Apple II
File Type Notes

Developer Technical Support

File Type: $1B (27)
Auxiliary Type: All

Full Name: AppleWorks Spreadsheet File
Short Name: AppleWorks SS File

Revised by: Matt Deatherage & John Kinder, CLARIS Corporation September 1989
Written by: Bob Lissner February 1984

Files of this type and auxiliary type contain an AppleWorks® Spreadsheet file.
Changes since May 1989: Updated to include AppleWorks 2.1 and AppleWorks 3.0.

Files of type $1B and any auxiliary type contain an AppleWorks Spreadsheet file. AppleWorks is published by CLARIS. CLARIS also has additional information on AppleWorks files SEG.PR and SEG.ER. For information on AppleWorks, contact CLARIS at:

CLARIS Corporation
5201 Patrick Henry Drive
P.O. Box 58168
Santa Clara, CA 95052-8168

Technical Support
Telephone: (408) 727-9054
AppleLink: Claris.Tech

Customer Relations
Telephone: (408) 727-8227
AppleLink: Claris.CR

AppleWorks was created by Bob Lissner. AppleWorks 2.1 was done by Bob Lissner and John Kinder of CLARIS. AppleWorks 3.0 was done by Rob Renstrom, Randy Brandt and Alan Bird of Beagle Bros Software with John Kinder of CLARIS.

Definitions

The following definitions apply to AppleWorks files in addition to those defined for all Apple II file types:

MRL Data base multiple record layout

SRL	Data base single record layout
RAC	Review/Add/Change screen
DB	AppleWorks or /// E-Z Pieces Data Base
SS	AppleWorks or /// E-Z Pieces Spreadsheet
WP	AppleWorks or /// E-Z Pieces Word Processor
AW	AppleWorks or /// E-Z Pieces

Auxiliary Type Definitions

The volume or subdirectory auxiliary type word for this file type is defined to control uppercase and lowercase display of filenames. The highest bit of the least significant byte corresponds to the first character of the filename, the next highest bit of the least significant byte corresponds to the second character, etc., through the second bit of the most significant byte, which corresponds to the fifteenth character of the filename.

AppleWorks performs the following steps when it saves a file to disk:

1. Zeros all 16 bits of the auxiliary type word.
2. Examines the filename for lowercase letters. If one is found, it changes the corresponding bit in the auxiliary type word to 1 and changes the letter to uppercase.
3. Examines the filename for spaces. If one is found, it changes the corresponding bit in the auxiliary type word to 1 and changes the space to a period.

When files are read from disk, the filename and auxiliary type information from the directory file entry are used to determine which characters should be lowercase and which periods should be displayed as spaces. If you use the auxiliary type bytes for a different purpose, AppleWorks will still display the filenames, but the wrong letters are likely lowercase.

File Version Changes

Certain features present in AppleWorks 3.0 files are not backward-compatible to 2.1 and earlier versions. Such features are noted in the text. AppleWorks spreadsheet files which may not be loaded by versions prior to 3.0 are identified by a non-zero byte at location +242, referred to as location `SSMinVers`.

Those features added for AppleWorks 2.0, 2.1 and 3.0 not previously documented are indicated with that version number in the margin.

Spreadsheet Files

Spreadsheet files start with a 300 byte header record that contains basic information about the file, including column widths, printer options, window definitions, and standard values.

Header Record

The spreadsheet header record contains the following entries:

