported.

DRAW BARCODE
Rotations of 90º and 270º are not supported for barcodes.

Barcodes may not display correctly in applications other than Optio e.ComPresent.

DRAW CURVE
The DRAW CURVE command is not supported.

DRAW LINE
The LINECAP clause is not supported; all line caps are squared.

For the LINESTYLE clause, only five styles are supported, as listed below:

Linestyle	Keywords		
solid	S	SOLID	
dash	F1	FIXED1	Dash
dot	F2	FIXED2	Dot
dash-dot	F3	FIXED3	DashDot
dash-dot-dot	F4	FIXED4	DashDotDot

If another keyword is used with the LINESTYLE clause, one of the supported styles is used, as listed below.

Specified Keywords			Resulting Linestyle
F5	FIXED5		dash
F6	FIXED6		dot
F7	FIXED7		dash-dot
F8	FIXED8		dash
SCALED1	A1	ADAPT1	dash
SCALED2	A2	ADAPT2	dot
SCALED3	A3	ADAPT3	dash-dot
SCALED4	A4	ADAPT4	dash-dot-dot
SCALED5	A5	ADAPT5	dash
SCALED6	A6	ADAPT6	dot
SCALED7	A7	ADAPT7	dash-dot
SCALED8	A8	ADAPT8	dash

DRAW OBJECT

The DRAW OBJECT command is not supported.

DRAW TEXT

You must select a TrueType font that is specified in the RTF device module or in a fontpack called by that module. You must have a TrueType metrics file for each font.

Text with a rotation of 180º may not display correctly in applications other than Optio e.ComPresent.

The FIELD clause is supported. Using this clause creates an RTF field (containing the text specified by the USING clause), for use in other applications, such as Optio e.ComPresent and Microsoft Word. For example, the statement below creates a field called **CustNumber**, which contains the value of the variable **custnum**.

```
DRAW TEXT AT 4.2,1.1 USING custnum FONT AR12
   FIELD "CustNumber"
```

PATTERN

The PATTERN clause setting affects boxes, circles, combs, ellipses, text bounding boxes, and object shadows.

All of the striped fill patterns are light. To produce darker striped patterns, add the clause

```
EXTENSION "FILL=DARK"
```

to the DRAW command.

The grey scale percentages are slightly different, as listed below:

Pattern	Keywords
0% grey	G1 GREY1
5% grey	G2 GREY2
20% grey	G3 GREY3
25% grey	G4 GREY4
40% grey	G5 GREY5
70% grey	G6 GREY6
90% grey	G7 GREY7

To produce other grey scale percentages, add the clause

```
EXTENSION "FILL=DARK"
```

to the DRAW command, using the keywords as listed below:

Pattern	Keywords
10% grey	G2 GREY2
30% grey	G4 GREY4
50% grey	G5 GREY5
80% grey	G6 GREY6

Barcode Types Reference Table

The table on the following page lists the available barcode types in alphabetical order. For each type, the following information is provided:

- **Keyword** - the valid DCL keyword used to specify the barcode type
- **Characters** - the types of characters that are valid as data for the barcode type, indicated as follows:

  N numbers (0-9)
  U upper case letters (A-Z)
  L lower case letters (a-z)
  P punctuation
  C control characters (below the space character)

- **Length** - the number of characters or digits for the barcode data
- **Ratios** - the valid ratios of narrow bar width to wide bar width

For more information on a particular barcode, see your printer documentation.

Type	Keywords	Characters	Length	Ratios
Codabar	CODABAR	N [1]	2 to 30	3
Codabar + checksum	CODABAR+			
Code 2 of 5, interleaved	CODE25	N	2 to 30 [2]	2, 2.5, 3
Code 2 of 5, interleaved + checksum	CODE25+			
Code 3 of 9	CODE39	N, U, P	2 to 30	2, 2.5, 3
Code 3 of 9 + checksum	CODE39+			
Code 3 of 9 extended	EXTEND39	N, U, L, P, C	2 to 30	2, 2.5, 3
Code 3 of 9 extended + checksum	CODE39X+			
Code 128 mode A + checksum	CODE128A+	N, U, P, C [3]	2 to 30	3
Code 128 mode B + checksum	CODE128B+	N, U, L, P [4]	2 to 30	3
Code 128 mode C + checksum	CODE128C+	N [5]	2 to 30	3
Code 128 Automatic[6] + checksum	CODE128+	N, U, L, P, C	2 to 30	3
Compressed code 3 of 9	CODE93	N, U, P [7]	2 to 30	2, 2.5, 3
Compressed code 3 of 9 + checksum	CODE93+			
Compressed code 3 of 9 extended	EXTEND93	N, U, L, P, C	2 to 30	2, 2.5, 3
Compressed code 3 of 9 extended + checksum	EXTEND93+			
EAN/JAN-8	EAN8	N	7, 9, or 12	3
EAN/JAN-13	EAN13	N	12, 14, or 17	3
MSI Plessey	MSIPLESSEY	N	2 to 15	2, 2.5, 3
MSI Plessey + checksum	MSIPLESSEY+			
POSTNET (ZIP+4) postal code[8]	POSTNET	N	5, 9, or 11	3
UCC-128 (Code128C)	UCC128	N	19	3
UCC-128 (Code128C) + checksum	UCC128+			
UPC Type A	UPCA	N	11, 13, or 16	3
UPC Type E	UPCE	N	11, 13, or 16	3

Special Instructions

[1]**Codabar** Valid characters are numerals and - $: / +
Start and end characters: A B C D

[2]**Code 2 of 5** The number of data characters must be even.

[3]**Code 128 mode A** Valid characters are the standard keyboard characters, and the following special characters:

NUL	BS	DLE	CAN	SHIFT	FNC1
SOH	HT	DC1	EM	BEL	FNC2
STX	LF	DC2	SI	SUB	FNC3
ETX	VT	DC3	US	ETB	FNC4
EOT	FF	DC4	FS	ESC	
ENQ	CR	NAK	GS	CODEB	
ACK	SO	SYN	RS	CODEC	

[4]**Code 128 mode B** Valid characters are the standard keyboard characters, and the following special characters:

CODEA	FNC2
CODEC	FNC3
FNC1	FNC4

[5]**Code 128 mode C** Valid characters are the set of 100 digit pairs from 00 to 99, and the following special characters:

CODEA
CODEB
FNC1

[6]**Code 128 automatic** Automatically selects mode A, B, or C for optimal barcode size.

[7]**Compressed Code 3 of 9** Valid characters are numerals, capital letters, the space character, and - . $ / + %

[8]**Postnet** This barcode does not have human readable text.

Using a GDI Print Device

To use a Windows NT print driver other than PCL, PostScript, or RTF, use a Graphics Device Interface, or GDI, device module. OptioDCS for Windows NT includes a program that creates GDI device modules.

Note: This feature is for OptioDCS for Windows NT only.

Note: Certain GDI device modules may produce erroneous results, such as excess copies of each page, due to defective Windows NT printer drivers. If this happens, create a new GDI device module, substituting a reliable driver that is comparable to the defective driver.

Creating a GDI Device

To create a GDI device module, use the Make GDI Device Files program provided with OptioDCS. The Windows NT printer that the device module uses must be added on the NT Server that is used to run OptioDCS. For more information on creating a GDI device module, see Chapter 7 of the *Advanced Features Guide*.

A sample GDI device module is created when you install OptioDCS. This device is in the file ~dev\defgdi.dev, and is based on the default Windows NT printer on your server.

Document Guidelines

When using a GDI device module with your document, use the following guidelines:

- Select the GDI device in your document or part module.
- The default output tray is always used. Therefore, you cannot select an output tray (or stacker).
- The device supports the paper sizes and trays as specified in the Windows NT settings for the printer. If you specify a conflicting paper size and paper tray (or feeder), the paper size overrides the tray selection. Therefore, it is best to specify only a paper tray.
- The device supports the Windows NT fonts that are available through the printer driver. You can add font aliases for these fonts using the TYPEFACE clause with the Windows NT typeface name and style. For more information, see Chapter 7 of the *Advanced Features Guide*.

Chapter 3

Label Printer Drivers

OptioDCS provides drivers for the following label printers:

- Intermec (ICL)
- Monarch (MCL)
- SATO (SCL)
- Zebra (ZPL)

To use a Windows NT print driver for any label printer, use a Graphics Device Interface, or GDI, device module. OptioDCS for Windows NT includes a program that creates GDI device modules. This enables you to use the Windows NT print driver for a label printer that is not otherwise supported by OptioDCS.

This chapter describes the unique features of each of these drivers.

For information on the other drivers included with OptioDCS, see Chapter 2 in this book.

Using the Intermec Printer Driver

The OptioDCS Intermec printer driver uses the provided Intermec device module, which contains configuration information for the Intermec printer. To print your OptioDCS documents as labels on an Intermec printer, you simply select the Intermec device and the desired label size. Before printing, you should check the printer resolution as described in the following section. You may also take advantage of the unique features of your Intermec printer, by adding new fonts or label sizes to the device module, and using the Intermec extensions to the DCL language in your documents.

