MICROSOFT EXCEL BINARY FILE FORMAT

Mark O'Brien
Microsoft Corporation
18-Feb-1988

Table of Contents

Introduction

BIFF (BInary File Format) is the file format in which Excel documents are saved on disk. A BIFF file is a complete description of an Excel document.

BIFF was designed to satisfy the following goals:

- Easy to understand and use
- Easy to expand the file format for future needs
- Files should save and load quickly

BIFF files consist of sequences of variable-length records. There are many different types of BIFF records. For example, one record type describes a formula entered into a cell; one describes the size and location of a window into a document; another describes a picture format.

General BIFF Record Format

Although different BIFF record types contain different information, every record follows the same basic format:

- Record type. This tells us what kind of data the record contains (e.g. a formula, a window, or a picture format).
- Record length. This tells us how long the data contained in the record is. The length of a record depends on the type of data it contains. For example, a window record may always be the same length, containing just the size and location of a window, while a formula record varies in length, depending on the length of the formula itself.
- Record data. This is the variable-length portion of the record containing the actual data.

All BIFF records are in the following format:

Offset	Length	Contents
0	word	record type
2	word	length of data portion
4	varies	data portion of record

The data portion of a BIFF record must be no longer than 2080 bytes long. Thus, counting the record type and length fields, the maximum length of a BIFF record is 2084 bytes.

Within this document, all numbers are decimal numbers unless they are preceded by "0x", in which case they are hexadecimal.

Some portions of BIFF records are marked as RESERVED. These portions are unavailable for application use. If they are marked "RESERVED - must be zero", then a BIFF-related application should ensure that their contents are always filled with zeros. If they are marked simply RESERVED, then they do not need to be set to any particular value.

Rows and Columns Within BIFF

Within BIFF files, rows and columns are always stored zero-based, rather than one-based as they appear on the screen. For example, cell A1 is stored as row 0, column 0; cell B3 is row 2, column 1.

Cell Table - Concepts

Microsoft Excel uses a sparse cell table to reduce memory requirements as much as possible. Cells that don't have values or formulas in them, have default format attributes, and are not referenced in any other formulas, are undefined cells and do not have any memory allocated for them.

For example, if a worksheet has a value in cell A3 and the formula =A3+A4 in cell B10, then the only defined cells on the worksheet are

A3, A4, and B10. No other cells need to exist. Entire rows can be undefined, if they have no defined cells in them. In this case, only rows 3, 4, and 10 are defined.

Cell Records

The term "cell record" refers to a BIFF record that defines a cell on an Excel document. A cell record is one of the following types:

BLANK
INTEGER
NUMBER
LABEL
BOOLERR
FORMULA

The following records can occur in conjunction with cell records:

CONTINUE
ARRAY
TABLE
TABLE2

Record Types

Here are the record types defined in BIFF. The record type and record length fields have been omitted from the descriptions, but they are present in every record.

BOF record - beginning of file (type = 9)

Offset	Name	Size	Contents
4	vers	2	version number
6	dt	2	document type 0x10 = worksheet 0x40 = macro sheet

Description

The BOF must be the first record in every BIFF file. The version number for Excel documents is currently 2; Microsoft may change this number in the future, as BIFF is modified for future needs.

Currently defined version numbers are:

Value	Name	Meaning
2	versExcel	Excel document
3	versMP	Multiplan document

All other version numbers are reserved for future use by Microsoft.

The high byte of the version number contains flag bits.

Current flag values are:

Mask	Name	Meaning
0x0100	bitFMP	=1 if the BIFF file is a Multiplan document
0xFE00		RESERVED for future use - must be zeros

The dt field specifies whether the document is a worksheet or a macro sheet. Chart BIFF files are different and are not described in this document.

FILEPASS record - file password key (type = 47)

Description

The FILEPASS record is used for an Excel document which was saved with a password in the File Save As command. If this record appears, it must directly follow the BOF record. All subsequent BIFF records will be encrypted, so you cannot read a password-protected BIFF file.

Note that this record specifies a file password, as opposed to the PASSWORD record, which specifies a document password.

INDEX record - ROW record index (type = 11)

Offset	Name	Size	Contents
4	ibRtName	4	absolute file position of the first NAME record
8	rwMic	2	first row that exists on the document
10	rwMac	2	last row that exists on the document, plus 1
12	rgibRw	var	array of absolute file positions of the blocks of ROW records

Description

The INDEX record is used to optimize searching through a file for a particular cell or name. This record is optional; if it occurs, it must occur directly after the document's FILEPASS record. If the document has no FILEPASS record, then the INDEX record must occur directly after the BOF record.

The ibRtName field gives the absolute file offset (0 = beginning of file) of the first NAME record. rwMic and rwMac are the range of defined rows in the document. The rgibRw field is an array of 4-byte absolute file offsets to the document's ROW records.

Excel always writes an INDEX record when it saves a BIFF file. If you are writing a BIFF file, you should probably not attempt to write an INDEX record.

The INDEX record is explained more fully in the section "Finding Values From BIFF Files."

CALCCOUNT record - iteration count (type = 12)

Offset	Name	Size	Contents
4	cIter	2	iteration count

Description

The CALCCOUNT record specifies the iteration count as set in the Options Calculation command.

CALCMODE record - calculation mode (type = 13)

Offset	Name	Size	Contents
4	fAutoRecalc	2	calculation mode =0 for manual =1 for automatic =-1 for automatic, no tables

Description

The CALCMODE record specifies the calculation mode as set in the Options Calculation command.

PRECISION record - precision (type = 14)

Offset	Name	Size	Contents
4	fFullPrec	2	document precision =1 for full precision =0 for precision as displayed

Description

The PRECISION record specifies the precision as set in the Options Calculation command.

REFMODE record - reference mode (type = 15)

Offset	Name	Size	Contents
4	fRefA1	2	reference mode =1 for A1 mode =0 for R1C1 mode

Description

The REFMODE record specifies the reference mode as set in the Options Desktop command.

DELTA record - maximum iteration change (type = 16)

Offset	Name	Size	Contents
4	numDelta	8	maximum change for iteration

Description

The DELTA record specifies the maximum change for an

iterative model, as set in the Options Calculation command.
The number is in 8-byte IEEE floating point format.

ITERATION record - iteration flag (type = 17)

Offset	Name	Size	Contents
4	fIter	2	iteration flag =1 for iteration on =0 for iteration off

Description
The ITERATION record specifes the state of iteration as set in the Options Calculation command.

1904 record - date system (type = 34)

Offset	Name	Size	Contents
4	f1904	2	=1 if the document uses the 1904 date system =0 otherwise

Description
The 1904 record specifies the date system used in an Excel document, as specified in the Options Calculation command.

BACKUP record - file backup option (type = 64)

Offset	Name	Size	Contents
4	fBackupFile	2	=1 if Excel should save a backup version of the file when it is saved =0 otherwise

Description
The BACKUP record specifies whether or not Excel should save backup versions of a BIFF file, as specified in the "Create Backup File" checkbox in the Save As dialog box.

PRINT ROW HEADERS record - print row headers flag (type = 42)

Offset	Name	Size	Contents
4	fPrintRwCol	2	=1 if we should print row and column headers when printing the document =0 otherwise

Description
The PRINT ROW HEADERS record controls whether or not Excel prints row and column headers when printing the document.

PRINT GRIDLINES record - print gridlines flag (type = 43)

Offset	Name	Size	Contents

4	fPrintGrid	2	=1 if we should print gridlines when printing the document =0 otherwise

Description

The PRINT GRIDLINES record controls whether or not Excel prints gridlines when printing the document.

HORIZONTAL PAGE BREAKS record - row page breaks (type = 27)

Offset	Name	Size	Contents
4	cbrk	2	number of page breaks
6	rgrw	var	array of rows

Description

The HORIZONTAL PAGE BREAKS record contains a list of explicit row page breaks. The cbrk field contains the number of page breaks. rgrw is an array of 2-byte integers specifying rows. Excel sets a page break before each row in the list. The rows must be sorted in increasing order.

VERTICAL PAGE BREAKS record - column page breaks (type = 26)

Offset	Name	Size	Contents
4	cbrk	2	number of page breaks
6	rgcol	var	array of columns

Description

The VERTICAL PAGE BREAKS record contains a list of explicit column page breaks. The cbrk field contains the number of page breaks. rgcol is an array of 2-byte integers specifying columns. Excel sets a page break before each column in the list. The columns must be sorted in increasing order.

DEFAULT ROW HEIGHT record - default row height (type = 37)

Offset	Name	Size	Contents
4	miyRwGhost	2	default row height

Description

The DEFAULT ROW HEIGHT record specifies the height of all undefined rows in the document. The miyRwGhost field contains the row height in units of 1/20 of a point. This record does not affect the row heights of any rows that are explicitly defined.

FONT record - document font (type = 49)

Offset	Name	Size	Contents
4	dy	2	height of the font
6	grbit	2	font attributes

8	cch	1	length of font name
9	rgch	var	the font name

Description

The FONT record describes an entry in the Excel document's font table. There are up to four different fonts on an Excel document, numbered 0 to 3. FONT records are read into the font table in the order in which they are encountered in the BIFF file.

The dy field gives the height of the font in units of 1/20 of a point. grbit contains the font attributes as follows:

Offset	Bits	Mask	Name	Contents
0	7-0	0xFF	RESERVED - must be zeros	
1	7-4	0xF0	RESERVED - must be zeros	
	3	0x08	fStrikeout	=1 if the font is struck out
	2	0x04	fUnderline	=1 if the font is underlined
	1	0x02	fItalic	=1 if the font is italic
	0	0x01	fBold	=1 if the font is bold

cch and rgch contain the font's face name.

FONT2 record - more font information (type = 50)

Description

The FONT2 record contains system-specific information about the font defined in the previous FONT record. This record is optional. If you are writing a BIFF file, do not write a FONT2 record.

HEADER record - print header string (type = 20)

Offset	Name	Size	Contents
4	cch	1	length of string
5	rgch	var	the string

Description

The HEADER record specifies a print header string for a document. This string appears at the top of every page when the document is printed.

FOOTER record - print footer string (type = 21)

Offset	Name	Size	Contents
4	cch	1	length of string
5	rgch	var	the string

Description

The FOOTER record specifies a print footer string for a document. This string appears at the bottom of every page when the document is printed.

LEFT MARGIN record - left print margin (type = 38)

Offset	Name	Size	Contents
4	num	8	left margin

Description

The LEFT MARGIN record specifies the left margin in inches when a document is printed. The num field is in 8-byte IEEE floating point format.

RIGHT MARGIN record - (type = 39)

Offset	Name	Size	Contents
4	num	8	right margin

Description

The RIGHT MARGIN record specifies the right margin in inches when a document is printed. The num field is in 8-byte IEEE floating point format.

TOP MARGIN record - (type = 40)

Offset	Name	Size	Contents
4	num	8	top margin

Description

The TOP MARGIN record specifies the top margin in inches when a document is printed. The num field is in 8-byte IEEE floating point format.

BOTTOM MARGIN record - (type = 41)

Offset	Name	Size	Contents
4	num	8	bottom margin

Description

The BOTTOM MARGIN record specifies the bottom margin in inches when a document is printed. The num field is in 8-byte IEEE floating point format.

COLWIDTH record - column width (type = 36)

Offset	Name	Size	Contents
4	colFirst	1	first column in the range
5	colLast	1	last column in the range
6	dx	2	column width

Description

The COLWIDTH record sets the column width for a range of columns specified by colFirst and colLast. The dx field is an unsigned integer specifying the column width in units of 1/256 of a character.

EXTERNCOUNT record - count of externally referenced documents (type = 22)

Offset	Name	Size	Contents
4	cxals	2	number of externally referenced documents

Description

The EXTERNCOUNT record specifies the number of documents that are referenced externally from an Excel document.

Both external references and Dynamic Data Exchange (DDE) references are counted here. For external references, only the supporting sheet name counts. For DDE references, the application-topic pair counts.

For example, suppose a worksheet contains the following formulas:

```
=SALES.XLS!Gross-SALES.XLS!Profits
=Signal|System!Formats
=Signal|StockInfo!IBM
```

This worksheet would have an EXTERNCOUNT of three: SALES.XLS, Signal|System, and Signal|StockInfo.

EXTERNSHEET record - externally referenced document (type = 23)

Offset	Name	Size	Contents
4	cch	1	length of document name
5	rgch	var	document name

Description

The EXTERNSHEET record specifes a document which is referenced externally from an Excel document. There must be as many EXTERNSHEET records in a BIFF file as were specified in the EXTERNCOUNT record. The order of EXTERNSHEET records in a BIFF file is important and should not be changed.

The document that is externally referenced is called the supporting document. The document which refers to it is called the dependent document.

The cch field gives the length of the supporting document name, which is contained in the rgch field. Whenever possible, document names are encoded to make BIFF files compatible with file systems other than DOS. Encoded document names are identified by the first character of the rgch field. The following special characters are recognized:

Name	Value	Meaning
chEmpty	0	empty sheetname
chEncode	1	encoded pathname
chSelf	2	self-referential external reference

chEmpty is used to store an external reference to the empty sheet, as in the formula =!$A$1. chSelf is used to store an external reference where the dependent and supporting documents are the same, for example a worksheet SALES.XLS which contains the formula =SALES.XLS!$A$1.

chEncode is used when the DOS file name of the supporting document has been translated to a less system-dependent name. The following special characters are recognized in an encoded document name:

Name	Value	Related DOS keys
chVolume	1	:
chSameVolume	2	none
chDownDir	3	.\
chUpDir	4	..\

The chVolume key is used to specify a DOS drive letter in a document name. It is followed by the drive letter. This replaces the DOS-specific ':' character, as in =C:SALES.XLS!Gross.