+000 to +003		Skip 4 bytes.
+004 to +130	**Bytes**	The column width for each column.
+131	**Byte**	Order of recalculation. ASCII R or C.
+132	**Byte**	Frequency of recalculation. ASCII A or M.
+133 to +134	**Word**	Last row referenced.
+135	**Byte**	Last column referenced.
+136	**Byte**	Number of windows: ASCII 1: just one window, S: side by side windows, T: top and bottom windows.
+137	**Byte**	Boolean: If there are two windows, are they synchronized?
+138 to +161		The next 20 (approximately) variables are for the current window. If there is only one window, it is the current window. If there are two windows, the current window is the window that had the cursor in it.
+138	**Byte**	Window standard format for label cells. 2: left justified, 3: right justified, 4: centered.
+139	**Byte**	Window standard format for value cells. 2: fixed, 3: dollars, 4: commas, 5: percent, 6: appropriate
+140	**Byte**	More of window standard format for value cells. Number of decimal places to display. Values from 0 to 7.
+141	**Byte**	Top screen line used by this window. This is the line that the `=====A=========B====` appears on. Normally 1 unless there are top and bottom windows.
+142	**Byte**	Leftmost screen column used by this window. This is the column that the hundreds digit of the row number appears in. Normally 0 unless there are side-by-side windows.
+143 to +144	**Word**	Top, or first, row appearing in titles area. This will probably be 0 if there are no top titles.
+145	**Byte**	Leftmost, or first, column appearing in left titles area. This will probably be 0 if there are no left titles.
+146 to +147	**Word**	Last row appearing in top titles area. This will probably be zero if there are no top titles.
+148	**Byte**	Last column appearing in left titles area. This will probably be zero if there are no left titles.
+149 to +150	**Word**	Top, or first, row appearing in the body of the window. The body is defined as those rows that are on the screen, but not in the titles area.
+151	**Byte**	Leftmost, or first, column appearing in the body of the window.
+152	**Byte**	The screen line that the top body row goes on. Normally 2, unless there are top titles or top and bottom windows.
+153	**Byte**	Leftmost screen column used for the leftmost body column. Normally 4 unless there are side titles, or side-by-side windows.
+154 to +155	**Word**	Bottom, or last, row appearing in this window.
+156	**Byte**	Rightmost, or last, column appearing in this window.

	+157	**Byte**	The screen line that the last body row goes on. Normally $13 (19) unless there are top and bottom windows.
	+158	**Byte**	The rightmost screen column used by this window. Normally $4E (78) unless there are side-by-side windows.
	+159	**Byte**	Number of horizontal screen locations used to display the body columns. Normally $48 (72), because 8 columns of 9 characters each are the standard display. This is affected by side-by-side windows, side titles, and variable column widths.
	+160	**Byte**	Boolean: Rightmost column is not fully displayed. This can only happen when the body portion of the window is narrower than the width of a particular column.
	+161	**Flag Byte**	Titles switch for this window. Bit 7: top titles, Bit 6: side titles. These bits represent top titles, side titles, both, and no titles.
	+162 to +185		Window information for the second window. This is meaningful only if there are two windows. This is the information for the window that the cursor is not currently in. See the descriptions for the current window (+138 to +161).
	+186 to +212		Not currently used.
	+213	**Byte**	Boolean: Cell protection is on or off.
	+214		Not currently used.
	+215	**Byte**	Platen width value, in 10ths of an inch. For example, a value of 80 inches entered by the user will show as 80 or $50.
	+216	**Byte**	Left margin value. All inches values are in 10ths of an inch.
	+217	**Byte**	Right margin value.
	+218	**Byte**	Characters per inch.
	+219	**Byte**	Paper length value, in 10ths of an inch.
	+220	**Byte**	Top margin value.
	+221	**Byte**	Bottom margin value.
	+222	**Byte**	Lines per inch. 6 or 8.
	+223	**Byte**	Spacing: S(ingle, D(ouble, or T(riple. Expect these three letters, even in European versions.
	+224 to +237	**Bytes**	If user has specified “Send special codes to printer,” this is a 13-byte string containing those codes.
	+238	**Byte**	Boolean: Print a dash when an entry is blank.
	+239	**Byte**	Boolean: Print report header.
	+240	**Byte**	Boolean: Zoomed to show formulas.
2.1	+241	**Byte**	Reserved; used internally.
3.0	+242	**Byte**	`SSMinVers`. The minimum version of AppleWorks needed to read this document. If this document contains version 3.0-specific functions (such as calculated labels or new functions), this byte will contain the version number 30 ($1E). Otherwise, it will be zero ($00).
	+243 to +249		Reserved for future use.
	+250 to +299		Available. Will never be used by AppleWorks. If you are creating these files, you can use this area to keep information that is important to your program.