Setting up the Intermec Device Module

The Intermec device module contains commands similar to the following:

```
LANGUAGE "ICL"
UNITS "DPI300"

PAPER "L4x1", "", 450, 1200,  0,0
PAPER "L4x3", "", 900, 1200,  0,0
PAPER "L4x6", "", 1800, 1200,  0,0

RESOLUTION 203, 203
```

The PAPER commands define names for various label sizes, such as "L4x3". The sizes are measured in the units specified by the UNITS command. If you want to use a label size other than the sizes already specified, you may add a new PAPER command, defining a name for the label size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width,  0,0
```

where *name* is the name for the new label size, *height* is the height in units, and *width* is the width in units.

Because Intermec provides printers in several DPI configurations, the printer resolution must also be specified in the device module using the RESOLUTION command. Ensure that the resolution specified in your device module matches your printer resolution.

The Intermec device module provides access to the built-in bitmapped fonts available in most Intermec printers. To add a new font, add the command

```
FONT "fontname" TYPEFACE "typenum" SYMBOLS "0" POINTS 1,1
```

where *fontname* is the name of the font, and *typenum* indicates the typeface.

For more information on device modules, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT). For more information on the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Note: Make a copy of the device module file, in order to preserve your changes to the device module when installing an upgrade or reinstalling OptioDCS.

Setting up your Intermec Label

To print a document as a label on an Intermec printer, you simply select the Intermec device and the desired label size. The sections below explain how to make these selections in the document and format modules. Otherwise, you create the label just as any other OptioDCS document, using DCL commands. For more information on document and format modules, and the commands shown below, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

Document Module

To select the Intermec device, add the command

```
SET DEVICE "~dev/intermec"
```

to your document module.

Format Module

To select a label size, use the PAPER clause at the beginning of each format module as shown below.

```
FORM "FileLabel"
   PAPER "L4x5"
```

This example selects the size named "L4x5". The label size must be defined in the device module, as described on page 2. The PAPER clause is a header clause, so it must be placed at the beginning of the format module.

There are limitations for certain commands and settings when producing Intermec output, as described in the following sections.

DRAW ARC

The DRAW ARC command is not supported.

DRAW BARCODE

The ALIGN clause is not supported.

The BORDER clause is not supported.

The OPAQUE clause is not supported.

DRAW BOX

The LINESTYLE clause is not supported.

The OPAQUE clause is not supported.

The PATTERN clause is not supported.

The SHADOW clause is not supported.

DRAW CURVE

The DRAW CURVE command is not supported.

DRAW ELLIPSE

The DRAW ELLIPSE command is not supported.

DRAW IMAGE

Only .TIF and .PCX image formats are supported.

Only one or two images may be printed on each label (in each format), and the total image size must be 3 KB or less.

The ALIGN clause is not supported.

The BORDER clause is not supported.

The OPAQUE clause is not supported.

The REVERSE clause is not supported.

The SHADOW clause is not supported.

DRAW LINE

Only horizontal and vertical lines are supported.

The LINECAP clause is not supported.

The LINESTYLE clause is not supported.

DRAW TEXT

The BORDER clause is not supported.

The PATTERN clause is not supported.

The REVERSE clause is not supported.

The SHADOW clause is not supported.

The SPACING clause is not supported.

The UNDERLINE clause is not supported.

The font typeface 27 prints as a Postnet barcode.

Using DCL Language Extensions

By using the Intermec extensions to the DCL language in your documents, you may take advantage of the unique features of your Intermec printer. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of printing labels, such as text magnification and barcode prefixes. Each extension command is described in the sections below.

General Extension

FORMNUMBER

A clause for the FORMAT command, used to assign a printer storage slot number to the format. This allows the printer to print the label repeatedly without completely re-imaging it. The default is 19.
Valid values are 1 through 19.

```
EXTENSION "FORMNUMBER=3"
```

Text Extensions

HORIZONTAL

A clause for the DRAW TEXT command, used to change the horizontal dimensions of a font temporarily, using a magnification factor. For example, the command below prints "hello" using the font ST4, but triples the width.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "HORIZONTAL=3"
   USING "hello"
```

The default magnification factor is 2.

VERTICAL

A clause for the DRAW TEXT command, used to change the vertical dimensions of a font temporarily, using a magnification factor. For example, the command below prints "hello" using the font ST4, but triples the height.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "VERTICAL=3"
   USING "hello"
```

The default magnification factor is 2.

BORDER

A clause for the DRAW TEXT command, used to print white text in a black bounding box. Any value other than 0 prints white text. The value determines the size of the box; a higher value prints a larger box. For example, the command below prints "hello" in white with a black bounding box.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "BORDER=2"
   USING "hello"
```

Codabar Barcode Extension

START

A clause for the DRAW BARCODE command, used to select the start character. The default value is "A".
Valid values are A, B, C, and D.
The example below selects the start code "C".

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE "CODABAR"
   EXTENSION "START=C"
```

STOP

A clause for the DRAW BARCODE command, used to select the stop character. The default value is "B".
Valid values are A, B, C, and D.

This is usually used with the START extension as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODABAR”
   EXTENSION "START=C" EXTENSION "STOP=D"
```

Code 11 Barcode Extension

PRINTERCD

A clause for the DRAW BARCODE command, used to select a one-digit check digit. For Code 11 barcodes, the default check digit length is 2. To select the one-digit check digit, use the clause

```
EXTENSION "PRINTERCD=1"
```

Code 2 of 5 Barcode Extension

STARTSTOP

A clause for the DRAW BARCODE command, used to select the 2-bar start/stop code. The default start/stop code is the 3-bar code. To select the 2-bar code, use the clause

```
EXTENSION "STARTSTOP=2"
```

Code 3 of 9 Barcode Extensions

PREFIX

A clause for the DRAW BARCODE command, used to specify a prefix for the barcode. The prefix may be up to 4 alphanumeric characters, using the uppercase characters A through Z and the digits 0 through 9. The prefix is not printed in the human-readable text. The example below indicates a prefix of “A1B2”.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODE39+”
   EXTENSION "PREFIX=A1B2"
```

BRAND

A clause for the DRAW BARCODE command, used to indicate which brand of Code 3 of 9 barcode to print. To print a Code 3 of 9 barcode using the 43-character set, use the clause

```
EXTENSION "BRAND=43CHAR"
```

To print a Code 3 of 9 barcode using the full ASCII character set, use the clause

```
EXTENSION "BRAND=FULLASCII"
```

To print a Code 3 of 9 barcode using the 8646 compatible character set, use the clause

```
EXTENSION "BRAND=8646"
```

This is the default value. The 8646 compatible brand is the same as full ASCII, except that it encodes the dollar sign ($), percent sign (%), slash (/), and plus sign (+) as single characters.

EAN/JAN and UPC Barcode Extensions

FLAG

A clause for the DRAW BARCODE command, used to disable flag 1, which determines which human-readable characters are printed inside and outside the guard patterns. To disable flag 1, use the clause

```
EXTENSION "FLAG=NO"
```

To enable flag 1, use the clause

```
EXTENSION "FLAG=YES"
```

This is the default value.

Valid for barcode types: EAN/JAN-8, EAN/JAN-13, UPC Type A

SUPPLEMENT

A clause for the DRAW BARCODE command, used to print a supplemental barcode after the barcode. For example, the command below prints a barcode and a supplemental barcode with a value of 32.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “EAN13”
   EXTENSION "SUPPLEMENT=32"
```

The value may be two or five characters.

Valid for barcode types: EAN/JAN-8, EAN/JAN-13, UPC Type A, UPC Type E, UPC Version D1, UPC Version D2, UPC Version D3, UPC Version D4, UPC Version D5

HIBC Barcode Extension

MODIFIER

A clause for the DRAW BARCODE command, used to select the type of modifier to add to the barcode data. All modifiers (including the default) add a check digit to the end of the data, which is also printed in the human-readable text.

For a Supplier Standard barcode, you may add one or two plus signs to the beginning of the data. To add a single plus sign to the data, use the clause

```
EXTENSION "MODIFIER=PRIMARY"
```

This is the default modifier. To add two plus signs to the data, use the clause

```
EXTENSION "MODIFIER=ALTERNATE"
```

For a Provider Standard barcode, you may add a check digit, or no modifier, to the beginning of the data. To add a check digit to the beginning of the data, use the clause

```
EXTENSION "MODIFIER=FIRSTDATA"
```

To add no modifier to the beginning of the data, use the clause

```
EXTENSION "MODIFIER=SINGLE"
```

or the clause

```
EXTENSION "MODIFIER=MULTIPLEDATA"
```

Postnet Barcode Extensions

VMAGNIFY

A clause for the DRAW BARCODE command, used to change the height of the barcode, using a magnification factor from 1 to 250. This factor is multiplied by the base character height of 13 dots. For example, the command below prints a Postnet barcode with a character height of 39 dots.

```
DRAW BARCODE AT 1,1.8 USING zipcode TYPE “POSTNET”
   EXTENSION "VMAGNIFY=3"
```

The default magnification factor is 2.

HMAGNIFY

A clause for the DRAW BARCODE command, used to change the width of the Postnet characters, using a magnification factor from 1 to 250. This factor is multiplied by the base character width of 22 dots. For example, the command below prints a Postnet barcode with a character width of 66 dots.

```
DRAW BARCODE AT 1,1.8 USING zipcode TYPE “POSTNET”
   EXTENSION "HMAGNIFY=3"
```

The default magnification factor is 2.