The chSameVolume key is used when the drive letter was omitted, to indicate that the supporting document is on the same DOS drive as the dependent document, as in =SALES.XLS!Gross.

The chDownDir key is used to go down a directory level. It is followed by the subdirectory name. This replaces the implicit DOS-specific sequence ".\", meaning subdirectory of the current directory. An example of such an external reference is =AUGUST\SALES.XLS!Gross.

The chUpDir key is used to go up a directory level. It replaces the DOS-specific sequence "..\", meaning the parent directory of the current directory.

DDE references are encoded differently. Only one translation is ever performed on a DDE reference, on the '|' character:

Name	Value	Related DOS keys
chDde	3	|

EXTERNNAME record - externally referenced name (type = 35)

Offset	Name	Size	Contents
4	cch	1	length of the name
5	rgch	var	the name

Description

The EXTERNNAME record specifes a name which is referenced externally from an Excel document. All EXTERNNAME records associated with a supporting document must directly follow the EXTERNSHEET record for the document.

The order of EXTERNNAME records in a BIFF file is important and should not be changed.

An externally referenced name is one of the following:

- A worksheet or macro sheet name in an external reference. In the formula =SALES.XLS!Gross, the name "Gross" is an externally referenced name.
- A DDE topic. In the formula =Signal|StockInfo!IBM, the topic "IBM" is an externally referenced name.

When the externally referenced name is a DDE topic, Excel may append the most recent values for the topic to the EXTERNNAME record. The values are written in the same format as array constant values in parsed expressions. See the explanation of "ptgArray" in the "Operand Tokens - Base" section for a full description of this format.

If there are many values, the EXTERNNAME record may become so long that it must be split into multiple records. In this case, the EXTERNNAME record will be followed by one or more CONTINUE records.

FORMAT record - cell format (type = 30)

Offset	Name	Size	Contents
4	cch	1	length of format string
5	rgch	var	picture format string

Description

The FORMAT record describes a picture format on the document.

All the FORMAT records should appear together in a BIFF file. The order of FORMAT records in an existing BIFF file is important and should not be changed. You can add new formats to a file, but they should be added at the end of the FORMAT list.

NAME record - user-defined name (type = 24)

Offset	Name	Size	Contents

4	grbit	1	name attributes
5	grbitPli	1	name attributes
6	chKey	1	keyboard shortcut
7	cch	1	length of the name text
8	cce	1	length of the name's definition
9	rgch	var	text of the name
var	rgce	var	parsed expression for the name's definition
var	cceDup	1	length of the name's definition (this is a duplicate of the cce field)

Description

The NAME record describes a user-defined name on the document. The cch field contains the length the name text; the text itself is in rgch. cce is the length of the name definition, and rgce contains the definition. The location of rgce within the record depends on the length of the name text. Following rgce, the length of the name definition appears again.

The name definition is stored in Excel's internal compressed format. See the section "Excel Formulas" for an explanation.

The grbit field contains bit attributes of the name:

Bits	Mask	Name	Contents
7-3	0xF8		RESERVED - must be zeros
2	0x04	fCalcExp	=1 if the name contains a complex function =0 otherwise
1	0x02	fProc	=1 for a Function or Command name =0 otherwise
0	0x01		RESERVED - must be zero

The fCalcExp bit is set if the name definition contains one or more of the following:

- A function that returns an array (e.g. TREND, MINVERSE)
- The ROW or COLUMN function
- A user-defined function

The fProc bit is set if the name is a Function or Command name on a macro sheet.

grbitPli and chKey are meaningful only when the fProc bit is set in the grbit field. grbitPli contains bit attributes for Function or Command names:

Bits	Mask	Name	Contents

7-2	0xFC		RESERVED - must be zeros
1	0x02	fRun	=1 for Command names
0	0x01	fFunc	=1 for Function names

chKey is the keyboard shortcut for a Command name. If the name is not a command name or has no keyboard shortcut, then chKey will be 0.

All the NAME records should appear together in a BIFF file. The order of NAME records in an existing BIFF file is important and should not be changed. You can add new names to a file, but they should be added at the end of the NAME list. Excel saves out the names in alphabetical order, but this is not a requirement; Excel will sort the name list, if necessary, when it loads a BIFF file.

DIMENSIONS record - cell table size (type = 0)

Offset	Name	Size	Contents
4	rwMic	2	first defined row on the document
6	rwMac	2	last defined row on the document, plus 1
8	colMic	2	first defined column on the document
10	colMac	2	last defined column on the document, plus 1

Description

The DIMENSIONS record contains the minimum and maximum bounds of the document. It tells us very quickly the approximate size of the document.

Note that both the rwMac and colMac fields are 1 greater than the actual last row and column. For example, for a worksheet that exists between cells B3 and D6, we would have rwMic = 2, colMic = 1, rwMac = 6, colMac = 4.

COLUMN DEFAULT record - default cell attributes (type = 32)

Offset	Name	Size	Contents
4	colMic	2	first column that has a default cell
6	colMac	2	last column that has a default cell, plus 1
8	rgrgbAttr	var	array of default cell attributes

Description

The COLUMN DEFAULT record is an optional record that controls the formats of cells that aren't defined on the worksheet. This is a space-saving technique. By specifying a default cell for a particular column, you are telling Excel that all undefined cells in the column should have the specified cell attributes. Default cells do not affect the formats of cells that are explicitly defined.

For example, if you want all of column C to be left-aligned, then you could define all 16,384 cells in the column and specify that each one be left-aligned. This would require a large amount of storage to represent all 16,384 cells. Or, you could simply set the default cell for column C to be left-aligned, and not define any cells at all in column C.

The rgrgbAttr field is an array of rgbAttr fields, with the range of the array being colMic to colMac-1, inclusive. Each rgbAttr field is 3 bytes long. See the "Cell Attributes" section for a description of the rgbAttr field.

If the COLUMN DEFAULT record is present, it must appear in the file before any ROW records or cell records.

ROW record - row descriptor (type = 8)

Offset	Name	Size	Contents
4	rw	2	row number
6	colMic	2	first defined column in the row
8	colMac	2	last defined column in the row, plus 1
10	miyRw	2	row height
12	irwMac	2	Microsoft internal use
14	fDefault	1	=1 if the row has default cell attributes =0 otherwise
15	dbRtcell	2	relative file offset to the cell records for this row
17	rgbAttr	3	default cell attributes

Description

A ROW record describes a single row on an Excel document. colMic and colMac give the range of defined columns in the row. miyRw is the row height in units of 1/20 of a point. irwMac is used by Microsoft Excel to optimize loading the file; if you are creating a BIFF file, set this field to 0.

The miyRw field may have the 0x8000 bit set, indicating that the row is standard height. The low 15 bits must still contain the row height.

Each row can have default cell attributes which control the format of all undefined cells in the row. This is a space-saving technique. By specifying default cell attributes for a particular row, you are effectively formatting all the undefined cells in the row, but without using up memory for those cells. Default cell attributes do not affect the formats of cells that are explicitly defined.

For example, if you want all of row 3 to be left-aligned, then you could define all 256 cells in the row and specify that each one be left-aligned. This would require storage for each of the 256 cells. Or, you could simply set the default cell for row 3 to be left-aligned, and not define any cells at all in row 3.

The fDefault field indicates whether a default cell is present or not. If it is, then rgbAttr contains the default cell attributes. See the "Cell Attributes" section for a description of the rbgAttr field.

dbRtcell is a relative file offset to the cell records for the row. This is described in the section "Finding Values From BIFF Files."

BLANK record - blank cell (type = 1)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column
8	rgbAttr	3	cell attributes

Description

A BLANK record describes a cell with no formula or value.

See the "Cell Attributes" section for a description of the rgbAttr field.

INTEGER record - cell with constant integer (type = 2)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column
8	rgbAttr	3	cell attributes
11	w	2	unsigned integer value

Description

An INTEGER record describes a cell containing a constant unsigned integer in the range 0 - 65535. Negative numbers and numbers outside this range must be stored as NUMBER records.

See the "Cell Attributes" section for a description of the rgbAttr field.

NUMBER record - cell with constant floating point number (type = 3)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column

8	rgbAttr	3	cell attributes
11	num	8	floating point number value

Description

A NUMBER record describes a cell containing a constant floating point number. The number is in 8-byte IEEE floating point format.

See the "Cell Attributes" section for a description of the rgbAttr field.

LABEL record - cell with constant string (type = 4)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column
8	rgbAttr	3	cell attributes
11	cch	1	length of the string
12	rgch	var	the string

Description

A LABEL record describes a cell with a constant string. The string length is in the range 0 - 255.

See the "Cell Attributes" section for a description of the rgbAttr field.

BOOLERR record - cell with constant boolean or error (type = 5)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column
8	rgbAttr	3	cell attributes
11	bBoolErr	1	boolean or error value
12	fError	1	specifies boolean or error =1 for error =0 for boolean

Description

A BOOLERR record describes a cell containing a constant boolean or error value.

Boolean values are 1 for TRUE and 0 or FALSE.
Error values are as follows:

0	#NULL!
7	#DIV/0!
15	#VALUE!
23	#REF!
29	#NAME?
36	#NUM!
42	#N/A

See the "Cell Attributes" section for a description of the rgbAttr field.

FORMULA record - cell with a formula (type = 6)

Offset	Name	Size	Contents
4	rw	2	row
6	col	2	column
8	rgbAttr	3	cell attributes
11	num	8	current value of formula
19	sbRecalc	1	recalc flag =0 if the formula is calculated =nonzero if the formula needs to be calculated =3 if the formula is part of a matrix that needs to be calculated
20	cce	1	length of parsed expression
21	rgce	var	parsed expression

Description

A FORMULA record describes a cell with a formula.

The sbRecalc field tells us whether the formula needs to be recalculated upon loading the file. Normally, when formulas are saved in BIFF files, they are fully calculated. In some cases, however, this is not possible. If the formula contains a circular reference or a "volatile" function which can never be considered truly calculated, like RAND() or NOW(), then we indicate that the formula needs to be calculated upon loading.

Any nonzero value for sbRecalc indicates that the formula needs to be calculated. The special value of 3 is reserved for FORMULA records belonging to cells which are part of matrices, when the entire matrix itself needs to be calculated.

The num field contains the current value of the formula in 8-byte IEEE format. For formulas that evaluate not to numbers but to strings, booleans, or error values, the last two bytes of the num field will be 0xFFFF. This covers the sign bit, the exponent, and four bits of the fraction.

A boolean is stored in the num field as follows:

Offset	Name	Size	Contents
0	otBool	1	=1 always
1		1	RESERVED - must be zero
2	f	1	boolean value
3		3	RESERVED - must be zero
6	fExprO	2	=0xFFFF always

An error is stored in the num field as follows:

Offset	Name	Size	Contents
0	otErr	1	=2 always
1		1	RESERVED - must be zero
2	err	1	error value
3		3	RESERVED - must be zero
6	fExprO	2	=0xFFFF always

See the BOOLERR record for a description of boolean and error values.

A string is stored in the num field as follows:

Offset	Name	Size	Contents
0	otString	1	=0 always
1		5	RESERVED - must be zero
6	fExprO	2	=0xFFFF always

The string value itself is not stored in the num field; instead, it is stored in a separate BIFF record, the STRING record.

The parsed expression is the cell's formula, stored in Excel's internal compressed format. See the section "Excel Formulas" for an explanation.

See the "Cell Attributes" section for a description of the rgbAttr field.

ARRAY record - array formula (type = 33)

Offset	Name	Size	Contents
4	rwFirst	2	first row of the array
6	rwLast	2	last row of the array
8	colFirst	1	first column of the array
9	colLast	1	last column of the array
10	sbRecalc	1	recalc flag =0 if the array is calculated =nonzero if the array needs to be calculated
11	cce	1	length of parsed expression
12	rgce	var	parsed expression

Description

An ARRAY record describes a formula which was array-entered into a range of cells. The range in which the array is entered is given by rwFirst, rwLast, colFirst, and colLast.

The ARRAY record occurs directly after the FORMULA record for the upper left corner cell of the array, i.e. cell (rwFirst, colFirst).

The sbRecalc field tells whether the array needs to be recalculated upon loading or not. See the FORMULA record for a description of this field. Note that in an ARRAY record, unlike a FORMULA record, sbRecalc will never have the value 3.

The parsed expression is the array formula, stored in Excel's internal compressed format. See the section "Excel Formulas" for an explanation.

CONTINUE record - (type = 60)

Offset	Name	Size	Contents
4	rgce	var	parsed expression

Description

Some parsed formulas are so long that they are split up into sections and written out as separate records. The first section appears in the FORMULA or ARRAY record; subsequent sections appear in CONTINUE records. Parsed expressions will be discussed in detail in future documents.

Some EXTERNNAME records are also long enough to need CONTINUE records.

STRING record - string value of a formula (type = 7)

Offset	Name	Size	Contents
4	cch	1	length of the string
5	rgch	var	the string

Description

A STRING record appears after a FORMULA record whose formula currently evaluates to a string. If the formula is part of an array, then the STRING record occurs after the ARRAY record.

TABLE record - one-input table definition (type = 54)

Offset	Name	Size	Contents
4	rwFirst	2	first row of the table
6	rwLast	2	last row of the table
8	colFirst	1	first column of the table
9	colLast	1	last column of the table
10	sbRecalc	1	recalc flag =0 if the table is calculated =nonzero if the table needs to be calculated
11	fRw	1	=1 if this is a row input table =0 if this is a column input table
12	rwInp	2	row of the input cell
14	colInp	2	column of the input cell

Description

A TABLE record describes a one-input row or column table created through the Data Table command.

The area in which the table is entered is given by rwFirst, rwLast, colFirst, and colLast. This is the interior of the table; it does not include the outer row or column, which contains table formulas or input values.

