Apple II
File Type Notes

Developer Technical Support

File Type: $E0 (224)
Auxiliary Type: $0005

Full Name: DiskCopy disk image
Short Name: DiskCopy disk image

Written by: Matt Deatherage, Dave Lyons & Steve Christensen May 1992

Files of this type and auxiliary type contain disk images from Apple's DiskCopy program on the Macintosh.

DiskCopy is a program written by Steve Christensen of Apple Computer, Inc., for internal use in duplicating and distributing 3.5" floppy disks. Because of its utility in distributing disk images on the Macintosh, DiskCopy is used in several Apple developer products even though DiskCopy is not an official Apple product and is not supported as such.

Since the monthly *Developer CD Series* discs contain many DiskCopy disk images, and since the AppleShare and HFS FSTs in System Software 6.0 and later automatically translate DiskCopy files (HFS file type `dImg` and creator `dCpy`) to Apple II file type $E0 and auxiliary type $0005, the format is provided here for your utility use only. Apple does not guarantee that files not generated by DiskCopy will work with DiskCopy.

Definitions

DiskCopy uses a simple checksum algorithm to help insure data integrity for archived disk images. The algorithm for generating the 32-bit checksum is as follows:

```
Initialize checksum to zero
For each data Reverse Word:
        Add the data Reverse Word to the checksum
        Rotate the 32-bit checksum right one bit (wrapping bit 0 to bit 31)
```

The following 65816 assembly language routine calculates a DiskCopy checksum. It's not a speedy operation—it takes about 12 seconds to calculate the checksum on an 800K disk image. Anyone finding an assembly routine that can perform this task in under 5 seconds may apply for their IIGS Certificate of Deityship, as documented in the File Type Note for file type $B6.

(Oh, by the way, any entries have to be under 1K in size—the following routine is 88 bytes. So don't think unwinding loops is your ticket to fame and fortune.)

```
*******************************************************************************
*
* Compute checksum for DiskCopy data
*
* v1.2  by David A. Lyons, 18-May-92
*
* MPW IIgs assembly format
*
* Inputs on stack:
*    Push pointer to data (long)
*    Push size of data (long)  (Must be even!)
*    JSL CalcChecksum
*    STA TheChecksum+2
*    STX TheChecksum
*
* Output:
*    Checksum in A and X (bytes +0 and +1 in X, bytes +2 and +3 in A)
*    (The inputs have been removed from the stack)
*
*******************************************************************************
CalcChecksum        PROC
                    phd                 ;save caller's direct page reg
                    lda #0
                    pha
                    pha                 ;push initial checksum value (zero)
                    tsc
                    tcd

checksum            equ 1
oldD                equ checksum+4
theRTL              equ oldD+2
dataSize            equ theRTL+3
dataPtr             equ dataSize+4

*** Set dataSize to -(dataSize/2)-1 so we can count up by one
*** (instead of down by two) to see when we're done
                    lda <dataSize+2
                    lsr a
                    eor #$ffff
                    sta <dataSize+2
                    lda <dataSize
                    ror a
                    eor #$ffff
                    sta <dataSize

                    ldy #0
nextWord            inc <dataSize
                    bne moreData
                    inc <dataSize+2
                    beq noMoreData
moreData

*** Get next 16-bit word from the data buffer
                    lda [<dataPtr],y
                    xba                 ;swap to 65816 byte order

*** Add the data word to the checksum
                    clc
                    adc <checksum
                    sta <checksum
                    bcc noCksumRoll
                    inc <checksum+2
noCksumRoll
```

```
*** Rotate the 32-bit checksum right one bit, wrapping bit 0 into bit 31
                    lda <checksum+2
                    lsr a
                    ror <checksum
                    bcc bit0was0
                    ora #$8000          ;if we rotated a 1 out of bit 0,
bit0was0            sta <checksum+2     ;  then set bit 31

*** Advance to the next word and go back for more
                    iny
                    iny
                    bne nextWord        ;go back for more data
                    inc <dataPtr+2
                    bra nextWord        ;go back for next bank of data

noMoreData          pla
                    xba
                    tay
                    pla
                    xba
                    tax                 ;pull checksum into YX (put in 68000 order)

                    pld                 ;restore caller's direct page reg

                    lda 2,s
                    sta 2+8,s
                    lda 1,s
                    sta 1+8,s
                    pla
                    pla
                    pla
                    pla                 ;discard input values

                    tya
                    rtl

                    EndP

                    END
```

The following definition is used in this document in addition to those defined for all Apple II file types:

Checksum A 32-byte quantity calculated using the previously-defined algorithm. When these are contained in the file, they are in **Reverse** order.

File Structure

All of the information for a DiskCopy disk image is in the data fork. The resource fork usually contains Macintosh resources (in Macintosh resource fork format), including `vers` resources listing the checksums. This allows Macintosh users to use the Macintosh Finder's "Get Info…" function to quickly examine the checksums.

The File Format

Because this is a native Macintosh file format, all the multi-byte constants are stored in **Reverse** order.

diskName	(+000)	**64 Bytes**	A Pascal **String** containing the name of the disk. This field takes 64 bytes regardless of the length of the **String**.
dataSize	(+064)	**Rev. Long**	The number of bytes (not blocks) of user data. User data is the 512 bytes of each block that a normal block-reading command returns.
tagSize	(+068)	**Rev. Long**	The number of bytes of tag data. Tag data is the extra 12 bytes of "scavenger" information present on 400K and 800K Mcaintosh disks. Apple II operating systems always leave these bytes zeroed, and they're not present on 720K or 1440K disks. If there are no tag bytes, this field will be zero.
dataChecksum	(+072)	**Checksum**	Checksum of all the user data on the disk. The checksum algorithm is called for the entire disk, not on a block-by-block or sector-by-sector basis. This is in **Reverse** order (most significant byte first).
tagChecksum	(+076)	**Checksum**	Checksum of all the tag data on the disk. If there is no tag data, this should be zero. This is in **Reverse** order (most significant byte first).
diskFormat	(+080)	**Byte**	0 = 400K 1 = 800K 2 = 720K 3 = 1440K (all other values are reserved)
formatByte	(+081)	**Byte**	$12 = 400K $22 = >400K Macintosh (DiskCopy uses this value for all Apple II disks not 800K in size, and even for some of those) $24 = 800K Apple II disk
private	(+082)	**Rev. Word**	Must be $0100. If this field is not $0100, the file may be in a different format.
userData	(+084)	**dataSize Bytes**	The data blocks for the disk. These are in order from block zero through the end of the disk.
tagData	(+xxx)	**tagSize Bytes**	The tag data for this disk, starting with the tag data for the first block and proceeding in order. This field is not present for 720K and 1440K disks, but it is present for all other formats even if all the data is zeroes.

Further Reference

- *GS/OS Reference*