UCC-128 Barcode Extension

PAREN

A clause for the DRAW BARCODE command, used to remove parentheses, brackets, and spaces from the barcode data. To specify that they should be removed, use the clause

```
EXTENSION "PAREN=IGNORE"
```

To specify that they should not be removed, use the clause

```
EXTENSION "PAREN=KEEP"
```

Spaces, parentheses, and brackets are not encoded in the barcode, but are retained in the human-readable text. This is the default behavior.

Barcode Extensions for Human-Readable Text

These extensions affect the appearance of the human-readable text that is printed with a barcode.

Note: These extensions do not work for EAN/JAN and UPC barcodes.

TYPEFACE

A clause for the DRAW BARCODE command, used to select a typeface for the human-readable text. The typeface values are defined in the device module. For example, the clause below selects the "20" typeface.

```
EXTENSION "TYPEFACE=20"
```

The default typeface is "0".

HORIZONTAL

A clause for the DRAW BARCODE command, used to change the horizontal dimensions of a font temporarily, using a magnification factor from 1 to 250. For example, the clause below triples the width of the human-readable font.

```
EXTENSION "HORIZONTAL=3"
```

The default magnification factor is 2.

VERTICAL

A clause for the DRAW BARCODE command, used to change the vertical dimensions of a font temporarily, using a magnification factor from 1 to 250. For example, the clause below triples the height of the human-readable font.

```
EXTENSION "VERTICAL=3"
```

The default magnification factor is 2.

POINT

A clause for the DRAW BARCODE command, used to select a point size for the human-readable text, from 4 to 212. The point size is the font height in points (1/72 inch), measured from the top of the tallest characters to the bottom of the lowest descenders. For example, the clause below sets the size to 14 points.

```
EXTENSION "POINT=14"
```

The default point size is 12.
Note: Do not use this extension with the HORIZONTAL and VERTICAL extensions.

PITCH

A clause for the DRAW BARCODE command, used to select a pitch size for the human-readable text, from 1 to 30. The pitch is the number of characters per inch (horizontally). This extension overrides the POINT, HORIZONTAL, and VERTICAL extensions. For example, the clause below sets the pitch to 10 characters per inch.

```
EXTENSION "PITCH=10"
```

The default pitch is 12.

OUTLINE

A clause for the DRAW BARCODE command, used to print white text in a black bounding box. Any value other than 0 prints white text. For example, the clause below causes the the human-readable text to be printed in white with a black bounding box.

```
EXTENSION "OUTLINE=2"
```

ROTATION

A clause for the DRAW BARCODE command, used to rotate the human-readable text around its point of origin. You may rotate the text counter-clockwise in 90 degree increments. To rotate the text 180 degrees, use the clause

```
EXTENSION "ROTATION=180"
```

To rotate the text 270 degrees, use the clause

```
EXTENSION "ROTATION=270"
```

To rotate the text 90 degrees, use the clause

```
EXTENSION "ROTATION=90"
```

and use the AT extension described below, to move the point of origin so the text does not cross the barcode.

SPIN

A clause for the DRAW BARCODE command, used to rotate the individual characters of the human-readable text (as opposed to the whole line of text) 90 degrees counter-clockwise. To rotate the characters 90 degrees, use the clause

```
EXTENSION "SPIN=90"
```

Depending on the size of the font, it may be necessary to use the AT extension described below, to move the point of origin so the text does not overlap the barcode.

AT

A clause for the DRAW BARCODE command, used to select the point of origin position for the human-readable text, in printer dots. For example, the clause below selects the point 100 points down and 150 points across as the point of origin.

```
EXTENSION "AT=100,150"
```

The default position is immediately below the barcode.

Barcode Reference

This section lists the supported barcode types and styles. See the barcode extension sections (starting on page 6) for the language extensions that apply to each barcode type. For more information on a particular barcode, see your printer documentation.

Types and Keywords

The following table lists the barcode types that are supported by OptioDCS for printing using the Intermec printer driver. The types are listed in alphabetical order, along with the valid DCL keyword used to specify the type.

Type	Keyword
Codabar	CODABAR
Code 2 of 5	CODE25
Code 2 of 5, interleaved	INT25
Code 2 of 5, interleaved + check digit	INT25+
Code 3 of 9	CODE39
Code 3 of 9 + check digit	CODE39+
Code 11 + check digit	CODE11+
Code 16K	CODE16K
Code 49 two-dimensional barcode	CODE49
Code 128 Automatic	CODE128
Compressed code 3 of 9	CODE93
EAN/JAN-8	EAN8
EAN/JAN-13	EAN13
HIBC Code 39	HIBC39
HIBC Code 128	HIBC128
POSTNET (ZIP+4) postal code	POSTNET
UCC-128 (Code128C)	UCC128
UPC Type A	UPCA
UPC Type E	UPCE
UPC Version D1	UPCD1
UPC Version D2	UPCD2
UPC Version D3	UPCD3
UPC Version D4	UPCD4
UPC Version D5	UPCD5

Styles

A barcode style is set using the STYLE clause with the SET or DRAW BARCODE command. The style determines whether human-readable text is printed with the barcode. The human-readable text is printed 2 dots below the bar code field, left justified, using the ST0 (7x9 standard) font. You may use DCL extensions to change the text attributes (see page 10). The keywords used to set the style are listed in the table below.

Keyword	**Style**
NOTEXT, TEXT0	without text, interpretive field disabled
TEXT1	with text, with start and stop characters, interpretive field enabled
TEXT2	with text, without start and stop characters, interpretive field enabled

The default setting is without text, and the interpretive field disabled. The style does not apply to the POSTNET barcode because this barcode type does not include human-readable text.

Using the Monarch Printer Driver

The OptioDCS Monarch printer driver uses the provided Monarch device module, which contains configuration information for the Monarch printer. To print your OptioDCS documents as labels on a Monarch printer, you simply select the Monarch device and the desired label size. Before printing, you should check the printer resolution as described in the following section. You may also take advantage of the unique features of your Monarch printer, by adding new font or label sizes to the device module, and using the Monarch extensions to the DCL language in your documents.

Setting up the Monarch Device Module

The Monarch device module contains commands similar to the following:

```
LANGUAGE "MCL"
UNITS "INCHES"

PAPER "L4x3", "", 3, 4,  0,0

RESOLUTION 192, 192
```

The PAPER commands define names for various label sizes, such as "L4x3" for a label that is 4 inches wide and 3 inches tall. The sizes are measured in the units specified by the UNITS command. If you want to use a label size other than the sizes already specified, you may add a new PAPER command, defining a name for the label size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width,  0,0
```

where *name* is the name for the new label size, *height* is the height in units, and *width* is the width in units.

Because Monarch provides printers in several resolutions, the printer resolution must also be specified in the device module using the RESOLUTION command. Ensure that the resolution specified in your device module matches your printer resolution.

The Monarch device module provides access to the built-in bitmapped fonts available in most Monarch printers, as well as a selection of sizes for the built-in scalable font. To add a new font, or a new font size for an existing font, add a new font command to

specify the characteristics of the new font or font size. For example, to add a new size for the scalable font, add the command

```
FONT "fontname" TYPEFACE "0" SYMBOLS "0" POINTS height,width
```

where *fontname* is the name of the font, and *height* and *width* indicate the character size magnification factors.

For more information on device modules, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT). For more information on the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Note: Make a copy of the device module file, in order to preserve your changes to the device module when installing an upgrade or reinstalling OptioDCS.

Setting up your Monarch Label

To print a document as a label on a Monarch printer, select the Monarch device and the desired label size. The sections below explain how to make these selections in the document and format modules. Otherwise, you create the label just as any other OptioDCS document, using DCL commands and the DCL language extensions described in the next section.

There are several sample labels included with the Monarch driver that may be used as examples; these are located in the ~/doc directory. For more information on document and format modules, and the commands shown below, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

Document Module

To select the Monarch device, add the command

```
SET DEVICE "~dev/mcl"
```

to your document module.

Format Module

To select a label size, use the PAPER clause at the beginning of each format module as shown below.

```
FORM "FileLabel"
   PAPER "L4x5"
```

This example selects the size named "L4x5". The label size must be defined in the device module, as described on page 2. The PAPER clause is a header clause, so it must be placed at the beginning of the format module.

There are limitations for certain commands and settings when producing Monarch output, as described below and in the following sections.

- Some Monarch printers do not allow more than 100 elements per label.
- When two elements overlap, one element may not be printed.
- The landscape format layout is not supported. However, individual elements may be rotated 90 or 270 degrees using the ROTATION clause.

DRAW ARC

The DRAW ARC command is not supported.

DRAW BARCODE

When using a variable for the barcode data, use the MAXSIZE extension described on page 21.

When using a check digit, use the MAXSIZE and OPTION extensions described on page 21.

The AT position specifies the lower left corner of the barcode, instead of the upper left corner.

The DENSITY clause is not supported for the SET and DRAW BARCODE commands. Use the DENSITY extension described on page 21.

The STYLE clause is not supported for the SET and DRAW BARCODE commands. Use the STYLE extension described on page 22.

DRAW BOX

Fill patterns are not supported. To draw a solid black box, use DRAW LINE to draw a very thick line.

When using the TO clause to specify a corner of a box, the AT clause must specify the upper left corner, and the TO clause must specify the lower right corner.

DRAW COMB

The DRAW COMB command is not supported.

DRAW ELLIPSE

The DRAW ELLIPSE command is not supported.

DRAW IMAGE

Large or multiple images may cause erros, because of printer memory limits.

Do not place other elements very close to an image. To print correctly, each image must have a small amount of white space around it.

DRAW LINE

Diagonal lines are not supported.

The LINECAP clause is not supported.

The LINESTYLE clause is not supported.

DRAW OBJECT

The DRAW OBJECT command is not supported.

DRAW TEXT

When using a text variable, use the MAXSIZE extension described on page 20.

The BORDER clause is not supported.

Using DCL Language Extensions

By using the Monarch extensions to the DCL language in your documents, you may take advantage of the unique features of your Monarch printer. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of printing labels, such as barcode densities and check digit algorithms. Each extension clause is described in the sections below.

Note: In order to use these extensions and other features of your printer, you will need information from the Monarch Programmer's Manual for your printer model. This manual may be ordered from Monarch.

General Extensions

CHECK_DIGIT_1 through CHECK_DIGIT_10

Clauses for the FORMAT command, used to define check digit algorithms or packets for barcodes. (These packets are not used for UPC and EAN barcodes, which have predefined check digit algorithms.) You may define up to 10 check digit packets for a format; the packet numbers are used to select the algorithm for each barcode. For example, the clause below defines check digit packet number 3.