The sbRecalc field tells whether the array needs to be recalculated upon loading or not. See the FORMULA record for a description of this field. Note that in an TABLE record, unlike a FORMULA record, sbRecalc will never have the value 3.

fRw tells us whether the input cell is a row input cell or a column input cell. In either case, the input cell is given by (rwInp, colInp).

rwInp is -1 in the case where the input cell is a deleted reference, i.e. displays as #REF!. colInp is unused in this case.

TABLE2 record - two-input table definition (type = 55)

Offset	Name	Size	Contents
4	rwFirst	2	first row of the table
6	rwLast	2	last row of the table
8	colFirst	1	first column of the table
9	colLast	1	last column of the table
10	sbRecalc	1	recalc flag =0 if the table is calculated =nonzero if the table needs to be calculated
11		1	RESERVED - must be zero
12	rwInpRw	2	row of the row input cell
14	colInpRw	2	column of the row input cell
16	rwInpCol	2	row of the column input cell
18	colInpCol	2	column of the column input cell

Description

A TABLE2 record describes a two-input table created through the Data Table command.

This record is the same as the TABLE record, with the following exceptions:

- There is no fRw field. The byte is unused.
- There are two input cells, a row input cell and a column input cell.
- Either input cell, or both input cells, may have a row of -1 to indicate that the corresponding input cell is a deleted

reference, i.e. displays as #REF!.

PROTECT record - worksheet protection (type = 18)

Offset	Name	Size	Contents
4	fLock	2	=1 if the document is protected =0 if the document is not protected

Description

The PROTECT record specifies whether or not an Excel document has been protected through the Options Protect Document command.

Note that this record specifies a document password, as opposed to the FILEPASS record, which specifies a file password.

WINDOW PROTECT record - window protection (type = 25)

Offset	Name	Size	Contents
4	fLockWn	2	=1 if the windows of the document are protected =0 otherwise

Description

The WINDOW PROTECT record specifies whether or not the document's windows are protected, as specified in the Protect Document command.

PASSWORD record - worksheet password (type = 19)

Offset	Name	Size	Contents
4	wPassword	2	encrypted password for a protected document

Description

The PASSWORD record contains the encrypted password for a document protected through the Options Protect Document command.

NOTE record - notes (type = 28)

Offset	Name	Size	Contents
4	rw	2	row of the note
6	col	2	column of the note
8	cch	2	length of the note
10	rgch	var	the note

Description

The NOTE record specifies a note associated with a cell. The cell is given by the rw and col fields. cch is the length of the note; rgch contains the text of the note.

Notes longer than 2048 characters must be spread across multiple NOTE records, each one containing at most 2048 characters. The first NOTE record contains the following fields:

Offset	Name	Size	Contents
4	rw	2	row of the note
6	col	2	column of the note
8	cch	2	total length of the note (>2048)
10	rgch	2048	the first 2048 characters of the note

Each subsequent NOTE record for the note contains the following fields:

Offset	Name	Size	Contents
4	rw	2	=-1 always
6		2	RESERVED - must be zero
8	cch	2	length of this section of the note (<=2048)
10	rgch	var	section of the note

WINDOW1 record - basic window information (type = 61)

Offset	Name	Size	Contents
4	x	2	horizontal position of the window
6	y	2	vertical position of the window
8	dx	2	width of the window
10	dy	2	height of the window
12	fHidden	1	=1 if the window is hidden =0 otherwise

Description

The WINDOW1 record provides basic information about an Excel window. The x and y fields give the location of the window in units of 1/20 of a point, relative to the upper left corner of the desktop. dx and dy give the window size, also in units of 1/20 of a point. fHidden is used to specify a hidden window.

If you are creating a BIFF file, you can omit the WINDOW1 record, and Excel will create a default window into your document.

WINDOW2 record - advanced window information (type = 62)

Offset	Name	Size	Contents
4	fDspFmla	1	=1 if the window should display

```
             formulas
          =0 if the window should display
             values
  5   fDspGrid 1  =1 if the window should display
             gridlines
          =0 otherwise
  6   fDspRwCol   1   =1 if the window should display
             row and column headers
          =0 otherwise
  7   fFrozen  1  =1 if the panes in the window
             should be frozen
          =0 otherwise
  8   fDspZeros   1   =1 if the window should display
             zero values
          =0 if the window should suppress
             display of zero values
  9   rwTop    2  top row visible in the window
  11  colLeft  2  leftmost column visible in the
          window
  13  fDefaultHdr  1  =1 if the row/column headers and
             gridlines should be drawn in
             the default foreground color
          =0 otherwise
  14  rgbHdr   4  row/column headers and gridline
          color
```

Description

The WINDOW2 record contains a fuller description of an Excel window. This record is optional. If it appears, it must directly follow the WINDOW1 record for the window it describes.

The fDspFmla, fDspGrid, fDspRwCol, and fDspZeros fields are window properties as set in the Options Display command. fFrozen is as set through the Options Freeze/Unfreeze Panes commands.

fDefaultHdr is 1 if the window's row and column headers and gridlines should be drawn in the window's default foreground color. If this field is 0, then the RGB color in rgbHdr is used instead.

PANE Record - window split information (type = 65)

```
  Offset  Name    Size Contents
  ------  ----    ---- --------
  4   x    2  horizontal position of the split,
          or zero if none
  6   y    2  vertical position of the split,
          or zero if none
  8   rwTop    2  top row visible in the bottom pane
  10  colLeft  2  leftmost column visible in the
          right pane
  12  pnnAct   1  pane number of the active pane
```

Description

The PANE record describes the number and position of panes in a window. The x and y fields give the position of the vertical and horizontal splits, respectively, in units of 1/20 of a point. Either of these fields may be zero, indicating that the window is not split in the corresponding direction.

For a window with a horizontal split, rwTop is the topmost row visible in the bottom pane or panes. For a window with a vertical split, colLeft gives the leftmost column visible in the right pane or panes.

The pnnAct field tells which pane is the active pane. It contains one of the following values:

0	Bottom right
1	Top right
2	Bottom left
3	Top left

If the document window associated with a pane has frozen panes, as specified in the WINDOW2 record, then x and y have special meaning. If there is a vertical split, then x contains the number of columns visible in the top pane. If there is a horizontal split, then y contains the number of rows visible in the left pane. Both types of splits can be present in a window, as in unfrozen panes.

SELECTION record - selection within a pane (type = 29)

Offset	Name	Size	Contents
4	pnn	1	pane number
5	rwAct	2	row number of the active cell
7	colAct	2	column number of the active cell
9	irefAct	2	reference number of the active cell
11	cref	2	number of references in the selection
13	rgref	var	array of references

Description

The SELECTION record specifies which cells are selected in a pane of a split window. This record may also be used to specify selected cells in a window which does not have any splits.

The pnn field tells which pane we are describing. It contains one of the following values:

0	Bottom right
1	Top right
2	Bottom left
3	Top left

For a window which has no splits, use pnn = 3.

rwAct and colAct specify which cell in the selection is the active cell.

The selection itself consists of rgref, a variable length array of references. The number of references in the record is given by the cref field. Each reference is six bytes long and contains the following fields:

Offset	Name	Size	Contents
0	rwFirst	2	first row in the reference
2	rwLast	2	last row in the reference
4	colFirst	1	first column in the reference
5	colLast	1	last column in the reference

irefAct is a zero-based index into the array of references, specifying which reference contains the active cell.

If a selection is so large that it won't fit in the maximum size BIFF record, 2084 bytes, then it is broken down into multiple consecutive SELECTION records. Each record contains a portion of the larger selection. Only the cref and rgref fields vary in the multiple records; the pnn, rwAct, colAct, and irefAct fields are the same over all records in the group. On each record, the cref field contains the number of references found on that record alone.

EOF record - end of file (type = 10)

Description

The EOF record must be the last record in the file. It has no data associated with it.

Cell Attributes

This section describes the cell attribute field found in the ROW, BLANK, INTEGER, NUMBER, LABEL, BOOLERR, FORMULA, and COLUMN DEFAULT records. The field is three bytes long and consists of bit fields:

Offset	Bits	Mask	Name	Contents
0	7	0x80	fHidden	=1 if the cell is hidden
	6	0x40	fLocked	=1 if the cell is locked
	5-0			RESERVED - must be zeros
1	7-6	0xC0	ifnt	font number
	5-0	0x3F	ifmt	the cell's format code
2	7	0x80	fShade	=1 if the cell is shaded
	6	0x40	fBottom	=1 if the cell has a bottom border
	5	0x20	fTop	=1 if the cell has a top border

4	0x10	fRight	=1 if the cell has a right border
3	0x08	fLeft	=1 if the cell has a left border
2-0	0x07	alc	the cell's alignment code

The ifnt field is a zero-based index into the document's table of fonts. The ifmt field is a zero-based index into the document's table of picture formats. See the FONT and FORMAT records for details.

The alc field has one of the following values:

0	General
1	Left
2	Center
3	Right
4	Fill
7	(Multiplan only) Default alignment

Order of Records

Here is the order in which records are written in a BIFF file:

BOF
FILEPASS
INDEX
CALCCOUNT
CALCMODE
PRECISION
REFMODE
DELTA
ITERATION
1904
BACKUP
PRINT ROW HEADERS
PRINT GRIDLINES
HORIZONTAL PAGE BREAKS
VERTICAL PAGE BREAKS
DEFAULT ROW HEIGHT
FONT
FONT2
HEADER
FOOTER
LEFT MARGIN
RIGHT MARGIN
TOP MARGIN
BOTTOM MARGIN
COLWIDTH
EXTERNCOUNT
EXTERNSHEET
EXTERNNAME
FORMAT
NAME
DIMENSIONS

```
   COLUMN DEFAULT
   Cell table
      ROW, BLANK, INTEGER, NUMBER, LABEL,
      BOOLERR, FORMULA, ARRAY, STRING,
      TABLE, TABLE2
   PROTECT
   WINDOW PROTECT
   PASSWORD
   NOTE
   WINDOW1
   WINDOW2
   PANE
   SELECTION
   EOF
```

Finding Values From BIFF Files

This section explains how to look up a cell value in a BIFF file, without having to load the file into Excel. You can look up values only in BIFF files that are not password-protected; protected BIFF files are encrypted and cannot be read.

One way to find the value of a particular cell in a BIFF file is to read every BIFF record, until we find a cell record for the cell. If we find one, we return its value. If we reach the EOF record without finding a cell record for the cell, then we return zero.

Fortunately, we don't have to go through such an exhaustive search. We can narrow down the area that we have to search by using BIFF's INDEX and ROW records.

If a non-protected BIFF file has an INDEX record, it will be the second record in the file (immediately after the BOF record). If the second record is not an INDEX record, then we must resort to the exhaustive record search described above. Or, alternatively, we could simply fail the search and return some sort of error code.

Having located the INDEX record, we fetch the rwMic and rwMac fields, which tell us the range of defined rows on the document. If the row we are searching for is outside of that range, then we know right away that the desired cell doesn't exist, so we can return zero.

The next step is to locate the ROW record for the row of the desired cell. To do this, we need to understand how Excel saves ROW records and cell records.

When Excel saves a document in BIFF format, it divides the document into blocks of 32 rows, starting at the first defined row on the document. Since rwMic is by definition the first defined row, the first block consists of rows rwMic through rwMic+31; the second, from rwMic+32 through rwMic+63; and in general, the i-th block, assuming that i is zero-based, consists of rows (rwMic+i*32) through (rwMic+i*32+31).

Excel writes a block of ROW records to the file, then follows this with all the cell records for cells in those rows. This process is repeated until all ROW and cell records have been written.

The INDEX record contains an array of file pointers to the blocks of ROW records. Working backwards from our rule above for ROW blocks, we see that to locate the block for row 'rw', we fetch array element (rw-rwMic)/32. Here, the '/' operator is integer division that truncates.

Having found the proper array element, we position the BIFF file at that location. The file pointer that we fetched from the array is an absolute byte offset from the beginning of the file, which is byte 0. For example, if the file pointer were 17,540, then we would position the file at byte 17,540.

The file is now positioned at the correct block of ROW records. The next step is to search for the correct ROW record. Since Excel documents have sparse cell tables, blocks of ROW records contain only the defined rows within the block range. This means that if the row we are searching for doesn't exist, then it won't have a ROW record in the BIFF file.

We must read at most 32 records at this point. If we do not find a ROW record for the desired row, then we know that the row doesn't exist, so we can return zero. We know that the row doesn't exist as soon as we find a non-ROW record, or a ROW record for a row beyond the one we are searching for.

Having found the correct ROW record, we fetch the colMic and colMac fields, which tell us the range of defined columns in the row. If the column we are searching for lies outside of the defined range, then we know that the desired cell doesn't exist, and we can return zero.

From the ROW record, we can now determine the position within the file of the cell records for the desired row. The next step is to position the file at that point and search for the cell record for the desired cell.

The dbRtcell field contains the offset to cells for the ROW record. This field is limited to 16 bits to save space in BIFF files; thus the largest offset that will fit is 65,535. In a large Excel document, however, it is possible for cells to be located farther than 65,535 bytes from their ROW record. Therefore we encode the offset to get more value from it.

> The first ROW record in a block contains an offset to cells relative to the second ROW record. This is because after reading the first ROW record, you are positioned at the second ROW record, so finding the cells is just a matter of skipping some number of

bytes relative to the current file position.

The second and all subsequent ROW records in a block contain offsets to cells relative to the previous ROW record's cells. This iterative approach works like this: after reading the first ROW record, you get its offset and add it to the current file position to get the absolute file position of the first ROW's cell records. When you read the second ROW record, you add the offset contained therein to your computed position of the first ROW's cells, and you get the position of the second ROW's cells. Continuing in this manner, you find that by the time you find the proper ROW record, you have already computed the absolute file position of its cells, so you position there and continue your search.

