File-IO Routines#

by Bill Wilkinson

Modified 6/30/88 by Don Davis

General Information

Author: Bill Wilkinson / Don Davis
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: 6/30/88

This is a substantially modified version of Bill Wilkinson's IOFUNC.ACT, which is also in this Data Library. It should be considered a replacement for the original file and for IOFUN2.ACT, which, as it turns out, didn't go far enough.

Modified 6/30/88 by Don Davis

The major failing of Bill's original set of routines was that none of the routines altered the bytes in the EOF array to reflect the proper end-of-file status of the corresponding IOCBs. While this was not a fatal omission, it was, at times, a nuisance. In addition, there was a bug (actually two bugs) in the F_gp() procedure. The first caused the FGet() and FGetD() functions to return the CIO status every time they were called, instead of returning a data byte. The second sometimes caused more than one byte to be read from or written to the appropriate channel by the get/put functions. The F_gp() support procedure had been modified to correct the bugs. Another modification was made to allow the FGet() and FGetD() functions to set the EOF flags when they encounter an end-of-file error. The F_rw() support procedure has been similarly altered to allow FRead() to set an EOF flag when it encouters an end-of-file status. FOpen() has been modified to close the specified IOCB before attempting to open it. FOpen() calls FClose() to do this, so if you need the first, you can't delete the second. FOpen() has also been changed so that a successful open operation clears the EOF flag for that IOCB, and a failure sets the flag to 1. F_eof() has been added to handle the actual setting and clearing of EOF flags.

Replacement I/O library for Action!

This library implements vital I/O via FUNCtions rather than PROCs... the calling routine can thus examine the returned (error?) value and act accordingly. Functions returning bytes simply return the normal Atari CIO OS error code. Functions returning INTegers return appropriate information (e.g., byte count) if the returned value is positive.

When the returned value is negative, the error code will be in the lower byte. Note that code block routines are generally broken up along assembly language instruction boundaries. CAUTION: I have not provided support for XIO routines that require a buffer address and length (e.g., some forms of the concurrent IO cmd for R:). You can NOT use XIOFN or XIOAXFN for these purposes, because the "FN" (filename) is copied to a buffer and has a zero byte appended to it. CAUTION: I do NOT put "D:" in front of file names that lack an colon. If you need this feature, add it. But I felt that "D:" is a pretty poor default today, what with multiple drives, subdirectories, etc.