```
EXTENSION "CHECK_DIGIT_3=10,5,P,\"65432\""
```

To define a check digit packet, you must indicate the modulus, field length, algorithm type, and weights. These parameters are described in the Monarch manual section on applying check digits.

Modified Printer Symbol Extensions

If your printer has been modified so that it does not use the standard MPCL punctuation, you may change these standards in your OptioDCS documents using the extensions described below.

FORM_START

A clause for the FORMAT command, used to set the delimiter that indicates the start of a packet. For example, the clause below sets the packet starting delimiter to **{**, which is the default.

```
EXTENSION "FORM_START={"
```

FORM_STOP

A clause for the FORMAT command, used to set the delimiter that indicates the end of a packet. For example, the clause below sets the packet ending delimiter to **}**, which is the default.

```
EXTENSION "FORM_STOP=}"
```

COMMAND_SEPARATOR

A clause for the FORMAT command, used to set the delimiter that indicates the end of a field. For example, the clause below sets the field delimiter to |, which is the default.

```
EXTENSION "COMMAND_SEPARATOR=|"
```

DATA_START

A clause for the FORMAT command, used to set the delimiter that indicates the start of a character string. For example, the clause below sets the string starting delimiter to **\"**, which is the default.

```
EXTENSION "DATA_START=\""
```

DATA_STOP

A clause for the FORMAT command, used to set the delimiter that indicates the end of a character string. For example, the clause below sets the string ending delimiter to \", which is the default.

```
EXTENSION "DATA_STOP=\""
```

DATA_SEPARATOR

A clause for the FORMAT command, used to set the delimiter that separates parameters within fields. For example, the clause below sets the parameter delimiter to **,**, which is the default.

```
EXTENSION "DATA_SEPARATOR=,"
```

Text Extensions

MAXSIZE

A clause for the DRAW TEXT command, used to set the maximum number of characters in the text string. This clause is required when printing text using a variable (instead of a literal string). For example, the command below prints the value of the variable **mystring**, which has maximum length of twelve characters.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "MAXSIZE=12"
   USING mystring
```

For more information, see the Monarch manual section on defining text fields.

OPTION

A clause for the DRAW TEXT command, used to select Monarch text field options. You can use any combination of the text field options with each text string. To apply an option, use the clause

```
EXTENSION "OPTION=num,arguments"
```

where *num* is the option number and *arguments* are the option arguments. For example, the command below prints text using options 1 and 4.

```
DRAW TEXT AT 1,1.7 FONT "ST4" USING mystring
   EXTENSION "OPTION=1,Made In _________"
   EXTENSION "OPTION=4,2,1,4,3,2"
```

For more information on these options, see the Monarch manual section on applying field options.

General Barcode Extensions

DENSITY

A clause for the DRAW BARCODE command, used to set the density for a barcode. (The DENSITY clause is not supported.) The valid density codes are listed in the Monarch manual section on defining barcode fields. For example, the clause below sets the density code to 2.

```
EXTENSION "DENSITY=2"
```

MAXSIZE

A clause for the DRAW BARCODE command, used to set the maximum number of characters in the barcode data. This clause is required when printing a barcode using a variable (instead of a literal string). For example, the command below prints a barcode with the value of the variable **itemnum**, which has maximum length of ten characters.

```
DRAW BARCODE AT 1,1.7 TYPE "UPCA" USING itemnum
   EXTENSION "MAXSIZE=10"
```

This length must include any check digit on the barcode.

OPTION

A clause for the DRAW BARCODE command, used to select Monarch barcode field options. You may use any combination of the barcode field options with each barcode. To apply an option, use the clause

```
EXTENSION "OPTION=num,arguments"
```

where *num* is the option number and *arguments* are the option arguments. For example, the command below prints a barcode using option 1 and 4.

```
DRAW BARCODE AT 1,1.7 TYPE "UPCA" USING itemnum
   EXTENSION "OPTION=1,Made In _________"
   EXTENSION "OPTION=4,2,1,4,3,2"
```

For more information on these options, see the Monarch manual section on applying field options. If you are using a check digit, you must include option 31 which indicates which check digit packet to use. For example, the clause below inidicates check digit packet number 2.

```
EXTENSION "OPTION=31,G,2"
```

Check digit packets are defined using the CHECK_DIGIT_# extensions described on page 19.

Code 16K Barcode Extension

CDHEIGHT

A clause for the DRAW BARCODE command, used to set the height for a barcode. (The HEIGHT clause is not supported for Code 16K barcodes.) The height may be from 2 to 16, in units of .25 mm. For example, the clause below sets the height at 1.5 mm.

```
EXTENSION "HEIGHT=6"
```

UPC and EAN Barcode Extension

STYLE

A clause for the DRAW BARCODE command, used to select a style for the human readable text. (The standard STYLE clause is not supported for UPC or EAN barcodes.) The valid barcode appearance codes are listed in the Monarch manual section on defining barcode fields. For example, the clause below sets the appearance code to 5.

```
EXTENSION "STYLE=5"
```

Barcode Reference

This section lists the supported barcode types and styles. See the barcode extension sections (starting on page 21) for the language extensions that apply to each barcode type. For more information on a particular barcode, see your printer documentation.

Types and Keywords

The following table lists the barcode types that are supported by OptioDCS for printing using the Monarch printer driver. The types are listed in alphabetical order, along with the valid DCL keyword used to specify the type.

Type	Keyword
Codabar	CODABAR
Code 2 of 5, interleaved	INT25
Code 2 of 5 with Barrier Bar	CODE25BB
Code 3 of 9	CODE39
Code 3 of 9 + check digit	CODE39+
Code 16K	CODE16K
Code 39 mode 43	CODE39M43
Code 128 Automatic	CODE128
EAN/JAN-8	EAN8
EAN/JAN-8 + 2	EAN82
EAN/JAN-8 + 5	EAN85
EAN/JAN-13	EAN13
EAN/JAN-13 + 2	EAN132
EAN/JAN-13 + 5	EAN135
EAN/JAN-13 & Price check digit	EAN13+
MSI	MSI
PDF417 two-dimensional barcode	PDF417
POSTNET (ZIP+4) postal code	POSTNET
UPC Type A	UPCA
UPC Type A + 2	UPCA2
UPC Type A + 5	UPCA5
UPC Type A & Price check digit	UPCA+
UPC Type E	UPCE
UPC Type E + 2	UPCE2
UPC Type E + 5	UPCE5

Styles

A barcode style is set for UPC and EAN barcodes using the STYLE extension (see page 22). The style determines how human readable text is printed with the barcode. The style codes are listed in the section on defining barcode fields in the Monarch Programmer's Manual for your printer model.

Note: Do not use the STYLE clause with the SET or DRAW BARCODE command.

Using the SATO Printer Driver

The OptioDCS SATO printer driver uses the provided SATO device module, which contains configuration information for the SATO printer. To print your OptioDCS documents as labels on a SATO printer, you simply select the SATO device and the desired label size. Before printing, you should check the printer resolution as described in the following section. You may also take advantage of the unique features of your SATO printer, by adding new fonts or label sizes to the device module, and using the SATO extensions to the DCL language in your documents.

Note: In order to print OptioDCS documents on your SATO printer, the printer must be set to accept non-standard command codes. Make sure that **Set Proto-Codes** is set to **Non-Standard**. This setting is located under **Mode I Options** in the printer menus.

Setting up the SATO Device Module

The SATO device module contains commands similar to the following:

```
LANGUAGE "SCL"
UNITS "DPI300"

RESOLUTION 203, 203

PAPER "L4x7", "", 2140, 1280,  0,0

FONT "ST4" COMMAND "OA" TYPEFACE "OA"
```

Because SATO provides printers in several DPI configurations, the printer resolution must also be specified in the device module using the RESOLUTION command. Ensure that the resolution specified in your device module matches your printer resolution.

