Apple II
File Type Notes

Developer Technical Support

File Type: $1A (26)
Auxiliary Type: All

Full Name: AppleWorks Word Processor File
Short Name: AppleWorks WP 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® Word Processor file.
Changes since May 1989: Updated to include AppleWorks 2.1 and AppleWorks 3.0.

Files of type $1A and any auxiliary type contain an AppleWorks Word Processor 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 Alan Bird, Rob Renstrom and Randy Brandt 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 Word Processor files which may not be loaded by versions prior to 3.0 are identified by a non-zero byte at location +183, referred to as location `SFMinVers`.

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

Word Processor Files

Word Processor files start with a 300 byte header, followed by a number of variable length line records, one for each line on the screen.

Header Record

The header contains the following information:

	+000 to +003		Not used.
	+004	**Byte**	$4F (79)
	+005 to +084	**Bytes**	Tab stops. Either equal sign (=) or vertical bar (|) If `SFMinVers` is non-zero, these will be one of the following values: "=" - no tab "<" - left tab "^" - center tab ">" - right tab "." - decimal tab.
	+085	**Byte**	Boolean: Zoom switch.
	+086 to +089		Four bytes not used.
	+090	**Byte**	Boolean: Whether file is currently paginated (i.e., whether the page break lines are displayed).
	+091	**Byte**	Minimum left margin that should be added to the margin that is appearing on the screen. This is normally one inch, shown in 10ths of an inch, 10 or $0A.
	+092	**Byte**	Boolean: Whether file contains any mail-merge commands.
	+093 to +175	**Bytes**	Not used. Reserved.
3.0	+176	**Byte**	Boolean: Whether there are multiple rulers in the document.
3.0	+177 to +182	**Bytes**	Used internally for keeping track of tab rulers.
3.0	+183	**Byte**	`SFMinVers`. The minimum version of AppleWorks needed to read this document. If this document contains 3.0 specific features (tabs and multiple tab rulers, for example), this byte will contain the version number 30 ($1E). Otherwise, it will be zero ($00).
	+184 to +249	**Bytes**	Reserved.
	+250 to +299	**Bytes**	Available. Will never be used by AppleWorks. If you are creating this type of file, you can use this area to keep information that is important to your program.

Line Records

Line records are of three different types. The first line record after the 300 byte header corresponds to line 1, the next is line 2, and so on. The first two bytes of each line record contain enough information to establish the type.

If `SFMinVers` is non-zero, the first line record (two bytes long) is invalid and should be skipped.

Carriage Return Line Records

Carriage return line records have a $D0 (208) in byte +001. Byte +000 is a one byte integer between 00 and 79 that is the horizontal screen position of this carriage return.

Command Line Records

Command line records are formatting commands that appear on the screen in the form:

```
--------Double Space
```

for example. These records can be identified by a value greater than $D0 (208) in byte +001. They are:

	Byte +001	Command	Byte +000	
3.0	$D4	reserved	(used internally as ruler)	
3.0	$D5	Page header end		
3.0	$D6	Page footer end		
3.0	$D7	Right justified		
	$D8	Platen width	**Byte**	10ths of an inch
	$D9	Left margin	**Byte**	10ths of an inch
	$DA	Right margin	**Byte**	10ths of an inch
	$DB	Chars per inch	**Byte**	
	$DC	Proportional-1		No meaning
	$DD	Proportional-2		
	$DE	Indent	**Byte**	Characters
	$DF	Justify		
	$E0	Unjustify		
	$E1	Center		
	$E2	Paper length	**Byte**	10ths of an inch
	$E3	Top margin	**Byte**	10ths of an inch
	$E4	Bottom margin	**Byte**	10ths of an inch
	$E5	Lines per inch	**Byte**	
	$E6	Single space		
	$E7	Double space		
	$E8	Triple space		
	$E9	New page		
	$EA	Group begin		
	$EB	Group end		
	$EC	Page header		
	$ED	Page footer		
	$EE	Skip lines	**Byte**	Count
	$EF	Page number	**Byte**	
	$F0	Pause each page		
	$F1	Pause here		
	$F2	Set marker	**Byte**	Marker number
	$F3	Page number	**Byte**	(add 256)
	$F4	Page break	**Byte**	Page number
	$F5	Page break	**Byte**	(add 256)
	$F6	Page break	**Byte**	(break in middle of paragraph)
	$F7	Page break	**Byte**	(add 256 in middle of paragraph)
	$FF	End of file		

Text Records

Text records are the lines where text has been typed. The format is:

	+000 to +001	**Word**	Number of bytes following this word. Since the maximum is about 80, byte +001 is always zero. Use byte +001 to identify text lines.
3.0	+002		If bit 7 is on, this line contains Tab and Tab Filler special codes (described below). The remaining seven bits are the screen column for the first text character. Usually will be zero, but may vary as a result of left margin, centering, and indent commands. If this byte is $FF, this text line is actually a ruler—ASCII equivalent of what appears on the top of the screen.
	+003	**Byte**	If bit 7 (the high bit) of this byte is on, there is a carriage return on the end of this line. If off, no carriage return. Bits 6-0: Number of bytes of text following this byte.
	+004 to nnn		Actual text bytes. Consists of ASCII characters and special codes. The special codes are values from $01 to $1F, and indicate special formatting features:

	Code	Meaning
	$01	Begin boldface
	$02	Boldface end
	$03	Superscript begin
	$04	Superscript end
	$05	Subscript begin
	$06	Subscript end
	$07	Underline begin
	$08	Underline end
	$09	Print page number
	$0A	Enter keyboard
	$0B	Sticky space
	$0C	Begin Mail merge
3.0	$0D	Reserved
3.0	$0E	Print Date
3.0	$0F	Print Time
3.0	$10	Special Code 1
3.0	$11	Special Code 2
3.0	$12	Special Code 3
3.0	$13	Special Code 4
3.0	$14	Special Code 5
3.0	$15	Special Code 6
3.0	$16	Tab character
3.0	$17	Tab fill character (used in formatting lines)
3.0	$18	Reserved

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.