These routines written by Bill Wilkinson of OSS They are public domain and may be freely used and copied.

```
; IOFUN3.ACT
;
; This is a substantially modified
; version of Bill Wilkinson's
```

```
; IOFUNC.ACT, which is also in this
; Data Library. It should be
; considered a replacement for the
; original file and for IOFUN2.ACT,
; which, as it turns out, didn't go
; far enough.

; Modified 6/30/88 by Don Davis

; The major failing of Bill's original
; set of routines was that none of the
; routines altered the bytes in the
; EOF array to reflect the proper
; end-of-file status of the
; corresponding IOCBs. While this was
; not a fatal omission, it was, at
; times, a nuisance.

; In addition, there was a bug (actually
; two bugs) in the F_gp() procedure.
; The first caused the FGet() and
; FGetD() functions to return the CIO
; status every time they were called,
; instead of returning a data byte.
; The second sometimes caused more than
; one byte to be read from or written
; to the appropriate channel by the
; get/put functions.

; The F_gp() support procedure had been
; modified to correct the bugs.
; Another modification was made to
; allow the FGet() and FGetD() functions
; to set the EOF flags when they
; encounter an end-of-file error.

; The F_rw() support procedure has been
; similarly altered to allow FRead()
; to set an EOF flag when it encouters
; an end-of-file status.

; FOpen() has been modified to close the
; specified IOCB before attempting to
; open it. FOpen() calls FClose() to do
; this, so if you need the first, you
; can't delete the second. FOpen() has
; also been changed so that a
; successful open operation clears the
; EOF flag for that IOCB, and a failure
; sets the flag to 1.

; F_eof() has been added to handle the
; actual setting and clearing of
; EOF flags.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Replacement I/O library for Action!
;
```

```
; This library implements vital I/O
; via FUNCtions rather than PROCs...
; the calling routine can thus
; examine the returned (error?) value
; and act accordingly.
;
; Functions returning bytes simply
; return the normal Atari CIO OS
; error code.
;
; Functions returning INTegers return
; appropriate information (e.g., byte
; count) if the returned value is
; positive.  When the returned value
; is negative, the error code will
; be in the lower byte.
;
; Note that code block routines are
; generally broken up along assembly
; language instruction boundaries.
;
; CAUTION:  I have not provided support
; for XIO routines that require a buffer
; address and length (e.g., some forms
; of the concurrent IO cmd for R:).  You
; can NOT use XIOFN or XIOAXFN for these
; purposes, because the "FN" (filename)
; is copied to a buffer and has a zero
; byte appended to it.
;
; CAUTION:  I do NOT put "D:" in front
; of file names that lack an colon.  If
; you need this feature, add it.  But
; I felt that "D:" is a pretty poor
; default today, what with multiple
; drives, subdirectories, etc.
;
; These routines written by
;           Bill Wilkinson of OSS
; They are public domain and may be
; freely used and copied.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; support subroutines
;
BYTE ARRAY F_buf(129)

BYTE POINTER F_bpt = F_buf

PROC F_eof=*() ; support for F_gp(),
;                    F_rw() and FOpen()
    [$A2$C0 $86$AC $A2$05 $86$AD
     $91$AC $60]
;
PROC F_xio=*() ; support various XIOs
    [$48 $8A $0A$0A$0A$0A $AA $68
     $9D$42$03 $98 $A4$A3 $60]
;
```

```
PROC F_rw=*() ;support for READ/WRITE
    [$85$A0 $86$A1 $0A$0A$0A$0A $AA
     $A5$A1 $9D$44$03 $A5$A2 $9D$45$03
     $98 $9D$42$03 $A5$A3 $9D$48$03
     $A5$A4 $9D$49$03 $20$56$E4 $10$11
     $A9$FF $C0$88 $D0$11 $86$A6 $A9$01
     $A4$A0 $20 F_eof $A6$A6 $BC$48$03
     $BD$49$03 $85$A1 $84$A0 $60]
;
PROC F_gp=*() ; support for GET/PUT
     [$86$A1 $85$A2 $0A$0A$0A$0A $AA
      $98 $9D$42$03 $A9$00 $9D$48$03
      $9D$49$03 $A5$A1 $20$56$E4 $85$A0
      $A2$00 $98 $10$14 $C9$88 $D0$0D
      $84$A1 $A4$A2 $A9$01 $20 F_eof
      $A2$00 $A4$A1 $CA $84$A0 $86$A1
      $60]
;
PROC F_cnm=*(); process CIO filenames
    [$85$AE $84$AF $A0$00 $B1$AE $A8
     $A9$00 $99 F_buf $B1$AE $88 $10$F8
     $AD F_bpt $9D$44$03
     $AD F_bpt+1 $9D$45$03
     $20$56$E4 $84$A0 $60]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Begin the call-able routines:
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; XIOAX -- used when an XIO command
;            needs to specify AUX1 & 2
;            but does not need to give
;            a file name.   XIO op's on