The PAPER commands define names for various label sizes, such as “L4x3”. The sizes are measured in the units specified by the UNITS command. If you want to use a label size other than the sizes already specified, you may add a new PAPER command, defining a name for the label size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width,  0,0
```

where *name* is the name for the new label size, *height* is the height in units, and *width* is the width in units.

The SATO device module provides access to the built-in fonts available in most SATO printers. To add a new font, use the command

```
FONT "fontname" COMMAND "satofont" TYPEFACE "satofont"
```

where *fontname* is a name for the font, and *satofont* is the SATO typeface. For more information on SATO typefaces or fonts, see your printer documentation.

For more information on device modules, see Chapter 3 of the *User's Guide*. For more information on the FONT command, see Chapter 10 of the *User's Guide*.

Note: Make a copy of the device module file, in order to preserve your changes to the device module when installing an upgrade or reinstalling OptioDCS.

Setting up your SATO Label

To print a document as a label on a SATO printer, you simply select the SATO device and the desired label size. The sections below explain how to make these selections in the document and format modules. Otherwise, you create the label just as any other OptioDCS document, using DCL commands.

There are several sample labels included with the SATO driver that may be used as examples; these are located in the ~/doc directory. For more information on document and format modules, and the commands shown below, see Chapter 3 of the *User's Guide*.

Document Module

To select the SATO device, add the command

```
SET DEVICE "~dev/sato"
```

to your document module.

Format Module

To select a label size, use the PAPER clause at the beginning of each format module as shown below.

```
FORM "FileLabel"
   PAPER "L4x5"
```

This example selects the size named "L4x5". The label size must be defined in the device module, as described on page 24. The PAPER clause is a header clause, so it must be placed at the beginning of the format module.

There are limitations for certain commands and settings when producing SATO output, as described in the following sections.

DRAW BARCODE
The ALIGN clause is not supported.

DRAW IMAGE
The ROTATION clause is not supported, except when printing an image from printer memory as described on page 31.

DRAW LINE
Diagnoal lines are not supported.

DRAW TEXT
The ALIGN clause is not supported.

Using DCL Language Extensions

By using the SATO extensions to the DCL language in your documents, you may take advantage of the unique features of your SATO printer. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of printing labels, such as text style and barcode incrementation. Each extension command is described in the sections below. For more information on the SATO printer features and settings mentioned in these sections, see your printer documentation.

Format Module Extensions

METHOD
A command used to indicate whether you are printing through the parallel line printer port or the serial com port. If you are printing through the com port, a wake-up command may be necessary before printing. To indicate the com port and have a wake-up call sent to the printer, use the command

```
EXTENSION "METHOD=COM"
```

To indicate the line printer, use the command

```
EXTENSION "METHOD=LPT"
```

The default value is LPT.

STORE

A command used to indicate whether to store images in printer memory or to print labels normally. If your printer contains the optional RAM card, you can use this extension to store images. To store images in optional RAM, use the command

```
EXTENSION "STORE=YES"
```

A format with this extension should contain only DRAW IMAGE commands with the SLOTNO and IMAGENO extensions described on page 31. The default value is "NO", meaning print labels normally.

Box Extension

METHOD

A clause for the DRAW BOX command, used to create a rectangular area that prints in reverse. This means that each black pixel becomes white and each white pixel becomes black. This extension causes the command to reverse print the area defined by the box, instead of drawing the box. Any other optional clauses used with the DRAW BOX command (such as LINESTYLE) are ignored. If two reverse areas overlap, the overlapping area is reversed twice, so it prints normally. To reverse print an area, use the command

```
DRAW BOX AT starty,startx TO endy,endx EXTENSION "METHOD=R"
```

where *starty* and *startx* are the vertical and horizontal measurements where the area should start, and *endy* and *endx* are the vertical and horizontal measurements where the area should end. (You may also use the HEIGHT and WIDTH clauses instead of the TO clause.)

General Text Extensions

PITCH

A clause for the DRAW TEXT command, used to change the spacing between characters, using a space width expressed in printer dots. Valid values are 00-99. The default value is 02. For non-vector fonts, this value is multiplied by the HORIZONTAL setting value (see page 30). This overrides any SPACING_MODE setting. For example, the command below prints "hello" using the font ST4, with a spacing width of 34 dots.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "PITCH=34"
   USING "hello"
```

SPACING_MODE

A clause for the DRAW TEXT command, used to change a proportional font to fixed spacing temporarily. Do not use the PITCH extension with this extension. To print text using fixed spacing, use the clause

```
EXTENSION "SPACING_MODE=PR"
```

The default value is PS for proportional spacing.

STEP

A clause for the DRAW TEXT command, used to enable incrementing text and set the increment. When printing multiple copies of the label, the numeric value in the text is incremented for each copy. The default value is "+0" (no incrementing). To set the increment, use the clause

```
EXTENSION "STEP=inc"
```

where *inc* is the increment. For example, the command below prints the value of **itemnum** on the first label, then increments it by +15 for each subsequent copy.

```
DRAW TEXT AT 1,1.8 USING itemnum FONT "ST4"
   EXTENSION "STEP=+15"
```

SET

A clause for the DRAW TEXT command, used with the STEP extension, to prevent incrementing barcodes and text. When printing multiple copies of the label, the SET value is the number of identical copies printed before incrementing. If multiple elements have SET values, the lowest SET value is used. The SET value must be lower than the COPIES value for incrementing to occur.

For example, the following document prints six labels for each page of data, and any incrementing elements are incremented for each copy (5 times), because none of the DRAW statements have a SET clause.

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 6
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   ⋮
   DRAW TEXT AT 1,2 USING itemnum
      EXTENSION "STEP=+10"
   DRAW BARCODE AT 1,2 USING itemnum TYPE “CODE39”
      EXTENSION "STEP=+5"
   ⋮
END FORM
```

The following document also prints six labels for each page of data, but the incrementing elements are only incremented once for each page. This produces two sets of labels for each page; each set is three identical labels.

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 6
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   ⋮
   DRAW TEXT AT 1,2 USING itemnum
      EXTENSION "STEP=+10" EXTENSION "SET=3"
   DRAW BARCODE AT 1,2 USING itemnum TYPE “CODE39”
      EXTENSION "STEP=+5"
   ⋮
END FORM
```

The following document prints five labels for each page of data, incrementing the text three times for four sets (the fourth set is only one label).

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 7
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   ⋮
   DRAW TEXT AT 1,2 USING itemnum
      EXTENSION "STEP=+10" EXTENSION "SET=2"
   DRAW TEXT AT 1,2 USING itemnum
      EXTENSION "STEP=+10" EXTENSION "SET=5"
   ⋮
END FORM
```

SEQUENTIAL

A clause for the DRAW TEXT command, used with the STEP extension, to limit the number of digits that are incremented. Valid values are 1-99. The default value is 8. To set a limit on incrementing digits, use the clause

```
EXTENSION "SEQUENTIAL=digits"
```

where *digits* is the limit.

For example, the following statement prints up to 10 labels with incrementing; any subsequent copies are printed with the final value: 12390.

```
DRAW TEXT AT 1,1.8 USING “12300”
   EXTENSION "STEP=+10", "SEQUENTIAL=2"
```

FREE

A clause for the DRAW TEXT command, used with the STEP extension, to prevent incrementing a certain number of the rightmost digits of the number. This sets a number of digits that are ignored when incrementing. The other digits are incremented as if the ignored digits do not exist. Valid values are 0-99. The default value is 0. The FREE value must be lower than the length of the number for incrementing to occur. To prevent incrementing rightmost digits, use the clause

```
EXTENSION "FREE=digits"
```

where *digits* is the number of digits. For example, the following statement prints values that end in “07”: 12307, 12507, 12707, etc.

```
DRAW TEXT AT 1,1.8 USING “12307”
   EXTENSION "STEP=+2”, “FREE=2”
```

Text Extensions for Non-Vector Fonts

HORIZONTAL

A clause for the DRAW TEXT command, used to change the horizontal dimensions of a font temporarily, using a magnification factor of the standard width of the font. Valid values are 01-12. For example, the command below prints “hello” using the font ST4, but uses a width that is double the standard width.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "HORIZONTAL=2"
   USING "hello"
```

VERTICAL

A clause for the DRAW TEXT command, used to change the vertical dimensions of a font temporarily, using a magnification factor of the standard height of the font. Valid values are 01-12. For example, the command below prints “hello” using the font ST4, but uses a height that is double the standard height.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "VERTICAL=2"
   USING "hello"
```

Text Extensions for Vector Fonts

VECTOR_WIDTH

A clause for the DRAW TEXT command, used to change the width of the font temporarily, using a character width expressed in printer dots. Valid values are 50-999.

The default value is 100. For example, the command below prints "hello" using the font ST8, with a character width of 300 dots.

```
DRAW TEXT AT 1,1.7 FONT "ST8" EXTENSION "VECTOR_WIDTH=300"
   USING "hello"
```

VECTOR_HEIGHT

A clause for the DRAW TEXT command, used to change the height of the font temporarily, using a character height expressed in printer dots. Valid values are 50-999. The default value is 200. For example, the command below prints "hello" using the font ST8, with a character height of 300 dots.

```
DRAW TEXT AT 1,1.7 FONT "ST8" EXTENSION "VECTOR_HEIGHT=300"
   USING "hello"
```

METHOD

A clause for the DRAW TEXT command, used to change the appearance of a font. Valid values are 0-9. See your printer documentation for the effect associated with each setting. For example, the command below prints "hello" using font ST8 and method 8.