Having computed the file position of our row's cell records, we set the file there and start sequentially searching for the desired cell. If we find the cell, we fetch its value and return it. Our search fails as soon as we encounter a cell record for a cell beyond ours, or we encounter a record which is not a cell record.

If a ROW has no defined cells, we will set its dbRtcell offset to zero. If a ROW's cells are more than 64K from the previous ROW's cells (which is rare but possible), we will write out a zero offset for that ROW and ALL subsequent ROW records in the same block. All this means is that we have to search a little harder for the correct cell record: instead of being able to start our search at cells in the desired row, we will have to start searching at cells in some previous row.

Excel Formulas

This section describes how Excel stores formulas within BIFF files. Formulas appear in FORMULA, ARRAY, and NAME records.

In this section, the term "formula" is a synonym for "parsed expression"; it is the internal tokenized representation of an Excel formula.

Formulas are stored in a reverse Polish scheme. A formula consists of a sequence of parse tokens, each of which is either an operand, operator, or a control token. Operand tokens provide values; operator tokens perform arithmetic operations upon the operands; and control tokens assist in formula evaluation by describing properties of the formula.

A token consists of two parts: a token type and a token value. Token types are called "ptg's" in Excel; they are one byte long, ranging in value from 1 to 0x7F. Ptg's above 0x7F are reserved for internal Excel use.

The ptg specifies only what kind of information is contained in a token. The information itself is stored in the token value, immediately following the ptg in the parsed expression. Some tokens consist only of a ptg, without an accompanying token value; for example, to specify an addition operation, only the token type, ptgAdd, is required. But to specify an integer operand, both the ptg, ptgInt, and the token value, an integer, must be specified.

As an illustration, consider the parsed expression for =5+6. This parsed expression consists of three tokens: two integer operands and an operator.

```
 ptgInt    0x0005    ptgInt    0x0006   ptgAdd
 <   token 1  >      <  token 2  >  <token 3>
```

Notice that each ptgInt is immediately followed by the integer token value.

In many cases, the token value consists of a structure of two or more fields. In describing structures for these cases, offset zero is assumed to be the first byte of the token value, i.e. the first byte immediately following the token type.

Unless otherwise noted, all tokens can occur in FORMULA, ARRAY, and NAME records. Some tokens do not appear in one or more of these record types; they are explained as encountered.

Expression Evaluation

The evaluation of Excel formulas is a straightforward process. One LIFO stack, the operand stack, is maintained during evaluation. When an operand is encountered, it is pushed onto the stack. When an operator is encountered, it operates on the topmost operand or operands. Operator precedence is irrelevant at evaluation time; operators are handled as soon as they are encountered.

There are three kinds of operators: unary, binary, and function. Unary operators, like the minus sign which negates a number, operate only on the topmost operand. Binary operators, like the addition operator, operate on the top two operands. Function operators, which implement Excel functions, operate on a variable number of operands, depending on how many arguments the function accepts.

All operators work by popping the required operands from the stack, performing calculations, and pushing the result back onto the operand stack.

Unary Operators

Here are the unary operator tokens. All of these operators pop the top argument from the operand stack, perform a calculation, and push the result back onto the operand stack.

ptgUplus - unary plus (ptg = 0x12)
This operator has no effect.

ptgUminus - unary minus (ptg = 0x13)
Negates the top operand.

ptgPercent - percent sign (ptg = 0x14)
Divides the top operand by 100

Binary Operators

Here are the binary operator ptg's. All of these operators pop the top two arguments from the operand stack, perform a calculation, and push the result back onto the operand stack.

ptgAdd - addition (ptg = 0x03)
Adds the top two operands together.

ptgSub - subtraction (ptg = 0x04)
Subtracts the top operand from the second-to-top.

ptgMul - multiplication (ptg = 0x05)
Multiplies the top two operands.

ptgDiv - division (ptg = 0x06)
Divides the top operand by the second-to-top.

ptgPower - exponentiation (ptg = 0x07)
Raises the second-to-top operand to the power of the top operand.

ptgConcat - concatenation (ptg = 0x08)
Appends the top operand to the second-to-top operand.

ptgLT - less than (ptg = 0x09)
Evaluates to TRUE if the second-to-top operand is less than the top operand; FALSE otherwise.

ptgLE - less than or equal (ptg = 0x0A)
Evaluates to TRUE if the second-to-top operand is less than or equal to the top operand; FALSE otherwise.

ptgEQ - equal (ptg = 0x0B)
Evaluates to TRUE if the top two operands are equal; FALSE otherwise.

ptgGE - greater than or equal (ptg = 0x0C)
Evaluates to TRUE if the second-to-top operand is greater than or equal to the top operand; FALSE otherwise.

ptgGT - greater than (ptg = 0x0D)
Evaluates to TRUE if the second-to-top operand is greater than the top operand; FALSE otherwise.

ptgNE - not equal (ptg = 0x0E)
Evaluates to TRUE if the top two operands are not equal; FALSE otherwise.

ptgIsect - intersection (ptg = 0x0F)
This is the Excel space operator. It computes the intersection of the top two operands.

ptgUnion - union (ptg = 0x10)
This is the Excel comma operator. It computes the union of the top two operands.

ptgRange - range (ptg = 0x11)
This is the Excel colon operator. It computes the minimal bounding rectangle of the top two operands.

Operand Tokens - Constant

The following operand tokens push a single constant operand onto the operand stack.

ptgMissArg - missing argument (operand, ptg = 0x16)
Missing argument to an Excel function. For example, the second argument to DCOUNT(Database,,Criteria) would be stored as a ptgMissArg.

ptgStr - string constant (operand, ptg = 0x17)
String constant. Followed by the string.

Offset	Name	Size	Contents
0	cch	1	length of the string
1	rgch	var	the string

ptgStr requires special handling when parsed expressions are scanned. See the section "Scanning a Parsed Expression" for an explanation.

ptgErr - error value (operand, ptg = 0x1C)
Error constant. Followed by the error value. See the BOOLERR record for a list of error values.

Offset	Name	Size	Contents
0	err	1	Excel error value

ptgBool - boolean (operand, ptg = 0x1D)
Boolean constant. Followed by a byte value.

Offset	Name	Size	Contents
0	f	1	=1 for TRUE

=0 for FALSE

ptgInt - integer (operand, ptg = 0x1E)
Integer constant. Followed by a word value.

Offset	Name	Size	Contents
0	w	2	unsigned integer value

ptgNum - number (operand, ptg = 0x1F)
Numeric constant. Followed by an 8-byte IEEE floating point number.

Offset	Name	Size	Contents
0	num	8	IEEE floating point number

Operand Tokens - Classes

As described above, operand tokens push operand values onto the operand stack. These values are divided into three different classes, depending on what type of value the formula expects from the operand. The type of value is determined at parse time by the context of the operand.

REFERENCE CLASS. Some operands are required by context to be references. In this case, the term "reference" is a general term meaning the specification of one or more areas on an Excel document, without regard for the underlying cell values in those areas. When the Excel expression evaluator encounters a reference type operand, it pushes only the reference itself onto the operand stack; it does not dereference it to find the underlying cell values.

For example, consider the formula CELL("width",B5), which returns the column width of cell B5. Clearly, only the reference to cell B5 is important here; the value stored at cell B5 is irrelevant to the cell's width.

VALUE CLASS. This is the most common type of operand; it pushes a single dereferenced value onto the operand stack.

For example, consider the formula A1+1. Here, we are interested in the value stored in cell A1, so we dereference the A1 reference.

ARRAY CLASS. This operand pushes an array of values onto the operand stack. The values may be specified either in an array constant or in a reference to cells.

For example, consider the formula SUM({1,2,3;4,5,6}). Here, to evaluate the SUM function, the expression evaluator must push an entire array of values onto the operand stack.

The three classes of operand tokens are numerically divided as follows:

Operand Class	Ptg's
Reference	0x20 - 0x3F
Value	0x40 - 0x5F
Array	0x60 - 0x7F

Notice that the numerical difference between ptg classes is 0x20. This is the basis for forming the class variants of ptg's. Class variants of ptg's are formed from the reference class ptg, also known as the "base" ptg. To form the value class ptg from the base ptg, you add 0x20 to the ptg and append "V" (for "value") to the ptg name. To form the array class ptg from the base ptg, you add 0x40 to the ptg and append "A" (for "array") to the ptg name. These rules are summarized below for a hypothetical ptg called ptgFoo:

Class	Name	Ptg
Reference	ptgFoo	ptgFoo
Value	ptgFooV	ptgFoo + 0x20
Array	ptgFooA	ptgFoo + 0x40

For example, the base ptg which specifies a cell reference is ptgRef, which is equal to 0x24. Thus the reference class ptg is ptgRef, which is 0x24; the value class ptg is ptgRefV, which is 0x44; and the array class ptg is ptgRefA, which is 0x64.

Here is a useful method for computing the base ptg from any class variant:

```
 if (ptg & 0x40)
    {
    /* We have a value class ptg.  We need to set the
       0x20 bit to make it reference class, then strip
       off the high order bits. */
    ptgBase = (ptg | 0x20) & 0x3F;
    }
 else
    {
    /* We have a reference or array class ptg.  The 0x20
       bit is already set, so we just have to strip off
       the high order bits. */
    ptgBase = ptg & 0x3F;
    }
```

A more efficient implementation in C is to define a macro which computes the base ptg:

```
#define PtgBase(ptg) (((ptg & 0x40) ? (ptg | 0x20) : ptg) & 0x3F)
```

This macro is safe, i.e. it can be used on any ptg without damage.

Operand Tokens - Base

This section lists the operand tokens in their base form (also known as reference class).

ptgArray - array constant (operand, ptg = 0x20)

Array constant. Followed by six bytes.

Offset	Name	Size	Contents
0		6	RESERVED

The token value for ptgArray consists of the array dimensions and the array values. ptgArray differs from most other operand tokens in that the token value does not follow the token type. Instead, the token value is appended to the saved parsed expression, immediately following the last token. The format of the token value is as follows:

Offset	Name	Size	Contents
0	ccol	1	number of columns in the array constant
1	crw	2	number of rows in the array constant
3	rgval	var	the array values

256-column arrays are stored with a ccol of zero, since the true number of columns does not fit into a byte field. This is acceptable since there are no zero-column array constants.

The number of values in the array constant is equal to the product of the array dimensions, crw*ccol. Each value is either an 8-byte IEEE floating point number, or a string. The two formats of these values are as follows:

Offset	Name	Size	Contents
0	grbit	1	=0x01 for a number
1	num	8	IEEE floating point number

Offset	Name	Size	Contents
0	grbit	1	=0x02 for a string
1	cch	1	length of the string
2	rgch	var	the string

If a formula contains more than one array constant, then the token values for the array constants are appended to the saved parsed expression in order: first, the values for the first array constant, then the values for the second, and so on.

If a formula contains very long array constants, then the BIFF record containing the parsed expression may overflow into CONTINUE records to accomodate all of the array values. An individual array value is never split between records; record boundaries occur between successive array values.

In practice, the reference class ptgArray never appears in an Excel formula; only the value and array classes are used.

ptgName - name (operand, ptg = 0x23)

This ptg specifies the usage of an Excel name. The token value specifies which name is referenced.

Offset	Name	Size	Contents
0	ilbl	2	index of the referenced name
2		5	RESERVED - must be zeros

For local (i.e. non-external) name references, the ilbl field specifies a 1-based index into the document's own name table. The order of this name table is the order of NAME records in the BIFF file.

For external name references, the ilbl field specifies a 1-based index into the table of externally referenced names defined on the supporting document. The order of this name table is the order of EXTERNNAME records which are associated with the supporting document.

ptgRef - cell reference (operand, ptg = 0x24)

This ptg specifies a reference to a single cell. It is followed by the row and column of the reference. The row is encoded as bit fields.

Offset	Name	Size	Contents
0	grbitRw	2	row bit fields (see below)
2	col	1	column of the reference

Only the low 14 bits of the grbitRw field store the row number of the reference. The high 2 bits specify whether the row or column portion of the reference is a relative reference. Here is the bit field structure of the grbitRw field:

Bits	Mask	Name	Contents
15	0x8000	fRwRel	=1 if the row portion of the reference is relative =0 otherwise
14	0x4000	fColRel	=1 if the column portion of the reference is relative =0 otherwise
13-0	0x3FFF	rw	the row number of the reference

For example, cell C5 is row number 4, column number 2 (since

Excel stores cell references zero-based). Therefore the absolute reference $C$5 is stored as

```
ptgRef    0x0004    0x02.
```

The relative reference C5 is stored as

```
ptgRef    0xC004    0x02.
```

The mixed reference $C5 (absolute row, relative column) is stored as

```
ptgRef    0x4004    0x02.
```

ptgArea - area reference (operand, ptg = 0x25)

This ptg specifies a reference to a rectangle of cells. It is followed by the first row of the rectangle, last row, first column, and last column. Both the first row and last row are stored as bit fields.

Offset	Name	Size	Contents
0	grbitRwFirst	2	first row bit fields (see below)
2	grbitRwLast	2	last row bit fields (see below)
4	colFirst	1	first column of the reference
5	colLast	1	last column of the reference

The high order 2 bits of grbitRwFirst specify whether the first row or first column are relative references. The high order 2 bits of grbitRwLast specify whether the last row or last column are relative references. See the ptgRef token for a fuller explanation of these bit fields.

For example, consider references to the area C5:D8. C5 is row 4, column 2; D8 is row 7, column 3 (since Excel stores cell references zero-based). Therefore the absolute reference $C$5:$D$8 is stored as

```
ptgArea    0x0004    0x0007    0x02    0x03.
```

The relative reference C5:D8 is stored as

```
ptgArea    0xC004    0xC007    0x02    0x03.
```

The mixed reference C$5:$D8 (absolute first row, relative first column, relative last row, absolute last column) is stored as

```
ptgArea    0x4004    0x8007    0x02    0x03.
```

ptgMemArea - constant reference subexpression (operand, ptg = 0x26)

This ptg is used to optimize reference expressions. A reference expression consists of operands, usually references to cells or areas, joined by reference operators (intersection, union, and range). Here are three examples of reference expressions:

- A1,C3,D3:D5. This evaluates to two single cells and a 3x1 area.
- (A1:C3) (B2:D4). This evaluates to a 2x2 area.
- Name:C3. This evaluates to the smallest area which contains both C3 and all the cells referenced in "Name".