;            previously OPENed files
;            can generally use this!

BYTE FUNC XioAX=*
      ( BYTE cmd,chan,aux1,aux2 )
    [$20 F_xio $9D$4A$03 $98
     $9D$4B$03 $20$56$E4 $84$A0 $60]

; XIOFN -- used when an XIO command
;            does NOT want to specify
;            AUX1 and AUX2 values! The
;            original ACTION library
;            did NOT set AUX1 or AUX2
;            when AUX1 was zero, so this
;            MAY be an appropriate sub-
;            stitute for that routine.

BYTE FUNC XioFN=*
      ( BYTE cmd,chan
        BYTE ARRAY filename )
    [$20 F_xio $4C F_cnm ]

; XIOAXFN -- full-blown form of XIO,
;            where you specify AUX1,
```

```
;            AUX2, and filename.  Note
;            that this version ALWAYS
;            stores AUX1 and AUX2 into
;            the IOCB (a la Atari BASIC)
;            and should be used under
;            appropriate circumstances.

BYTE FUNC XioAXFN=*
      ( BYTE cmd,chan,aux1,aux2
         BYTE ARRAY filename )
    [$20 F_xio $9D$4A$03 $98
     $9D$4B$03 $A5$A4 $A4$A5
     $4C F_cnm ]

; FCLOSE - Similarly, this is now a
;            FUNCtion.  Otherwise as
;            in original library.

BYTE FUNC FClose=*
      ( BYTE chan )
    [$0A$0A$0A$0A $AA $A9$0C
     $9D$42$03 $20$56$E4 $84$A0 $60]

; FOPEN -- Same form and order as
;            ATARI BASIC!   Note that
;            in original ACTION library
;            routine the order of arg-
;            uments is different.  Also
;            this is now a FUNCtion.

BYTE FUNC FOpen=*
      ( BYTE chan
         BYTE aux1,aux2
         BYTE ARRAY filename )
    [$85$A2 $86$A1 $84$A5 $20 FClose
     $98 $30$18 $A6$A2 $A4$A5 $A9$03
     $20 F_xio $9D$4B$03 $A5$A1
     $9D$4A$03 $A5$A3 $A4$A4 $20 F_cnm
     $A9$00 $C0$80 $30$02 $A9$01 $A4$A2
     $4C F_eof ]

; FREAD and FWRITE -- both these are
;            written to be similar to
;            the standard C routines
;            of the same names.  They
;            are designed to transfer
;            an arbitrary number of
;            bytes to/from a file (up to
;            32767 bytes).  Then, if the
;            operation was successful,
;            they return the count of
;            bytes actually read.
;            NOTE:  an end-of-file (EOF,
;            error 136 in Atari parlance
;            is NOT considered an error.
;            If the count of bytes read
;            does not match the request,
;            then presumably an EOF is
;            the culprit. If so, the
```

```
;            corresponding byte in the
;            EOF array will show a value
;            of 1.

INT FUNC FRead=*
      ( BYTE chan
         BYTE ARRAY buffer
         INT buffersize )
     [$84$A2 $A0$07 $4C F_rw ]

INT FUNC FWrite=*
      ( BYTE chan
         BYTE ARRAY buffer
         INT buffersize )
     [$84$A2 $A0$0B $4C F_rw ]

; FGET/FPUT/FGETD/FPUTD -- these
;            routines work much the same
;            as GET/PUT/GETD/PUTD do,
;            but again these are FUNCs,
;            not PROCs, so the user may
;            test the returned value
;            for an error instead of
;            having to mess with an
;            ERROR() routine!

INT FUNC FGet=*() ; get from channel 0
     [$A9$00] ; fall thru to FGetD!!

INT FUNC FGetD=*
     ( BYTE chan )
     [$A0$07 $4C F_gp ]

INT FUNC FPut=* ; put to channel 0
     ( BYTE data )
     [$AA $A9$00] ; fall thru to FPutD!!

INT FUNC FPutD=*
     ( BYTE chan,data )
     [$A0$0B $4C F_gp ]

;**********************************
;**********************************

; The following code is _NOT_ part
; of the library!!!!!!!!
;
; When using the library, delete all
; code past the two lines of asterisks
; above!!!!

;
; Demonstration:  using the above
;    routines to copy a file to the
;    screen.    Stupid, but it works!
;

PROC DemoCopy()
  DEFINE bigsize="8000" ; or ANY size!
```

```
  BYTE ARRAY bigbuf(bigsize)
  INT count

     Close(2) ; old form! ignore error!
     IF FOpen(2,4,0,"D:IOFUNCS.ACT")>127
     THEN
        PrintE("Can't find that file!")
     ELSE
        DO
          count=FRead(2,bigbuf,bigsize)
          IF count>0
          THEN
             FWrite(0,bigbuf,count)
          FI
        UNTIL count<>bigsize OD
     FI
     Close(2) ; old form, file was open
                ; for read, so ignore err
     RETURN

;
; note the valid mixing of I/O styles!
; feel free to change the filename in
; the FOpen line!
;

; Good luck, Bill W.

;**********************************
;**********************************
```