```
DRAW TEXT AT 1,1.7 FONT "ST8" EXTENSION "METHOD=8"
   USING "hello"
```

SMOOTH

A clause for the DRAW TEXT command, used to disable auto smoothing. Some larger fonts have a jagged appearance when printed without auto smoothing. To print text without auto smoothing, use the clause

```
EXTENSION "SMOOTH=0"
```

The default value is 1 which enables auto smoothing.

Image Extensions

SLOTNO and IMAGENO

Clauses for the DRAW IMAGE command, used to store images in printer memory or to print an image from printer memory. Your printer must contain the optional RAM card to use these clauses.

- To store images, include the command

  ```
  EXTENSION "STORE=YES"
  ```

  in the format module, and use a DRAW IMAGE command for each image. (Do not include other DRAW statements in the format module.) Use the SLOTNO and IMAGENO extensions to assign slot and image numbers to the image in the printer memory. Do not use the AT clause.
  Valid values for the slot number are CC1 and CC2.

Valid values for the image number are 000-999.
To assign numbers to an image in optional RAM, use the clause

```
EXTENSION "SLOTNO=CCn","IMAGENO=imagenum"
```

where *Ccn* is a slot number and *imagenum* is a number for the image.

- To print stored images, do not use the STORE command in the format module. Use a DRAW IMAGE command for each image, using the SLOTNO and IMAGENO extensions to select the image by slot and image numbers. Do not use the USING clause. Valid values for the slot number are CC1 and CC2. To select an image from optional RAM for printing use the clause

  ```
  EXTENSION "SLOTNO=CCn","IMAGENO=imagenum"
  ```

  where *Ccn* is the slot number and *imagenum* is the number assigned to the image.

General Barcode Extensions

STEP

A clause for the DRAW BARCODE command, used to enable an incrementing barcode and set the increment. When printing multiple copies of the label, the numeric value in the barcode data is incremented for each copy. The default value is "+0" (no incrementing). To set the increment, use the clause

```
EXTENSION "STEP=inc"
```

where *inc* is the increment. For example, the command below prints the value of **itemnum** on the first label, then increments it by +15 for each subsequent copy.

```
DRAW BARCODE AT 1,2 USING itemnum TYPE "CODE39"
   EXTENSION "STEP=+15"
```

For Data Matrix barcodes, you must also use the SEQSTART and SEQLENGTH extensions.

SET

A clause for the DRAW BARCODE command, used with the STEP extension, to prevent incrementing barcodes and text. When printing multiple copies of the label, the SET value is the number of identical copies printed before incrementing. If multiple elements have SET values, the lowest SET value is used. The SET value must be lower than the COPIES value for incrementing to occur. For more set printing examples, see page 28.

SEQUENTIAL

A clause for the DRAW BARCODE command, used with the STEP extension, to limit the number of digits in the data that are incremented. Valid values are 1-99. The default value is 8. To set a limit on incrementing digits, use the clause

```
EXTENSION "SEQUENTIAL=digits"
```

where *digits* is the limit. For example, the following statement prints up to 10 labels with incrementing; any subsequent copies are printed with the final value: 12390.

```
DRAW BARCODE AT 1,8 TYPE “CODE39” USING “12300”
   EXTENSION "STEP=+10", "SEQUENTIAL=2"
```

This extension does not apply to Data Matrix barcodes.

FREE

A clause for the DRAW BARCODE command, used with the STEP extension, to prevent incrementing a certain number of the rightmost digits of the number. This sets a number of digits in the data that are ignored when incrementing. The other digits are incremented as if the ignored digits do not exist. Valid values are 0-99. The default value is 0. The FREE value must be lower than the length of the number for incrementing to occur. To prevent incrementing rightmost digits, use the clause

```
EXTENSION "FREE=digits"
```

where *digits* is the number of digits. For example, the following statement prints using values that end in “07”: 12307, 12507, 12707, etc.

```
DRAW BARCODE AT 1,8 TYPE “CODE39” USING “12307”
   EXTENSION "STEP=+2”, “FREE=2”
```

This extension does not apply to Data Matrix barcodes.

Codabar Barcode Extensions

START

A clause for the DRAW BARCODE command, used to select the start character. Valid values are A, B, C, and D. The default value is "A". The following example selects the start code “C”.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODABAR”
   EXTENSION "START=C"
```

STOP

A clause for the DRAW BARCODE command, used to select the stop character. Valid values are A, B, C, and D. The default value is "A". This is usually used with the START clause as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODABAR”
   EXTENSION "START=C","STOP=B"
```

PDF417 Barcode Extensions

SECURITY

A clause for the DRAW BARCODE command, used to set the security. Valid values are 1-8. The default value is 4. The following example sets the security to 7.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “PDF417”
   EXTENSION "SECURITY=7"
```

MOD_PITCH

A clause for the DRAW BARCODE command, used to set the module pitch in dots. Valid values are 4-24. The default value is 4. This is usually used with the MOD_WIDTH clause as shown below.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “PDF417”
   EXTENSION "MOD_PITCH=6","MOD_WIDTH=4"
```

MOD_WIDTH

A clause for the DRAW BARCODE command, used to set the module width in dots. Valid values are 3-9. The default value is 3. This is usually used with the MOD_PITCH clause as shown below.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “PDF417”
   EXTENSION "MOD_PITCH=6","MOD_WIDTH=4"
```

Data Matrix Barcode Extensions

FORMAT_ID

A clause for the DRAW BARCODE command, used to select the format. Valid values are 1-6 and 11-16. The default value is 5. The following example sets the format to 4.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "FORMAT_ID=4"
```

CORRECTION

A clause for the DRAW BARCODE command, used to set the correction. Valid values are 0-1 and 4-14. The default value is 5. The following example sets the correction to 4.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "CORRECTION=4"
```

HSIZE

A clause for the DRAW BARCODE command, used to set the cell width in dots. Valid values are 3-12. The default value is 10. The following example sets the width to 4.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "HSIZE=4"
```

VSIZE

A clause for the DRAW BARCODE command, used to set the cell height in dots. Valid values are 3-12. The default value is 10. The following example sets the height to 4.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "HSIZE=4"
```

THICKNESS

A clause for the DRAW BARCODE command, used to set the guide cell thickness in dots. Valid values are 1-15. The default value is 1. The following example sets the thickness to 4.

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "THICKNESS=4"
```

PRINT

A clause for the DRAW BARCODE command, used to print the barcode in reverse. This means that each black pixel becomes white and each white pixel becomes black. To reverse print, use the clause

```
DRAW BARCODE AT 1,8 USING itemnum TYPE “DATAMATRIX”
   EXTENSION "PRINT=1"
```

SEQSTART

A clause for the DRAW BARCODE command, used with the STEP extension, to prevent incrementing a certain number of the leftmost digits of the number. This sets the leftmost digit that is incremented. Digits to the left of this one are ignored when incrementing. Valid values are 1-999. The default value is 1 (so that all digits are included). To indicate the leftmost digit to increment, use the clause

```
EXTENSION "SEQSTART=start"
```

where *start* is the position of the digit. This is used with the SEQSTART extension. For example, the following statement prints up to 10 labels with incrementing; any subsequent copies are printed with the final value: 12390.

```
DRAW TEXT AT 1,1.8 USING “12300”
   EXTENSION "STEP=+10","SEQSTART=4",“SEQLENGTH=999”
```

SEQLENGTH

A clause for the DRAW TEXT command, used with the STEP extension, to limit the number of digits that are incremented. Digits are counted from the left, starting with the digit indicated by the SEQSTART setting. Valid values are 1-999. The default value is 999 (so that all digits are included). To set a limit on incrementing digits, use the clause

```
EXTENSION "SEQLENGTH=digits"
```

where *digits* is the limit. This is used with the SEQLENGTH extension. For example, the following statement prints values that end in "07": 12307, 12507, 12707, etc.

```
DRAW TEXT AT 1,1.8 USING "12307"
   EXTENSION "STEP=+2","SEQSTART=1","SEQLENGTH=3"
```

Maxicode Extensions

For more information on Maxicode settings and requirements, see your printer documentation and the Maxicode specifications from UPS.

POSTAL_CODE

A clause for the DRAW BARCODE command, used to set the destination postal code. This clause is required for printing a Maxicode. The code must be 5 or 9 digits for delivery within the USA, and 6 digits for an international destination. The following example sets the postal code to the USA zip+4 code 30005-1234.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=300051234"
```

SERVICE_CLASS

A clause for the DRAW BARCODE command, used to set the class of service. Valid values are 001-999. The default value is 001. The following example sets the class of service to 002.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=300051234","SERVICE_CLASS=002"
```

COUNTRY_CODE

A clause for the DRAW BARCODE command, used to set the destination country by a country code. Valid values are 001-999. The default value is 840 (USA). The following example sets the country to 276.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE "MAXICODE"
   EXTENSION "POSTAL_CODE=351234","MODE=3",
   "COUNTRY_CODE=276"
```

MODE

A clause for the DRAW BARCODE command, used to set the shipping mode. Valid values are: 2 for a destination within the USA and 3 for an international destination. The

default value is 2 (domestic shipping). The following example sets the mode to 3, indicating a destination outside the USA.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=351234”,"MODE=3",
   "COUNTRY_CODE=276"
```

SYMBOLS

A clause for the DRAW BARCODE command, used to specify the number of Maxicodes in a set. Valid values are 1-8. The default value is 1. The following example specifies a set of 4 Maxicodes.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SYMBOLS=4","SYMBOL=3"
```

SYMBOL

A clause for the DRAW BARCODE command, used to number the Maxicodes within a set. Valid values are 1-8. The default value is 1. The following example specifies the third Maxicode in a set of 4.

```
DRAW BARCODE AT 1,8 USING maxistring TYPE “MAXICODE”
   EXTENSION “POSTAL_CODE=300051234”,"SYMBOLS=4","SYMBOL=3"
```

Barcode Reference

This section lists the supported barcode types and keywords. See the barcode extension sections (starting on page 32) for the language extensions that apply to each barcode type. For more information on a particular barcode, see your printer documentation.