Many reference expressions evaluate to constant references. In the examples above, the first two expressions always evaluate to the same reference. The third example does not evaluate to a constant reference, since it depends on the name "Name", and Name's definition might change, which would cause the reference expression to evaluate differently.

When a reference expression does evaluate to a constant reference, Excel stores the constant reference in the parsed formula through a ptgMemArea token. This saves time during expression evaluation, since part of the expression being evaluated will have been pre-evaluated. This part of the expression is known as a reference subexpression.

ptgMemArea only occurs in FORMULA and ARRAY records, never in NAME records.

The token value for ptgMemArea consists of two parts: the length of the reference subexpression, and the value of the reference subexpression. The length is stored immediately following the ptgMemArea, but the value is appended to the saved parsed expression, immediately following the last token.

The format of the length is as follows:

Offset	Name	Size	Contents
0		3	RESERVED
3	cce	1	length of the reference subexpression

Immediately following this part of the token value is the reference subexpression itself, whose length is given by the cce field.

The rest of the token value, i.e. the value of the reference subexpression, is appended to the parsed expression in the following format:

Offset	Name	Size	Contents
0	cref	2	number of rectangles to follow
2	rgref	var	array of rectangles

Each rectangle is six bytes long and contains the following fields:

Offset	Name	Size	Contents

0	rwFirst	2	first row
2	rwLast	2	last row
4	colFirst	1	first column
5	colLast	1	last column

If a formula contains more than one ptgMemArea, then the token values are appended to the saved parsed expression in order: first, the values for the first ptgMemArea, then the values for the second, and so on.

If a formula contains very long reference expressions, then the BIFF record containing the parsed expression may overflow into CONTINUE records to accomodate all of the array values. An individual rectangle is never split between records; record boundaries occur between successive rectangles.

ptgMemErr - bad constant reference subexpression (operand, ptg = 0x27)

This ptg is closely related to ptgMemArea. It is used for pre-evaluating reference subexpressions which do not evaluate to references.

For example, consider the formula SUM(C:C 3:3). The argument to the SUM function is a valid reference subexpression, which generates a ptgMemArea for pre-evaluation. If the user deletes column C, then the formula adjusts to SUM(#REF! 3:3). In this case, the argument to SUM is still a constant reference subexpression, but it does not evaluate to a reference. Therefore a ptgMemErr is used for pre-evaluation.

The token value consists of the error value and length of the reference subexpression. Its format is as follows:

Offset	Name	Size	Contents
0		2	RESERVED
2	err	1	error value
3	cce	1	length of the reference subexpression

See the BOOLERR record for a list of error values.

ptgRefErr - deleted cell reference (operand, ptg = 0x2A)

This ptg specifies a cell reference that was adjusted to #REF! as a result of spreadsheet editing (e.g. cut and paste, delete). It is followed by three unused bytes.

Offset	Name	Size	Contents
0		3	RESERVED

The original base type of the adjusted ptg is always ptgRef or ptgRefN.

ptgAreaErr - deleted area reference (operand, ptg = 0x2C)

This ptg specifies an area reference that was adjusted to #REF! as a result of spreadsheet editing (e.g. cut and paste, delete). It is followed by six unused bytes.

Offset	Name	Size	Contents
0	6	RESERVED	

The original base type of the adjusted ptg is always ptgArea or ptgAreaN.

ptgRefN - cell reference within a name (operand, ptg = 0x2C)

This ptg only occurs in the parsed expression of a NAME record, never in a FORMULA or ARRAY record. It specifies a reference to a single cell. It is followed by the row and column of the reference. The row is encoded as bit fields.

The only difference between ptgRefN and ptgRef is the way relative references are stored. Relative references within name definitions are stored as offsets, not as row and column numbers. For example, if a name "Prev_cell" is defined to the relative reference =R[-2]C[-3] (assuming R1C1 mode), then the parsed expression for Prev_cell is

```
 ptgRefN    0xFFFE    0xFD.
```

The row offset, -2, in hexadecimal is 0xFFFE; the column offset, -3, is 0xFFFD. The row portion of the token value consists of the low 14 bits of the row offset, plus two high-order '1' bits to indicate that both the row and column portions are relative references. The column portion of the token value is simply the low byte of 0xFFFD.

If instead the name is =R[-2]C3, i.e. with an absolute column reference, then the parsed expression is

```
 ptgRefN    0xBFFE    0x02.
```

ptgAreaN - area reference within a name (operand, ptg = 0x2D)

This ptg only occurs in the parsed expression of a NAME record, never in a FORMULA or ARRAY record. It specifies a reference to a rectangle of cells. It is followed by the first row of the rectangle, last row, first column, and last column. Both the first row and last row are stored as bit fields.

The only difference between ptgAreaN and ptgArea is the way relative references are stored. See ptgRefN for an explanation of this.

Control Tokens
----

ptgExp - array formula (ptg = 0x01)

This ptg indicates an array formula. It only occurs in a FORMULA record, never in an ARRAY or NAME record. When ptgExp occurs in a formula, it is the only token in the formula. This indicates that the cell containing the formula is part of an array; the array formula is found in an ARRAY record.

The token value for ptgExp consists of the row and column of the upper left corner of the array formula.

Offset	Name	Size	Contents
0	rwFirst	2	row number of upper left corner
2	colFirst	1	column number of upper left corner

ptgTbl - data table (ptg = 0x02)

This ptg indicates a data table. It only occurs in a FORMULA record, never in an ARRAY or NAME record. When ptgTbl occurs in a formula, it is the only token in the formula. This indicates that the cell containing the formula is an interior cell in a data table; the table description is found in a TABLE record. Rows and columns which contain input values to be substituted in the table do not contain ptgTbl.

The token value for ptgTbl consists of the row and column of the upper left corner of the table's interior.

Offset	Name	Size	Contents
0	rwFirst	2	row number of upper left corner
2	colFirst	1	column number of upper left corner

ptgParen - parenthesis (ptg = 0x15)

This ptg is used only in unparsing a parsed expression, not in evaluating it. It indicates that the previous token in the parsed expression should be parenthesized. If the previous token is an operand, then only that operand is parenthesized. If the previous token is an operator, then the operator and all of its operands are parenthesized.

For example, the formula 1+(2) is stored as

```
 ptgInt   0x0001   ptgInt   0x0002   ptgParen   ptgAdd,
```

and only the operand 2 is parenthesized. But the formula (1+2) is stored as

```
 ptgInt   0x0001   ptgInt   0x0002   ptgAdd   ptgParen,
```

so the parenthesized quantity consists of the ptgAdd operator and both of its operands.

ptgAttr - special attribute (ptg = 0x19)

This ptg is used for a variety of purposes. In all cases, the token value consists of a byte of flags and a byte dependent on the flags.

Offset	Name	Size	Contents
0	grbit	1	bit flags
1	b	1	data

The grbit field contains the following flags:

Bit	Mask	Name	Contents
0	0x01	bitFAttrSemi	=1 if the formula contains a "volatile" function
1	0x02	bitFAttrIf	=1 to implement an optimized IF function
2	0x04	bitFAttrChoose	=1 to implement an optimized CHOOSE function
3	0x08	bitFAttrGoto	=1 to jump to another location within the parsed expression
4	0x10	bitFAttrSum	=1 to implement an optimized SUM function
5	0x20	bitFAttrBaxcel	=1 if the formula is a BASIC-style assignment statement

bitFAttrSemi is set to 1 if the formula contains a volatile function, i.e. a function which is calculated in every recalculation. The volatile functions in Excel are:

INDEX
RAND
NOW
AREAS
ROWS
COLUMNS
CELL
INDIRECT

If ptgAttr is used to indicate a volatile function, then it must be the first token in the parsed expression. The b field is unused.

bitFAttrGoto instructs the expression evaluator to skip part of the parsed expression during evaluation. The b field specifies the number of bytes to skip, minus one.

bitFAttrIf indicates an optimized IF function. An IF function contains 3 parts: a condition, a TRUE subexpression, and a FALSE subexpression. The syntax of an associated Excel formula would be IF(condition, TRUE subexpression, FALSE subexpression).

bitFAttrIf immediately follows the condition portion of the parsed expression. The b field specifies the offset to the FALSE subexpression; the TRUE subexpression is found immediately following the ptgAttr token.

At the end of the TRUE subexpression, there is a bitFAttrGoto token which causes a jump to beyond the FALSE subexpression. In this way, Excel only evaluates the correct subexpression, instead of evaluating both of them and discarding the wrong one.

The FALSE subexpression is optional in Excel. If it is missing, then the b field of the bitFAttrIf token specifies an offset to beyond the TRUE subexpression.

bitFAttrChoose indicates an optimized CHOOSE function. The b field specifies the number of cases in the CHOOSE function, and is followed by a sequence of byte offsets to those cases. The number of byte offsets in the sequence is one more than the number of cases in the CHOOSE function. Here is the format of this complex token value:

Offset	Name	Size	Contents
0	grbit	1	bitFAttrChoose
1	cCases	1	the number of cases in the CHOOSE function
2	rgb	var	a sequence of byte offsets to the CHOOSE cases. The number of bytes in this field is equal to the cCases field, plus one.

bitFAttrChoose requires special handling when parsed expressions are scanned. See the section "Scanning a Parsed Expression" for an explanation.

bitFAttrSum indicates an optimized SUM function. This is only used to optimize SUM functions with a single argument. The b field is unused.

ptgSheet - external reference (ptg = 0x1A)

This ptg indicates the start of an external reference. The token value indicates which sheet is being externally referenced. When this token is encountered during evaluation, it indicates that any following references to cells or names are external references, not local references, until the matching ptgEndSheet token is encountered.

Offset	Name	Size	Contents
0		4	RESERVED
4	ixals	2	index of the supporting sheet
6		1	RESERVED - must be zero

The ixals field specifies a 1-based index into the table of externally referenced documents. The order of this table is the order of EXTERNSHEET records.

ptgEndSheet - end external reference (ptg = 0x1B)

This ptg indicates the end of an external reference. It is followed by three bytes.

Offset	Name	Size	Contents
0		3	RESERVED

ptgMemNoMem - incomplete constant reference subexpression (ptg = 0x28)

This ptg is closely related to ptgMemArea. It is used to indicate a constant reference subexpression which could not be pre-evaluated because of low memory conditions. It only occurs in FORMULA and ARRAY records, never in NAME records.

The token value consists of the length of the reference subexpression.

Offset	Name	Size	Contents
0		3	RESERVED
3	cce	1	length of the reference subexpression

ptgMemFunc - variable reference subexpression (ptg = 0x29)

This ptg indicates a reference subexpression which does not evaluate to a constant reference. Any reference subexpression containing one or more of the following will generate a ptgMemFunc:

- an Excel function
- a usage of a name
- an external reference.

Here are examples of the three kinds of ptgMemFunc's:

- INDEX(ref,row_num,column_num,area_num):C3
- Name:$B$2
- SALES.XLS!$A$1:SALES.XLS!$C$3

The token value consists of the length of the reference subexpression.

Offset	Name	Size	Contents
0	cce	1	length of the reference subexpression

ptgMemAreaN - reference subexpression within a name (ptg = 0x2E)

This ptg only occurs in the parsed expression of a NAME record, never in a FORMULA or ARRAY record. It indicates a constant reference subexpression within a name definition. Unlike ptgMemArea, ptgMemAreaN is not used to pre-evaluate the reference subexpression.

The token value consists of the length of the reference subexpression.

Offset	Name	Size	Contents
0	cce	1	length of the reference subexpression

ptgMemNoMemN - incomplete reference subexpression within a name (control, ptg = 0x2F)

This ptg is closely related to ptgMemAreaN. It is used to indicate a constant reference subexpression within a name which could not be evaluated because of low memory conditions. It only occurs in NAME records, never in FORMULA or ARRAY records.

The token value consists of the length of the reference subexpression.

Offset	Name	Size	Contents
0	cce	1	length of the reference subexpression

Function Operators

Here are the function operator ptg's. All of these operators pop arguments from the operand stack, compute a function, and push the result back onto the operand stack. The number of operands popped from the stack is equal to the number of arguments passed to the Excel function. Some Excel functions always require a fixed number of arguments, while others may accept a variable number of arguments. The SUM function, for example, accepts from 1 to 14 arguments.

Although they are operators, function tokens also behave like operands in that they can occur in any of the three ptg classes (reference, value, and array).

ptgFunc - Excel function (operator, ptg = 0x21)

Indicates an Excel function with a fixed number of arguments. Followed by the index of the function within the function table. See the section "Excel Function Table" for a list of Excel functions.

Offset	Name	Size	Contents
0	iftab	1	index of the function

ptgFuncVar - Excel function (operator, ptg = 0x22)

Indicates an Excel function with a variable number of arguments. Followed by the number of arguments and the index of the function within the function table. See the section "Excel Function Table" for a list of Excel functions.

Offset	Name	Size	Contents
0	carg	1	number of arguments to the function
1	iftab	1	index of the function

ptgFuncCE - command-equivalent function (operator, ptg = 0x38)

Indicates an Excel command-equivalent function. Followed by the number of arguments and the index of the function within the command-equivalent function table. See the section "Command Equivalent Function Table" for a list of Excel command-equivalent functions.