Row Records

Row records contain a variable amount of information about each row that is non-blank. Each row record contains enough information to completely build one row of the spreadsheet:

3.0	+000 to +001	**Word**	Number of additional bytes to read from disk. $FFFF means end of file. If `SFMinVers` is not zero, these two bytes are invalid and should be skipped. The first row record begins at +302 in an AW 3.0 SS file.
	+002 to +003	**Word**	Row number.
	+004	**Byte**	Beginning of actual information for the row. This byte of each record will always be a control byte. Other control bytes within each record define the contents of the record. Control bytes may be:

$01-$7F	This is a count of the number of following bytes that are the contents of a cell entry.
$81-$FE	This (minus $80) is a count of the number of columns to be skipped. For example, $82 means skip two columns.
$FF	This indicates the end of the row.

Cell Entries

Cell entries contain all the information that is necessary to build one cell. There are several types:

Value Constants

Value constants are cells that have a value that cannot change. This means that someone typed a constant into the cell, 3.14159, for example.

+000	**Flag Byte**	Bit 7 is always on. Bit 6 on means that if the value is zero, display a blank instead of a zero. This is for pre-formatted cells that still have no value. Bit 5 is always on. Bit 4 on means that labels cannot be typed into this cell. Bit 3 on means that values cannot be typed into this cell. Bits 2,1, and 0 specify the formatting for this cell:

1	Use spreadsheet standard
2	Fixed
3	Dollars
4	Commas
5	Percent
6	Appropriate

+001	**Flag Byte**	Bit 7 is always zero. Bit 6 is always zero. Bit 5 is always zero.

		Bit 4 on indicates that this cell must be calculated the next time this spreadsheet is calculated, even if none of the referenced cells are changed. This bit makes sense on for cells that have a calculated formula. Bits 2, 1, and 0: Number of decimal places for fixed, dollars, commas, or percent formats.
+002 to +009		8-byte SANE double format floating point number.

Value Labels

Note: The entire Value Labels cell record entry requires AppleWorks 3.0 or later.

Value labels are cells whose function has returned a label value. Formulas like `@Lookup`, `@Choose` and `@IF` can all return labels as their results. Specific format:

+000	**Flag Byte**	Bit 7 is always one. Bit 6 on means not to display the cell. This was originally intended for pre-formatted cells that still have no value. If a value is placed in this cell, be sure to turn this bit off. Bit 5 is always zero. Bits 4, 3, 2, 1, and 0 are the same as regular label cells.
+001	**Flag Byte**	Bit 7 is always one. Bit 6 set indicates the last evaluation of this formula resulted in `@NA`. Bit 5 set indicates the last evaluation of this formula resulted in `@Error`. Bit 4 on indicates that this cell must be calculated the next time this spreadsheet is calculated, even if none of the referenced cells are changed. Bit 3 is always one. Bits 2 – 0 are ignored.
+002 to nnn	**String**	Pascal string containing characters to display.
+nnn+1 to xxx	**Bytes**	Various control bytes that are "tokens" representing the formula that was typed by the user. They are defined below.

Value Formulas

Value formulas are cells that contain information that has to be evaluated. Formulas like `AA17+@sum(r19...r21)` and `@Error` are examples. Specific format:

+000	**Flag Byte**	Bit 7 is always on. Bit 6 on means to not display the cell. This was originally intended for pre-formatted cells that still have no value. If a value is placed in this cell, be sure to turn off this bit. Bit 5 is always off. Bits 4, 3, 2, 1, and 0 are the same as value constants.
+001		Bit 7 is always on. Bit 6 on indicates that the last evaluation of this formula resulted in an `@NA`. Bit 5 on indicates that the last evaluation of this formula resulted in an `@Error`. Bits 4, 2, 1, and 0 are the same as value constants.
+002 to +009		8-byte SANE double floating point number that is the most recent evaluation of this cell.

+010 to nnn

Various control bytes that are tokens representing the formula that was entered by the user. They are:

	Byte	Means
3.0	$C0	@Deg
3.0	$C1	@Rad
3.0	$C2	@Pi
3.0	$C3	@True
3.0	$C4	@False
3.0	$C5	@Not
3.0	$C6	@IsBlank
3.0	$C7	@IsNA
3.0	$C8	@IsError
3.0	$C9	@Exp
3.0	$CA	@Ln
3.0	$CB	@Log
3.0	$CC	@Cos
3.0	$CD	@Sin
3.0	$CE	@Tan
3.0	$CF	@ACos
3.0	$D0	@ASin
3.0	$D1	@ATan2
3.0	$D2	@ATan
3.0	$D3	@Mod
3.0	$D4	@FV
3.0	$D5	@PV
3.0	$D6	@PMT
3.0	$D7	@Term
3.0	$D8	@Rate
2.0	$D9	@Round
2.0	$DA	@Or
2.0	$DB	@And
	$DC	@Sum
	$DD	@Avg
	$DE	@Choose
	$DF	@Count
	$E0	@Error (followed by 3 bytes of zero)
3.0	$E1	@IRR
	$E2	@If
	$E3	@Int
	$E4	@Lookup
	$E5	@Max
	$E6	@Min
	$E7	@NA (followed by three bytes of zero)
	$E8	@NPV
	$E9	@Sqrt
	$EA	@Abs
	$EB	Not currently used
	$EC	Not equal (<>)
	$ED	greater than or equal to (>=)
	$EE	less than or equal to (<=)
	$EF	equals (=)
	$F0	greater than (>)
	$F1	less than (<)
	$F2	comma (,)
	$F3	exponentiation sign (^)

	$F4	right parenthesis (“)”)
	$F5	minus (-)
	$F6	plus (+)
	$F7	divide (/)
	$F8	multiply (*)
	$F9	left parenthesis (“(”)
	$FA	unary minus (-) i.e., -A3
	$FB	(unary plus (+) i.e., +A3)
	$FC	ellipses (…)
	$FD	Next 8 bytes are SANE double number
	$FE	Next 3 bytes are row, column reference
3.0	$FF	Next n bytes are a Pascal string

Three of the codes require special information. Code $FD indicates that the next 8 bytes are a SANE numerics package double precision floating point number. All constants within formulas are carried in this manner.

Code $FE indicates that the next three bytes point at a cell:

+000	**Byte**	$FE
+001	**Byte**	Column reference. Add this byte to the column number of the current cell to get the column number of the pointed at cell. This value is sometimes negative, but Add always works.
+002 to +003	**Word**	Row reference. Add this word to the row number of the current cell to get the row number of the pointed at cell. This value is sometimes negative, but Add always works.

Code $FF indicates that the next bytes are a **String**, where the byte immediately following the $FF contains the length.

Propagated Label Cells

Propagated label cells are labels that place one particular ASCII character in each position of a window. Handy for visual effects like underlining.

+000	**Flag Byte**	Bit 7 is always zero. Bit 6 is meaningless. Bit 5 is always on. Bit 4 and bit 3 are protection, just like value cells. Bits 2, 1, and 0 are meaningless. Put a 1 here.
+001	**Byte**	This is the actual character that is to be put in each position in the cell.

Regular Label Cells

Regular label cells contain alphanumeric information, such as headings, names, and other descriptive information.

+000	**Flag Byte**	Bits 7, 6, and 5 are always zero. Bits 4 and 3 are same as value cells. Bits 2, 1, and 0 determine cell formatting: 01 Use spreadsheet standard formatting 02 Left justify 03 Right justify 04 Center
+001 to +nnn	**Bytes**	ASCII characters that actually display. The actual length was defined earlier in the word that contained the actual number of bytes to read from disk.

File Tags

All AppleWorks files normally end with two bytes of $FF; **tags** are anything after that. Although File Tags were primarily designed by Beagle Bros, they can be used by any application that needs to create or modify an AppleWorks 3.0 file.

Because versions of AppleWorks before 3.0 stop at the double $FF, they simply ignore tags.

The File Tag structure is as follows:

+000	**Byte**	Tag ID. Should be $FF.
+001	**Byte**	2nd ID byte. These values will be defined and arbitrated by Beagle Bros Software. Beagle may be reached at: Beagle Bros Inc 6215 Ferris Square, #100 San Diego, CA 92121

+002 to +003	**Word**	Data length. If this is the last tag on the file, the low byte (+002) will be a count of the tags in this file, and the high byte (+003) will be $FF.
+004 to nnn	**Bytes**	Actual tag data, immediately followed by the next four-byte tag ID. These bytes do not exist for the last tag.

There is a maximum of 64 tags per file. Each tag may be no larger than 2K.

AppleWorks is a registered trademark of Apple Computer, Inc. licensed to Claris Corporation.