The following table lists the barcode types that are supported by OptioDCS for printing using the SATO printer driver. The types are listed in alphabetical order, along with the valid DCL keyword used to specify the type.

Type	Keyword
Codabar	CODABAR
Code 2 of 5, interleaved	INT25
Code 2 of 5, industrial	IND25
Code 2 of 5, matrix	MTX25
Code 3 of 9	CODE39
Code 3 of 9, compressed	CODE93
Code 128 mode A	CODE128A
Code 128 mode B	CODE128B
Code 128 mode C	CODE128C
Data Matrix	DATAMATRIX
EAN/JAN-8	EAN8
EAN/JAN-13	EAN13
Maxicode	MAXICODE
MSI	MSI
PDF417	PDF417
Postnet	POSTNET
UCC 128	UCC128
UPC Type A	UPCA
UPC Type E	UPCE
UPC Supplement/Bookland	SUPPLEMENT

Styles

A barcode style is set using the STYLE clause with the SET or DRAW BARCODE command. The style determines whether human readable text is printed with the barcode. The keywords to set the style are listed in the following table. The default setting is without text. These styles apply only to the UCC128 barcode type.

Keyword	Style
TEXT0	without text
TEXT1	text below the barcode
TEXT2	text above the barcode

Using the Zebra Printer Driver

The OptioDCS Zebra printer driver uses the provided Zebra device module, which contains configuration information for the Zebra printer. To print your OptioDCS documents as labels on a Zebra printer, you simply select the Zebra device and the desired label size. Before printing, you should check the printer resolution as described in the following section. You may also take advantage of the unique features of your Zebra printer, by adding new fonts or label sizes to the device module, and using the Zebra extensions to the DCL language in your documents.

Note: If you print labels using the printer ribbon, the line thickness may not be consistent (horizontal lines are thicker than vertical lines), causing barcodes to print inaccurately. If this occurs, adjust the printer to lighten or darken the print, so that barcodes print correctly.

Setting up the Zebra Device Module

The Zebra device module contains commands similar to the following:

```
LANGUAGE "ZPL"
UNITS "DPI300"

PAPER "L4x1", "", 300, 1200,  0,0
PAPER "L4x3", "", 900, 1200,  0,0
PAPER "L4x5", "", 1500, 1200,  0,0
PAPER "L4x6", "", 1800, 1200,  0,0

RESOLUTION 203.2, 203.2
```

The PAPER commands define names for various label sizes, such as “L4x3”. The sizes are measured in the units specified by the UNITS command. If you want to use a label size other than the sizes already specified, you may add a new PAPER command, defining a name for the label size you want to use. The syntax for the PAPER command is

```
PAPER "name", "", height, width,  0,0
```

where *name* is the name for the new label size, *height* is the height in units, and *width* is the width in units.

Because Zebra provides printers in several DPI configurations, the printer resolution must also be specified in the device module using the RESOLUTION command. Ensure that the resolution specified in your device module matches your printer resolution.

The Zebra device module provides access to the built-in bitmapped fonts available in most Zebra printers, as well as a selection of sizes for the built-in scalable font. To add a

new font, or a new font size for an existing font, add a new font command to specify the characteristics of the new font or font size. For example, to add a new size for the scalable font, add the command

```
FONT "fontname" TYPEFACE "0" SYMBOLS "0" POINTS height,width
```

where *fontname* is the name of the font, and *height* and *width* indicate the character size.

For more information on device modules, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT). For more information on the FONT command, see Chapter 8 of the *Advanced Features Guide*.

Note: Make a copy of the device module file, in order to preserve your changes to the device module when installing an upgrade or reinstalling OptioDCS.

Setting up your Zebra Label

To print a document as a label on a Zebra printer, you simply select the Zebra device and the desired label size. The sections below explain how to make these selections in the document and format modules. Otherwise, you create the label just as any other OptioDCS document, using DCL commands.

There are several sample labels included with the Zebra driver that may be used as examples; these are located in the ~/doc directory. For more information on document and format modules, and the commands shown below, see the *User's Guide* (Chapter 5 for Unix; Chapter 3 for Windows NT).

Document Module

To select the Zebra device, add the command

```
SET DEVICE "~dev/zebra"
```

to your document module.

Format Module

To select a label size, use the PAPER clause at the beginning of each format module as shown below.

```
FORM "FileLabel"
   PAPER "L4x5"
```

This example selects the size named "L4x5". The label size must be defined in the device module, as described on page 2. The PAPER clause is a header clause, so it must be placed at the beginning of the format module.

Using DCL Language Extensions

By using the Zebra extensions to the DCL language in your documents, you may take advantage of the unique features of your Zebra printer. The basic format of the extension commands is

```
EXTENSION "PARAMETER=value"
```

where *PARAMETER* is the name of a printing parameter and *value* is a literal value to assign to the parameter. If you want to use a variable to assign the value, use the syntax

```
EXTENSION "PARAMETER=" & varname
```

where *PARAMETER* is the name of the parameter and *varname* is the variable containing the value.

The extension commands affect various aspects of printing labels, such as line color and image magnification. Each extension command is described in the sections below.

General Extensions

FORM

A command used to indicate whether the printer should save the static text and graphics of the format. It may be placed in a format, document, or part module. Normally, when OptioDCS sends the first label of a job to the printer, the printer saves the static elements of the format until the last label of the job is printed. When subsequent labels call the same format module, the static elements of the format are not sent to the printer again, and the saved format is used. You may use this extension to alter this process in the following ways:

- To have OptioDCS send the static elements of the format to the printer each time the format module is called, use the command

  ```
  EXTENSION "FORM=YES"
  ```

- To have OptioDCS delete the format from the printer after printing each label, use the command

  ```
  EXTENSION "FORM=DELETE"
  ```

- To have OptioDCS not send the static elements of the format to the printer for the current label, use the command

  ```
  EXTENSION "FORM=NO"
  ```

  To print labels with this command, you must have called this format previously, without using "FORM=NO" or "FORM=DELETE".

HOME

A command used to indicate where the origin, or home position, should be located on the label. It may be placed in a format, segment, document, or part module. The default origin is (0,0), the upper left corner. To set a different origin, use the command

```
EXTENSION "HOME=homex,homey"
```

where *homex* and *homey* are the horizontal and vertical distances from the upper left corner to the origin, measured in printer dots. All elements in the format module are printed in relation to this origin.

SET

A header clause for format modules, used to print multiple identical copies of labels, with no changes to any incrementing (or counter) text or barcodes. (Any incrementing elements printed on multiple copies specified by the SET COPIES command are incremented with each copy.) Header clauses must be placed immediately after the FORM or FORMAT command at the beginning of the format module, as shown below.

```
FORMAT "shform"
   FEEDER "Lower"
   EXTENSION "SET=3"
```

In this example, the header clauses select the lower paper tray and cause three identical labels to print. The header clauses may be in any order.

The total number of labels printed for each page of input data is the number of copies mulitiplied by the number of sets. For example, the document below prints four labels for each page of data, and any incrementing elements are incremented for each copy. (The format module does not have a SET header clause.)

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 4
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   ⋮
END FORM
```

The document below also prints four labels for each page of data, but the incrementing elements are only incremented once for each page. This produces four identical labels for each page.

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 1
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   EXTENSION "SET=4"
   ⋮
END FORM
```

The document below prints four sets of four labels for each page of data, for a total of 16 labels per page. Any incrementing elements are incremented four times for each page.

```
DOCUMENT "shelflabl"
   ⋮
   SET COPIES 4
   ⋮
   DRAW "shform"
   ⋮
END DOCUMENT

FORM "shform"
   EXTENSION "SET=4"
   ⋮
END FORM
```

Text Extensions

METHOD

A clause for the DRAW TEXT command, used to print characters from the Zebra graphic symbol font. The only setting is "G". The character size is determined by the specified font (the FONT clause). To select a character, you use the corresponding alphabetic character. For example, the command below uses "A" to select the first character in the graphic symbol font, which is the registered trademark symbol. The size information of the ST4 font is used.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "METHOD=G" USING "A"
```

HORIZONTAL

A clause for the DRAW TEXT command, used to change the horizontal dimensions of a scalable font temporarily, in printer dots. For example, the command below prints "hello" using the font ST4, but changes the width to 100 dots.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "HORIZONTAL=100"
   USING "hello"
```

VERTICAL

A clause for the DRAW TEXT command, used to change the vertical dimensions of a scalable font temporarily, in printer dots. For example, the command below prints "hello" using the font ST4, but changes the height to 100 dots.

```
DRAW TEXT AT 1,1.7 FONT "ST4" EXTENSION "VERTICAL=100"
   USING "hello"
```