Offset	Name	Size	Contents
0	carg	1	number of arguments to the function
1	icetab	1	index of the function

Reserved Ptg's

All ptg's between 0 and 0xFF not explicitly mentioned in this document are reserved by Microsoft for future use.

Scanning a Parsed Expression

One fairly common operation on parsed expressions is to scan them, taking appropriate actions at each ptg. This is accomplished with a loop using a pointer variable, which points to the next ptg to scan. However, this pointer must be incremented carefully, since different ptg's may have token values of different lengths.

One good solution to this problem is to maintain a array, with one element per ptg, containing the size of the token value. To increment the pointer, you simply add the array element corresponding to the current ptg. A possible space optimization here is to limit the array indices to the range 0 - 0x3F, and then index it using the base ptg instead of the fully classed ptg. This works because the token value is the same for all classes of a particular ptg.

There are two tokens which are variable length, and so do not fit this framework. These tokens must be handled as special cases in any formula scanning loop.

The first exception is ptgStr, which is followed by a variable length string. The token value specifies the length of the string, so the pointer is incrementing by fetching and adding the string length from the token value.

The other exception is the bitFAttrChoose token of ptgAttr. The token value contains a variable number of bytes in sequence. The number of bytes in the sequence is specified in the token value, so the proper method of incrementing is to fetch and add the sequence length.

Here is sample C code which scans a parsed expression:

```
Scan(rgb, cb)
char rgb[];   /* The parsed expression */
int cb;   /* The length of the parsed expression */
  {
  char *pb;   /* Pointer to the current token */
  char *pbMac; /* Pointer to the end of the p.e. */
  int ptg; /* Raw ptg */
  int ptgBase; /* Base ptg */
  extern char token_size[];   /* Array of token value sizes */

#define bitFAttrChoose 0x04   /* CHOOSE type of ptgAttr */

  pb = rgb;
  pbMac = &rgb[cb];
  while (pb < pbMac)
   {
   /* Fetch the next token and determine its base type.
      Note that the postincrement conveniently leaves pb
      pointing to the token value. */
   ptg = *pb++;
   ptgBase = PtgBase(ptg);
   switch (ptgBase)
      {
      ...
   case ptgAttr:
      /* Check for a CHOOSE ptgAttr and skip over the
         table of offsets if found. */
      if (*pb & bitFAttrChoose)
       pb += *(pb + 1) + 1;
      break;
   case ptgStr:
      /* String constant.  Skip over the size byte and
         the string itself. */
      pb += *pb + 1;
      break;
   default:
      /* Look up the token value size and add it to the
         pointer.  The token_size array is only indexed
         by the base ptg as a space optimization, since
         the token sizes of the value and array classes
         are the same as the base class'. */
      pb += token_size[ptgBase];
      break;
      ...
      }
   }
  }
```

Excel Function Table

Here is a list of Excel functions sorted by index. The Excel name for a function index is "iftab". iftab's appear in ptgFunc and ptgFuncVar tokens. Unused iftab's are reserved for future use.

Function	iftab
COUNT	0x00
IF	0x01
ISNA	0x02
ISERROR	0x03

SUM	0x04
AVERAGE	0x05
MIN	0x06
MAX	0x07
ROW	0x08
COLUMN	0x09
NA	0x0A
NPV	0x0B
STDEV	0x0C
DOLLAR	0x0D
FIXED	0x0E
SIN	0x0F
COS	0x10
TAN	0x11
ATAN	0x12
PI	0x13
SQRT	0x14
EXP	0x15
LN	0x16
LOG10	0x17
ABS	0x18
INT	0x19
SIGN	0x1A
ROUND	0x1B
LOOKUP	0x1C
INDEX	0x1D
REPT	0x1E
MID	0x1F
LEN	0x20
VALUE	0x21
TRUE	0x22
FALSE	0x23
AND	0x24
OR	0x25
NOT	0x26
MOD	0x27
DCOUNT	0x28
DSUM	0x29
DAVERAGE	0x2A
DMIN	0x2B
DMAX	0x2C
DSTDEV	0x2D
VAR	0x2E
DVAR	0x2F
TEXT	0x30
LINEST	0x31
TREND	0x32
LOGEST	0x33
GROWTH	0x34
GOTO	0x35
HALT	0x36
RETURN	0x37

PV	0x38
FV	0x39
NPER	0x3A
PMT	0x3B
RATE	0x3C
MIRR	0x3D
IRR	0x3E
RAND	0x3F
MATCH	0x40
DATE	0x41
TIME	0x42
DAY	0x43
MONTH	0x44
YEAR	0x45
WEEKDAY	0x46
HOUR	0x47
MINUTE	0x48
SECOND	0x49
NOW	0x4A
AREAS	0x4B
ROWS	0x4C
COLUMNS	0x4D
OFFSET	0x4E
ABSREF	0x4F
RELREF	0x50
ARGUMENT	0x51
SEARCH	0x52
TRANSPOSE	0x53
ERROR	0x54
STEP	0x55
TYPE	0x56
ECHO	0x57
SET.NAME	0x58
CALLER	0x59
DEREF	0x5A
WINDOWS	0x5B
SERIES	0x5C
DOCUMENTS	0x5D
ACTIVE.CELL	0x5E
SELECTION	0x5F
RESULT	0x60
ATAN2	0x61
ASIN	0x62
ACOS	0x63
CHOOSE	0x64
HLOOKUP	0x65
VLOOKUP	0x66
LINKS	0x67
INPUT	0x68
ISREF	0x69
GET.FORMULA	0x6A
GET.NAME	0x6B

SET.VALUE	0x6C
LOG	0x6D
EXEC	0x6E
CHAR	0x6F
LOWER	0x70
UPPER	0x71
PROPER	0x72
LEFT	0x73
RIGHT	0x74
EXACT	0x75
TRIM	0x76
REPLACE	0x77
SUBSTITUTE	0x78
CODE	0x79
NAMES	0x7A
DIRECTORY	0x7B
FIND	0x7C
CELL	0x7D
ISERR	0x7E
ISTEXT	0x7F
ISNUMBER	0x80
ISBLANK	0x81
T	0x82
N	0x83
FOPEN	0x84
FCLOSE	0x85
FSIZE	0x86
FREADLN	0x87
FREAD	0x88
FWRITELN	0x89
FWRITE	0x8A
FPOS	0x8B
DATEVALUE	0x8C
TIMEVALUE	0x8D
SLN	0x8E
SYD	0x8F
DDB	0x90
GET.DEF	0x91
REFTEXT	0x92
TEXTREF	0x93
INDIRECT	0x94
REGISTER	0x95
CALL	0x96
ADD.BAR	0x97
ADD.MENU	0x98
ADD.COMMAND	0x99
ENABLE.COMMAND	0x9A
CHECK.COMMAND	0x9B
RENAME.COMMAND	0x9C
SHOW.BAR	0x9D
DELETE.MENU	0x9E
DELETE.COMMAND	0x9F

GET.CHART.ITEM	0xA0
DIALOG.BOX	0xA1
CLEAN	0xA2
MDETERM	0xA3
MINVERSE	0xA4
MMULT	0xA5
FILES	0xA6
IPMT	0xA7
PPMT	0xA8
COUNTA	0xA9
CANCEL.KEY	0xAA
FOR	0xAB
WHILE	0xAC
BREAK	0xAD
NEXT	0xAE
INITIATE	0xAF
REQUEST	0xB0
POKE	0xB1
EXECUTE	0xB2
TERMINATE	0xB3
RESTART	0xB4
HELP	0xB5
GET.BAR	0xB6
PRODUCT	0xB7
FACT	0xB8
GET.CELL	0xB9
GET.WORKSPACE	0xBA
GET.WINDOW	0xBB
GET.DOCUMENT	0xBC
DPRODUCT	0xBD
ISNONTEXT	0xBE
GET.NOTE	0xBF
NOTE	0xC0
STDEVP	0xC1
VARP	0xC2
DSTDEVP	0xC3
DVARP	0xC4
TRUNC	0xC5
ISLOGICAL	0xC6
DCOUNTA	0xC7
DELETE.BAR	0xC8

Here is a list of Excel functions sorted alphabetically by function:

Function	iftab
ABS	0x18
ABSREF	0x4F
ACOS	0x63
ACTIVE.CELL	0x5E
ADD.BAR	0x97

ADD.COMMAND	0x99
ADD.MENU	0x98
AND	0x24
AREAS	0x4B
ARGUMENT	0x51
ASIN	0x62
ATAN	0x12
ATAN2	0x61
AVERAGE	0x05
BREAK	0xAD
CALL	0x96
CALLER	0x59
CANCEL.KEY	0xAA
CELL	0x7D
CHAR	0x6F
CHECK.COMMAND	0x9B
CHOOSE	0x64
CLEAN	0xA2
CODE	0x79
COLUMN	0x09
COLUMNS	0x4D
COS	0x10
COUNT	0x00
COUNTA	0xA9
DATE	0x41
DATEVALUE	0x8C
DAVERAGE	0x2A
DAY	0x43
DCOUNT	0x28
DCOUNTA	0xC7
DDB	0x90
DELETE.BAR	0xC8
DELETE.COMMAND	0x9F
DELETE.MENU	0x9E
DEREF	0x5A
DIALOG.BOX	0xA1
DIRECTORY	0x7B
DMAX	0x2C
DMIN	0x2B
DOCUMENTS	0x5D
DOLLAR	0x0D
DPRODUCT	0xBD
DSTDEV	0x2D
DSTDEVP	0xC3
DSUM	0x29
DVAR	0x2F
DVARP	0xC4
ECHO	0x57
ENABLE.COMMAND	0x9A
ERROR	0x54
EXACT	0x75
EXEC	0x6E

EXECUTE	0xB2
EXP	0x15
FACT	0xB8
FALSE	0x23
FCLOSE	0x85
FILES	0xA6
FIND	0x7C
FIXED	0x0E
FOPEN	0x84
FOR	0xAB
FPOS	0x8B
FREAD	0x88
FREADLN	0x87
FSIZE	0x86
FV	0x39
FWRITE	0x8A
FWRITELN	0x89
GET.BAR	0xB6
GET.CELL	0xB9
GET.CHART.ITEM	0xA0
GET.DEF	0x91
GET.DOCUMENT	0xBC
GET.FORMULA	0x6A
GET.NAME	0x6B
GET.NOTE	0xBF
GET.WINDOW	0xBB
GET.WORKSPACE	0xBA
GOTO	0x35
GROWTH	0x34
HALT	0x36
HELP	0xB5
HLOOKUP	0x65
HOUR	0x47
IF	0x01
INDEX	0x1D
INDIRECT	0x94
INITIATE	0xAF
INPUT	0x68
INT	0x19
IPMT	0xA7
IRR	0x3E
ISBLANK	0x81
ISERR	0x7E
ISERROR	0x03
ISLOGICAL	0xC6
ISNA	0x02
ISNONTEXT	0xBE
ISNUMBER	0x80
ISREF	0x69
ISTEXT	0x7F
LEFT	0x73
LEN	0x20

LINEST	0x31
LINKS	0x67
LN	0x16
LOG	0x6D
LOG10	0x17
LOGEST	0x33
LOOKUP	0x1C
LOWER	0x70
MATCH	0x40
MAX	0x07
MDETERM	0xA3
MID	0x1F
MIN	0x06
MINUTE	0x48
MINVERSE	0xA4
MIRR	0x3D
MMULT	0xA5
MOD	0x27
MONTH	0x44
N	0x83
NA	0x0A
NAMES	0x7A
NEXT	0xAE
NOT	0x26
NOTE	0xC0
NOW	0x4A
NPER	0x3A
NPV	0x0B
OFFSET	0x4E
OR	0x25
PI	0x13
PMT	0x3B
POKE	0xB1
PPMT	0xA8
PRODUCT	0xB7
PROPER	0x72
PV	0x38
RAND	0x3F
RATE	0x3C
REFTEXT	0x92
REGISTER	0x95
RELREF	0x50
RENAME.COMMAND	0x9C
REPLACE	0x77
REPT	0x1E
REQUEST	0xB0
RESTART	0xB4
RESULT	0x60
RETURN	0x37
RIGHT	0x74
ROUND	0x1B
ROW	0x08

ROWS	0x4C
SEARCH	0x52
SECOND	0x49
SELECTION	0x5F
SERIES	0x5C
SET.NAME	0x58
SET.VALUE	0x6C
SHOW.BAR	0x9D
SIGN	0x1A
SIN	0x0F
SLN	0x8E
SQRT	0x14
STDEV	0x0C
STDEVP	0xC1
STEP	0x55
SUBSTITUTE	0x78
SUM	0x04
SYD	0x8F
T	0x82
TAN	0x11
TERMINATE	0xB3
TEXT	0x30
TEXTREF	0x93
TIME	0x42
TIMEVALUE	0x8D
TRANSPOSE	0x53
TREND	0x32
TRIM	0x76
TRUE	0x22
TRUNC	0xC5
TYPE	0x56
UPPER	0x71
VALUE	0x21
VAR	0x2E
VARP	0xC2
VLOOKUP	0x66
WEEKDAY	0x46
WHILE	0xAC
WINDOWS	0x5B
YEAR	0x45

Command Equivalent Function Table

Here is a list of command equivalent functions sorted by index. The Excel name for a comand equivalent function index is "icetab". icetab's appear in ptgFuncCE tokens. Unused icetab's are reserved for future use.