Text Counter Extensions

STEP

A clause for the DRAW TEXT command, used to set the increment for incrementing text. The default value is "+1". This may be used with the ZEROS clause as shown below.

```
DRAW TEXT AT 1,1.8 USING itemnum FONT "ST4"
   EXTENSION "STEP=+15","ZEROS=Y"
```

ZEROS

A clause for the DRAW TEXT command, used to indicate whether to print leading zeros. The default value is "N", indicating not to print leading zeros. This may be used with the STEP clause as shown below.

```
DRAW TEXT AT 1,1.8 USING itemnum FONT "ST4"
   EXTENSION "STEP=+15","ZEROS=Y"
```

COUNTER

A clause for the DRAW TEXT command, used both to set the increment for an incrementing text and to indicate whether to print leading zeros. The example below indicates printing leading zeros and incrementing by 15.

```
DRAW TEXT AT 1,1.8 USING itemnum FONT "ST4"
   EXTENSION "COUNTER=+15,Y"
```

Line and Box Extension

COLOR

A clause for the DRAW LINE and DRAW BOX command, used to change the line color to white (from the default black). To specify white lines, use the extension

```
EXTENSION "COLOR=WHITE"
```

Image Extensions

HORIZONTAL

A clause for the DRAW IMAGE command, used to set the horizontal magnification factor of the image. The default value is "1". For example, the command below prints the image **logo**, with double horizontal magnification.

```
DRAW IMAGE AT 1.0,6.0 USING logo
   EXTENSION "HORIZONTAL=2","VERTICAL=3"
```

VERTICAL

A clause for the DRAW IMAGE command, used to set the vertical magnification factor of the image. The default value is "1". For example, the command below prints the image **logo**, with double vertical magnification.

```
DRAW IMAGE AT 1.0,6.0 USING logo
   EXTENSION "HORIZONTAL=3",“VERTICAL=2”
```

Barcode Counter Extensions

STEP

A clause for the DRAW BARCODE command, used to set the increment for an incrementing barcode. The default value is "+1". This may be used with the ZEROS clause as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODE39”
   EXTENSION "STEP=+15","ZEROS=Y"
```

ZEROS

A clause for the DRAW BARCODE command, used to indicate whether to print leading zeros. The default value is "N", indicating not to print leading zeros. This may be used with the STEP clause as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODE39”
   EXTENSION "STEP=+15","ZEROS=Y"
```

COUNTER

A clause for the DRAW BARCODE command, used both to set the increment for an incrementing barcode and to indicate whether to print leading zeros. The example below indicates printing leading zeros and incrementing by 15.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “CODE39”
   EXTENSION "COUNTER=+15,Y"
```

Start/Stop Code Extensions

START

A clause for the DRAW BARCODE command, used to select the start character for Codabar and Code49 barcodes. The default value is "A".
Valid values for Codabar barcodes are A, B, C, D, *, N, E, and T.
Valid values for Code49 barcodes are A, 0, 1, 2, 3, 4, and 5.
The example below selects the start code "4" for a Code49 barcode.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE "CODE49"
   EXTENSION "START=4"
```

STOP

A clause for the DRAW BARCODE command, used to select the stop character for Codabar barcodes. The default value is "A".
Valid values are A, B, C, D, *, N, E, and T.
This is usually used with the START clause as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE "CODABAR"
   EXTENSION "START=C","STOP=*"
```

MSI Check Digit Extension

MSICD

A clause for the DRAW BARCODE command, used to select the check digit type for MSI barcodes. The default value is "A", indicating no check digit.
Valid values are A, B, C, and D.
The example below selects the type of check digit corresponding to C.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE "MSI"
   EXTENSION "MSICD=C"
```

PDF 417 Barcode Extensions

SECURITY

A clause for the DRAW BARCODE command, used to select the security level for PDF 417 barcodes.
Valid values are 0, 1, 2, 3, 4, 5, 6, 7, and 8.
This is usually used with the other PDF 417 clauses as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE "PDF417" EXTENSION
 "SECURITY=4","HEIGHT=20","ROWS=3","COLUMNS=10","TRUNCATE=Y"
```

HEIGHT

A clause for the DRAW BARCODE command, used to select the bar height for each row of a PDF 417 barcode. The default value is 10 dots.
Valid values range from 1 to the height of the label, in printer dots.

This is usually used with the other PDF 417 clauses as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “PDF417” EXTENSION
 "SECURITY=4","HEIGHT=20","ROWS=3","COLUMNS=10","TRUNCATE=Y"
```

ROWS

A clause for the DRAW BARCODE command, used to specify the maximum number of data rows to encode in a PDF 417 barcode. This is usually used with the other PDF 417 clauses as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “PDF417” EXTENSION
 "SECURITY=4","HEIGHT=20","ROWS=3","COLUMNS=10","TRUNCATE=Y"
```

If the ROWS and COLUMNS extensions are not used, the printer defaults to 1:2 ratio of rows to columns.

COLUMNS

A clause for the DRAW BARCODE command, used to specify the maximum number of data columns to encode in a PDF 417 barcode. This is usually used with the other PDF 417 clauses as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “PDF417” EXTENSION
 "SECURITY=4","HEIGHT=20","ROWS=3","COLUMNS=10","TRUNCATE=Y"
```

If the ROWS and COLUMNS extensions are not used, the printer defaults to 1:2 ratio of rows to columns.

TRUNCATE

A clause for the DRAW BARCODE command, used to indicate whether to truncate the stop pattern for a PDF 417 barcode.
The default value is “N”, meaning do not truncate.
This is usually used with the other PDF 417 clauses as shown below.

```
DRAW BARCODE AT 1,1.8 USING itemnum TYPE “PDF417” EXTENSION
 "SECURITY=4","HEIGHT=20","ROWS=3","COLUMNS=10","TRUNCATE=Y"
```

Barcode Reference

This section lists the supported barcode types and styles. See the barcode extension sections (starting on page 45) for the language extensions that apply to each barcode type. For more information on a particular barcode, see your printer documentation.

Types and Keywords

The following table lists the barcode types that are supported by OptioDCS for printing using the Zebra printer driver. The types are listed in alphabetical order, along with the valid DCL keyword used to specify the type.

Type	Keyword
Codabar	CODABAR
Code 2 of 5, interleaved	INT25
Code 2 of 5, interleaved + check digit	INT25+
Code 2 of 5, industrial	IND25
Code 2 of 5, standard	STA25
Code 3 of 9	CODE39
Code 3 of 9 + check digit	CODE39+

Type	Keyword
Code 11	CODE11
Code 11 + check digit	CODE11+
Code 49 two-dimensional barcode	CODE49
Code 128 Automatic	CODE128
Code 128 Automatic + check digit	CODE128+
Code 128 mode A	CODE128A
Code 128 mode A + check digit	CODE128A+
Code 128 mode B	CODE128B
Code 128 mode B + check digit	CODE128B+
Code 128 mode C	CODE128C
Code 128 mode C + check digit	CODE128C+
Compressed code 3 of 9	CODE93
Compressed code 3 of 9 + check digit	CODE93+
EAN/JAN-8	EAN8
EAN/JAN-13	EAN13
Logmars	LOGMARS
MSI	MSI
PDF417 two-dimensional barcode	PDF417
Plessey	PLESSEY
Plessey + check digit	PLESSEY+
POSTNET (ZIP+4) postal code	POSTNET
UPC Type A	UPCA
UPC Type A & Price check digit	UPCA+
UPC Type E	UPCE
UPC Type E + check digit	UPCE+

Styles

A barcode style is set using the STYLE clause with the SET or DRAW BARCODE command. The style determines whether human readable text is printed with the barcode. The keywords are to set the style are listed in the table below.

Keyword	**Style**
NOTEXT, TEXT0	without text
TEXT1	text below the barcode
TEXT2	text above the barcode
TEXT3	text and check digit below the barcode

The default setting is without text. The style does not apply to the PDF417 or POSTNET barcodes because these barcodes type does not include human readable text.

Using a GDI Print Device

Using a Graphics Device Interface device module enables you to use any Windows NT print driver for a label printer for OptioDCS output. OptioDCS for Windows NT includes a program that creates Graphics Device Interface, or GDI, device modules. The Windows NT print driver must match the model of the label printer, and the DPI level used by the printer if necessary. The driver must be configured properly to work with the label printer.

Note: This feature is for OptioDCS for Windows NT only.

Creating a GDI Device

To create a GDI device module, use the Make GDI Device Files program provided with OptioDCS. The Windows NT printer that the device module uses must be added on the NT Server that is used to run OptioDCS. You may have to add paper and font statements to the device module, to indicate the label sizes and fonts supported by the label printer. For more information on creating a GDI device module, see Chapter 7 of the *Advanced Features Guide*.

Document Guidelines

When using a GDI device module with your document, use the following guidelines:

- Select the GDI device in your document or part module.
- Some printer drivers do not support all features of OptioDCS documents. For more information, see the documentation for your printer driver.
- The device supports the Windows NT fonts that are available through the printer driver and specified in the device module. You can add font aliases for these fonts using the TYPEFACE clause with the Windows NT typeface name and style. For more information, see Chapter 7 of the *Advanced Features Guide*.