Command Equivalent	icetab
BEEP	0x00
OPEN	0x01

OPEN.LINKS	0x02
CLOSE.ALL	0x03
SAVE	0x04
SAVE.AS	0x05
FILE.DELETE	0x06
PAGE.SETUP	0x07
PRINT	0x08
PRINTER.SETUP	0x09
QUIT	0x0A
NEW.WINDOW	0x0B
ARRANGE.ALL	0x0C
SIZE	0x0D
MOVE	0x0E
FULL	0x0F
CLOSE	0x10
RUN	0x11
SET.PRINT.AREA	0x16
SET.PRINT.TITLES	0x17
SET.PAGE.BREAK	0x18
REMOVE.PAGE.BREAK	0x19
FONT	0x1A
DISPLAY	0x1B
PROTECT.DOCUMENT	0x1C
PRECISION	0x1D
A1.R1C1	0x1E
CALCULATE.NOW	0x1F
CALCULATION	0x20
DATA.FIND	0x22
EXTRACT	0x23
DATA.DELETE	0x24
SET.DATABASE	0x25
SET.CRITERIA	0x26
SORT	0x27
DATA.SERIES	0x28
TABLE	0x29
FORMAT.NUMBER	0x2A
ALIGNMENT	0x2B
STYLE	0x2C
BORDER	0x2D
CELL.PROTECTION	0x2E
COLUMN.WIDTH	0x2F
UNDO	0x30
CUT	0x31
COPY	0x32
PASTE	0x33
CLEAR	0x34
PASTE.SPECIAL	0x35
EDIT.DELETE	0x36
INSERT	0x37
FILL.RIGHT	0x38
FILL.DOWN	0x39
DEFINE.NAME	0x3D

CREATE.NAMES	0x3E
FORMULA.GOTO	0x3F
FORMULA.FIND	0x40
SELECT.LAST.CELL	0x41
SHOW.ACTIVE.CELL	0x42
GALLERY.AREA	0x43
GALLERY.BAR	0x44
GALLERY.COLUMN	0x45
GALLERY.LINE	0x46
GALLERY.PIE	0x47
GALLERY.SCATTER	0x48
COMBINATION	0x49
PREFERRED	0x4A
ADD.OVERLAY	0x4B
GRIDLINES	0x4C
SET.PREFERRED	0x4D
AXES	0x4E
LEGEND	0x4F
ATTACH.TEXT	0x50
ADD.ARROW	0x51
SELECT.CHART	0x52
SELECT.PLOT.AREA	0x53
PATTERNS	0x54
MAIN.CHART	0x55
OVERLAY	0x56
SCALE	0x57
FORMAT.LEGEND	0x58
FORMAT.TEXT	0x59
PARSE	0x5B
JUSTIFY	0x5C
HIDE	0x5D
UNHIDE	0x5E
WORKSPACE	0x5F
FORMULA	0x60
FORMULA.FILL	0x61
FORMULA.ARRAY	0x62
DATA.FIND.NEXT	0x63
DATA.FIND.PREV	0x64
FORMULA.FIND.NEXT	0x65
FORMULA.FIND.PREV	0x66
ACTIVATE	0x67
ACTIVATE.NEXT	0x68
ACTIVATE.PREV	0x69
UNLOCKED.NEXT	0x6A
UNLOCKED.PREV	0x6B
COPY.PICTURE	0x6C
SELECT	0x6D
DELETE.NAME	0x6E
DELETE.FORMAT	0x6F
VLINE	0x70
HLINE	0x71
VPAGE	0x72

HPAGE	0x73
VSCROLL	0x74
HSCROLL	0x75
ALERT	0x76
NEW	0x77
CANCEL.COPY	0x78
SHOW.CLIPBOARD	0x79
MESSAGE	0x7A
PASTE.LINK	0x7C
APP.ACTIVATE	0x7D
DELETE.ARROW	0x7E
ROW.HEIGHT	0x7F
FORMAT.MOVE	0x80
FORMAT.SIZE	0x81
FORMULA.REPLACE	0x82
SEND.KEYS	0x83
SELECT.SPECIAL	0x84
APPLY.NAMES	0x85
REPLACE.FONT	0x86
FREEZE.PANES	0x87
SHOW.INFO	0x88
SPLIT	0x89
ON.WINDOW	0x8A
ON.DATA	0x8B
DISABLE.INPUT	0x8C
LIST.NAMES	0x8F
FILE.CLOSE	0x90
SAVE.WORKSPACE	0x91
DATA.FORM	0x92
COPY.CHART	0x93
ON.TIME	0x94
WAIT	0x95
FORMAT.FONT	0x96
FILL.UP	0x97
FILL.LEFT	0x98
DELETE.OVERLAY	0x99
SHORT.MENUS	0x9B
CHANGE.LINK	0xA6
CALCULATE.DOCUMENT	0xA7
ON.KEY	0xA8
APP.RESTORE	0xA9
APP.MOVE	0xAA
APP.SIZE	0xAB
APP.MINIMIZE	0xAC
APP.MAXIMIZE	0xAD
MAIN.CHART.TYPE	0xB9
OVERLAY.CHART.TYPE	0xBA
SELECT.END	0xBB

Here is a list of command equivalent functions sorted alphabetically by function name.

Command Equivalent	icetab
A1.R1C1	0x1E
ACTIVATE	0x67
ACTIVATE.NEXT	0x68
ACTIVATE.PREV	0x69
ADD.ARROW	0x51
ADD.OVERLAY	0x4B
ALERT	0x76
ALIGNMENT	0x2B
APPLY.NAMES	0x85
APP.ACTIVATE	0x7D
APP.MAXIMIZE	0xAD
APP.MINIMIZE	0xAC
APP.MOVE	0xAA
APP.RESTORE	0xA9
APP.SIZE	0xAB
ARRANGE.ALL	0x0C
ATTACH.TEXT	0x50
AXES	0x4E
BEEP	0x00
BORDER	0x2D
CALCULATE.DOCUMENT	0xA7
CALCULATE.NOW	0x1F
CALCULATION	0x20
CANCEL.COPY	0x78
CELL.PROTECTION	0x2E
CHANGE.LINK	0xA6
CLEAR	0x34
CLOSE	0x10
CLOSE.ALL	0x03
COLUMN.WIDTH	0x2F
COMBINATION	0x49
COPY	0x32
COPY.CHART	0x93
COPY.PICTURE	0x6C
CREATE.NAMES	0x3E
CUT	0x31
DATA.DELETE	0x24
DATA.FIND	0x22
DATA.FIND.NEXT	0x63
DATA.FIND.PREV	0x64
DATA.FORM	0x92
DATA.SERIES	0x28
DEFINE.NAME	0x3D
DELETE.ARROW	0x7E
DELETE.FORMAT	0x6F
DELETE.NAME	0x6E
DELETE.OVERLAY	0x99
DISABLE.INPUT	0x8C
DISPLAY	0x1B
EDIT.DELETE	0x36

EXTRACT	0x23
FILE.CLOSE	0x90
FILE.DELETE	0x06
FILL.DOWN	0x39
FILL.LEFT	0x98
FILL.RIGHT	0x38
FILL.UP	0x97
FONT	0x1A
FORMAT.FONT	0x96
FORMAT.LEGEND	0x58
FORMAT.MOVE	0x80
FORMAT.NUMBER	0x2A
FORMAT.SIZE	0x81
FORMAT.TEXT	0x59
FORMULA	0x60
FORMULA.ARRAY	0x62
FORMULA.FILL	0x61
FORMULA.FIND	0x40
FORMULA.FIND.NEXT	0x65
FORMULA.FIND.PREV	0x66
FORMULA.GOTO	0x3F
FORMULA.REPLACE	0x82
FREEZE.PANES	0x87
FULL	0x0F
GALLERY.AREA	0x43
GALLERY.BAR	0x44
GALLERY.COLUMN	0x45
GALLERY.LINE	0x46
GALLERY.PIE	0x47
GALLERY.SCATTER	0x48
GRIDLINES	0x4C
HIDE	0x5D
HLINE	0x71
HPAGE	0x73
HSCROLL	0x75
INSERT	0x37
JUSTIFY	0x5C
LEGEND	0x4F
LIST.NAMES	0x8F
MAIN.CHART	0x55
MAIN.CHART.TYPE	0xB9
MESSAGE	0x7A
MOVE	0x0E
NEW	0x77
NEW.WINDOW	0x0B
ON.DATA	0x8B
ON.KEY	0xA8
ON.TIME	0x94
ON.WINDOW	0x8A
OPEN	0x01
OPEN.LINKS	0x02
OVERLAY	0x56

OVERLAY.CHART.TYPE	0xBA
PAGE.SETUP	0x07
PARSE	0x5B
PASTE	0x33
PASTE.LINK	0x7C
PASTE.SPECIAL	0x35
PATTERNS	0x54
PRECISION	0x1D
PREFERRED	0x4A
PRINT	0x08
PRINTER.SETUP	0x09
PROTECT.DOCUMENT	0x1C
QUIT	0x0A
REMOVE.PAGE.BREAK	0x19
REPLACE.FONT	0x86
ROW.HEIGHT	0x7F
RUN	0x11
SAVE	0x04
SAVE.AS	0x05
SAVE.WORKSPACE	0x91
SCALE	0x57
SELECT	0x6D
SELECT.CHART	0x52
SELECT.END	0xBB
SELECT.LAST.CELL	0x41
SELECT.PLOT.AREA	0x53
SELECT.SPECIAL	0x84
SEND.KEYS	0x83
SET.CRITERIA	0x26
SET.DATABASE	0x25
SET.PAGE.BREAK	0x18
SET.PREFERRED	0x4D
SET.PRINT.AREA	0x16
SET.PRINT.TITLES	0x17
SHORT.MENUS	0x9B
SHOW.ACTIVE.CELL	0x42
SHOW.CLIPBOARD	0x79
SHOW.INFO	0x88
SIZE	0x0D
SORT	0x27
SPLIT	0x89
STYLE	0x2C
TABLE	0x29
UNDO	0x30
UNHIDE	0x5E
UNLOCKED.NEXT	0x6A
UNLOCKED.PREV	0x6B
VLINE	0x70
VPAGE	0x72
VSCROLL	0x74
WAIT	0x95
WORKSPACE	0x5F

List of Ptg's

Here is a list of all ptg's that appear in BIFF files. All other ptg's are reserved for future use.

Name	Ptg	Type
ptgExp	0x01	control
ptgTbl	0x02	control
ptgAdd	0x03	operator
ptgSub	0x04	operator
ptgMul	0x05	operator
ptgDiv	0x06	operator
ptgPower	0x07	operator
ptgConcat	0x08	operator
ptgLT	0x09	operator
ptgLE	0x0A	operator
ptgEQ	0x0B	operator
ptgGE	0x0C	operator
ptgGT	0x0D	operator
ptgNE	0x0E	operator
ptgIsect	0x0F	operator
ptgUnion	0x10	operator
ptgRange	0x11	operator
ptgUplus	0x12	operator
ptgUminus	0x13	operator
ptgPercent	0x14	operator
ptgParen	0x15	control
ptgMissArg	0x16	operand
ptgStr	0x17	operand
ptgAttr	0x19	control
ptgSheet	0x1A	control
ptgEndSheet	0x1B	control
ptgErr	0x1C	operand
ptgBool	0x1D	operand
ptgInt	0x1E	operand
ptgNum	0x1F	operand
ptgArray	0x20	operand, reference class
ptgFunc	0x21	operator
ptgFuncVar	0x22	operator
ptgName	0x23	operand, reference class
ptgRef	0x24	operand, reference class
ptgArea	0x25	operand, reference class
ptgMemArea	0x26	operand, reference class
ptgMemErr	0x27	operand, reference class
ptgMemNoMem	0x28	control
ptgMemFunc	0x29	control
ptgRefErr	0x2A	operand, reference class
ptgAreaErr	0x2B	operand, reference class
ptgRefN	0x2C	operand, reference class
ptgAreaN	0x2D	operand, reference class

ptgMemAreaN	0x2E	control
ptgMemNoMemN	0x2F	control
ptgFuncCE	0x38	operator
ptgArrayV	0x40	operand, value class
ptgFuncV	0x41	operator
ptgFuncVarV	0x42	operator
ptgNameV	0x43	operand, value class
ptgRefV	0x44	operand, value class
ptgAreaV	0x45	operand, value class
ptgMemAreaV	0x46	operand, value class
ptgMemErrV	0x47	operand, value class
ptgMemNoMemV	0x48	control
ptgMemFuncV	0x49	control
ptgRefErrV	0x4A	operand, value class
ptgAreaErrV	0x4B	operand, value class
ptgRefNV	0x4C	operand, value class
ptgAreaNV	0x4D	operand, value class
ptgMemAreaNV	0x4E	control
ptgMemNoMemNV	0x4F	control
ptgFuncCEV	0x58	operator
ptgArrayA	0x60	operand, array class
ptgFuncA	0x61	operator
ptgFuncVarA	0x62	operator
ptgNameA	0x63	operand, array class
ptgRefA	0x64	operand, array class
ptgAreaA	0x65	operand, array class
ptgMemAreaA	0x66	operand, array class
ptgMemErrA	0x67	operand, array class
ptgMemNoMemA	0x68	control
ptgMemFuncA	0x69	control
ptgRefErrA	0x6A	operand, array class
ptgAreaErrA	0x6B	operand, array class
ptgRefNA	0x6C	operand, array class
ptgAreaNA	0x6D	operand, array class
ptgMemAreaNA	0x6E	control
ptgMemNoMemNA	0x6F	control
ptgFuncCEA	0x78	operator

Data writing object - (c) J.G. Ferreira, CIS 100326,1361.

This upload allows a program to save records from TPW (and BP7 with minor changes) to ascii space-delimited files, ascii tab-delimited files, and MS-EXCEL vers 2.1, 3, 4 and 5.

I use it to save data to file from a database in TPW and paradox engine, and results from mathematical modelling programs.

A sample app. BIFFALL.PAS is provided which creates a file called test.xls in the root directory of C:\ and saves different types of data to it. You may then use Excel to open the file (or notepad for ASCII files).

Parts of this are adapted from:

BIFFLib 1.00 object:
Object for reading and writing BIFF-files
Copyright (C) Marcus Hettlage 1993

uploaded to the Pascal forum on CIS, and MS-SDK data.
I found the object rather difficult to work with, so here a different approach is used.

Some of the BIFF code is translated from C from an example by Todd Lucas from Microsoft. The classes were built by me, but the approach for writing the record header separately and then the data is from him. It is however made much simpler by using objects.

If you want to further develop the classes to read excel files, save formatting info, etc. refer to the Excel SDK books from Microsoft Press or to Marcus Hettlage's object.
Feel free to use these files as is, extend them, change the code etc.

Let me know of problems/comments.

With thanks to all (on this forum an otherwise) for help in so many different ways, specially to Marcus Hettlage, Todd Lucas and Frank Plas.

----- WINSAVE.PAS -------

```
{$I-,N+}

{General purpose save library - (c) J. Gomes Ferreira 1994
 Writes records in Excel v.2.1, v.3, v.4,
 ASCII comma separated text,
 and tab-delimited excel text files

 Excel BIFF: parts translated from C by Todd Lucas - Microsoft corp.}
```

```
Unit WinSave;

Interface

uses strings, winfdlg, winprocs, wintypes;

Const

  Space : char = chr(32);
  Tab   : char = chr(9);
  CR    : char = chr(13);
  LF    : char = chr(10);

  {BOF}
  BOF       = $0009;
  BIT_BIFF5 = $0800;
  BIT_BIFF4 = $0400;
  BIT_BIFF3 = $0200;
  BOF_BIFF5 = BOF or BIT_BIFF5;
  BOF_BIFF4 = BOF or BIT_BIFF4;
  BOF_BIFF3 = BOF or BIT_BIFF3;
  {EOF}
  BIFF_EOF = $000a;
  {Dimensions}
  DIMENSIONS = $0000;
  DIMENSIONS_BIFF4 = DIMENSIONS or BIT_BIFF3;
  DIMENSIONS_BIFF3 = DIMENSIONS or BIT_BIFF3;
  {Document types}
  DOCTYPE_XLS = $0010;
  DOCTYPE_XLC = $0020;
  DOCTYPE_XLM = $0040;
  DOCTYPE_XLW = $0100;
  {Use with output functions}
  VER_BIFF4 = $04;
  VER_BIFF3 = $03;
  VER_BIFF2 = $02;
  {Structures}
  LEN_RECORDHEADER = 4;
  {Data types }
  CellBlank   = 1;
  CellInteger = 2;
  CellDouble  = 4;
  CellLabel   = 8;
  CellBoolean = 16; { or error }

Type

  string10 = String[10]; String255 = string[255];
  chartype = array[0..255] of char;

  PBaseSave = ^TBaseSave;
  TBaseSave = object
    Charfile : file of char;
    DataString : String255; Separator : char;
    MinSaveRecs, MaxSaveRecs, MinSaveCols, MaxSaveCols : word;
    CellType, Row, Col : integer;
    Data : pointer;
    EndOfLine : boolean;

    Constructor Init(SaveFileName : TFileName );
    procedure WriteBlank; virtual;
    procedure WriteInteger; virtual;
    procedure WriteDouble; virtual;
    procedure WriteLabel (var w : word); virtual;
    procedure WriteData(AType, ARow, ACol: Integer; AData: Pointer);
virtual;
    Destructor Done; virtual;
  end;

  PASCII = ^TASCII;
  TASCII = object(TBaseSave)
    Constructor Init( SaveFileName : TFileName );
    Destructor Done; virtual;
  end;

  PExcelTab = ^TExcelTab;
  TExcelTab = object(TBaseSave)
    Constructor Init(SaveFileName : TFileName );
    Destructor Done; virtual;
  end;

  PBIFF2 = ^TBIFF2;
  TBIFF2 = object(TBaseSave)
    {BIFFtime, BIFFdata : double;} BIFFColumn : byte;
    ExcelFile : File;
    VerBIFF, TypeDOC : word;
    typerec, lendata : word;

    constructor Init(AFileName : TFileName);
    destructor Done; virtual;
    procedure BIFFBOF; virtual;
    procedure BIFFDIM; virtual;
    procedure WriteBOF; virtual;
    procedure WriteRecordHeader; virtual;
    procedure WriteDimensions; virtual;
    procedure WriteEOF; virtual;
    procedure WriteData(AType, ARow, ACol: Integer; AData: Pointer);
virtual;
    procedure WriteBlank; virtual;
    procedure WriteInteger; virtual;
    procedure WriteDouble; virtual;
    procedure WriteLabel (var w : word); virtual;
    procedure WriteBoolean; virtual;
  end;

  PBIFF3 = ^TBIFF3;
  TBIFF3 = object(TBIFF2)
    procedure BIFFBOF; virtual;
    procedure BIFFDIM; virtual;
  end;

  PBIFF4 = ^TBIFF4;
  TBIFF4 = object(TBIFF3)
    procedure BIFFBOF; virtual;
  end;

  PBIFF5 = ^TBIFF5;
  TBIFF5 = object(TBIFF4)
    procedure BIFFBOF; virtual;
  end;

var PSaveFile : PBaseSave;

Implementation

{Generic save object}

Constructor TBaseSave.Init;
begin
  MinSaveRecs := 0; MaxSaveRecs := 100;
  MinSaveCols := 0; MaxSaveCols := 100;
  EndOfLine := false;
end;

Procedure TBaseSave.WriteBlank;
begin
  write( CharFile, separator );
end;

Procedure TBaseSave.WriteInteger;
var AIntegerP : ^integer; AInteger : integer;
begin
  AIntegerP := Data; AInteger := AIntegerP^;
  str(AInteger, DataString );
end;

Procedure TBaseSave.WriteDouble;
var ADoubleP : ^double; ADouble : double;
begin
  ADoubleP := Data; ADouble := ADoubleP^;
  str(ADouble, DataString );
end;

Procedure TBaseSave.WriteLabel;
var ALabelP : ^CharType; ALabel : CharType;
```

```
begin
  ALabelP := Data; ALabel := ALabelP^;
  DataString  := StrPas( ALabel );
  w := length(DataString); {unused by calling method}
end;

Procedure TBaseSave.WriteData;
var i : integer; AWordLength : word;
begin
  CellType := AType;
  if Row <> -1 then if Row <> ARow then EndOfLine := true else EndOfLine :=
false;
  Row := ARow;
  Col := ACol;
  Data := AData;

  case CellType of
    CellBlank   : WriteBlank;
    CellInteger : WriteInteger;
    CellDouble  : WriteDouble;
    CellLabel   : WriteLabel(AWordLength);
    CellBoolean : exit; {No boolean types in text files}
    else exit;
  end;

  if EndOfLine then begin write ( CharFile, CR ); write ( CharFile, LF )
end;
  for i := 1 to length(DataString) do write( CharFile, DataString[i] );
  write( CharFile, separator );

end;

Destructor TBaseSave.Done;
begin
end;

{ASCII files object}

Constructor TASCII.Init;
begin
  TBaseSave.Init( SaveFileName );
  Separator := Space;
  assign( CharFile, SaveFileName );
  Row := -1; col := -1;
  rewrite ( CharFile );
end;

Destructor TASCII.Done;
begin
  TBaseSave.Done; close( CharFile );
end;
```

```
{Excel tab-delimited files object}

Constructor TExcelTab.Init;
begin
  TBaseSave.Init( SaveFileName );
  Separator := tab;
  assign( CharFile, SaveFileName );
  Row := -1; col := -1;
  rewrite ( CharFile );
end;

Destructor TExcelTab.Done;
begin
  TBaseSave.Done; close( CharFile );
end;

{Excel BIFF2 object}

Constructor TBIFF2.Init;
begin
  TBaseSave.Init( AFileName );
  Assign( ExcelFile, AFileName); Rewrite( ExcelFile, 1 );
  WriteBOF;
  WriteDimensions;
end;

Destructor TBIFF2.Done;
begin
  TBaseSave.Done;
  WriteEOF;
  Close (ExcelFile);
end;

procedure TBIFF2.BIFFBOF;
begin
  typerec := BOF;
  lendata := 4;
end;

procedure TBIFF2.BIFFDIM;
begin
  typerec := DIMENSIONS;
  lendata := 8;
end;

procedure TBIFF2.WriteBOF;
var awBuf : array[0..2] of word;
begin
  awBuf[0] := 0;
  awBuf[1] := DOCTYPE_XLS;
  awBuf[2] := 0;
  BIFFBOF;
  WriteRecordHeader;
  Blockwrite(Excelfile, awbuf, lendata);
end;

procedure TBIFF2.WriteRecordHeader;
var awBuf : array[0..1] of word;
begin
  awBuf[0] := typerec;
  awBuf[1] := lendata;
  Blockwrite(Excelfile, awbuf, LEN_RECORDHEADER);
end;

procedure TBIFF2.WriteDimensions;
var awBuf : array[0..4] of word;
begin
  awBuf[0] := MinSaveRecs;
  awBuf[1] := MaxSaveRecs;
  awBuf[2] := MinSaveCols;
  awBuf[3] := MaxSaveCols;
  awBuf[4] := 0;
  BIFFDIM;
  WriteRecordHeader;
  Blockwrite(Excelfile, awbuf, lendata);
end;

procedure TBIFF2.WriteEOF;
begin
  typerec := BIFF_EOF;
  lendata := 0;
  WriteRecordHeader;
end;

Procedure TBIFF2.WriteBlank;
begin
  typerec := 1;
  lendata := 7;
  WriteRecordHeader;
  lendata := 0;
end;

Procedure TBIFF2.WriteInteger;
begin
  typerec := 2;
  lendata := 9;
  WriteRecordHeader;
  lendata := 2;
end;

Procedure TBIFF2.WriteDouble;
begin
  typerec := 3;
  lendata := 15;
  WriteRecordHeader;
  lendata := 8;
end;

Procedure TBIFF2.WriteLabel(var w : word);
begin
  w := strlen(Data);
  typerec := 4;
  lendata := 8+w;
  WriteRecordHeader;
  lendata := w;
end;

Procedure TBIFF2.WriteBoolean;
begin
  typerec := 5;
  lendata := 9;
  WriteRecordHeader;
  lendata := 0;
end;

Procedure TBIFF2.WriteData;
const
  Attribute: Array[0..2] Of Byte = (0, 0, 0); { 24 bit bitfield }
var
  awBuf : array[0..1] of word;
  AWordLength : word; ABoolByte : byte;
begin
  CellType := AType;
  Row := ARow;
  Col := ACol;
  Data := AData;

  case CellType of
    CellBlank   : WriteBlank;
    CellInteger : WriteInteger;
    CellDouble  : WriteDouble;
    CellLabel   : WriteLabel(AWordLength);
    CellBoolean : WriteBoolean; { or error }
    else exit;
  end;
  awBuf[0] := Row;
  awBuf[1] := Col;
  Blockwrite(Excelfile, awbuf, sizeof(awBuf));
  BlockWrite(Excelfile, Attribute, SizeOf(Attribute));

  if CellType = CellLabel then begin
    ABoolByte := AWordLength;
    BlockWrite(Excelfile, ABoolByte, SizeOf(ABoolByte))
  end else if CellType = CellBoolean then begin
    if byte(Data^) <> 0 then ABoolByte := 1 else ABoolByte := 0;
    BlockWrite(Excelfile, ABoolByte, SizeOf(ABoolByte));
    ABoolByte := 0;
    BlockWrite(Excelfile, ABoolByte, SizeOf(ABoolByte));
  end;
  if lendata <> 0 then BlockWrite(Excelfile, Data^, lendata);
end;

{Excel BIFF3 object}

procedure TBIFF3.BIFFBOF;
begin
  typerec := BOF_BIFF3;
  lendata := 6;
end;

procedure TBIFF3.BIFFDIM;
begin
  typerec := DIMENSIONS_BIFF3;
  lendata := 10;
end;

{Excel BIFF4 object}

procedure TBIFF4.BIFFBOF;
begin
  typerec := BOF_BIFF4;
  lendata := 6;
end;

{Excel BIFF5 object}

procedure TBIFF5.BIFFBOF;
begin
  typerec := BOF_BIFF5;
  lendata := 6;
end;

end.
---------End of WINPAS.PAS ------------

------ BIFFALL.PAS (Hauptprogramm) -----

{General purpose save library - (c) J. Gomes Ferreira 1994
 Writes records in Excel v.2.1, v.3, v.4,
 ASCII comma separated text,
 and tab-delimited excel text files

 {You have a royalty-free right to use, modify, reproduce, and distribute
  this file (and/or any modified version) in any way you find useful,
  provided that you agree that I offer no warranty, and have no obligations
  or liability for anything whatsoever relating to the use of the files
  contained herein.}
```

```
Program TestBiff;
{N+}
Uses Winsave,WinFdlg,strings,wintypes,winprocs;
var
  i,j : integer;
  ADouble : Double;
  AInteger : Integer;
  ALabel : array[0..10] of char;
  ABoolean : boolean;
  FullFileName : TFileName;
begin
  StrCopy(FullFileName,'c:\test.xls');
  {substitute the appropriate BIFF in the line below}
  PSavefile := New(PBIFF5,Init(FullFileName));
  ADouble := 1234.5678;
  AInteger := 25;
  StrCopy(ALabel,'10/11/94');
  ABoolean := True;
  with PSaveFile^ do begin
    for i := 0 to 9 do
      for j := 0 to 9 do
      begin
        case j of
          0: PSaveFile^.WriteData(CellLabel,i,j,@ALabel);
          1: PSaveFile^.WriteData(CellInteger,i,j,@AInteger);
          2: PSaveFile^.WriteData(CellBoolean,i,j,@ABoolean);
        else PSaveFile^.WriteData(CellDouble,i,j,@ADouble);
        end;
      end;
  end;
  dispose(PSaveFile,done);
  messagebox(0,'Job complete','BIFF any version',mb_ok);
end.
```