Assembly and Files under ProDOS

RTK, last update: 23-Jul-97

Assembly and Files

This file is specific to the Apple II series of computers running ProDOS. It explains how to make simple use of disk files via assembly language.

==========

MLI and commands

Most disk access from within an Apple II assembly language program uses the machine language interface supplied by the ProDOS operating system. Commands consist of a call to $BF00 followed by a table of command parameters. If ProDOS returns a value it is returned in a location within this table. MacQForth implements a useful subset of the standard ProDOS commands. The general form of a command is:

```
JSR $BF00           ;  call ProDOS to do the command
(command-number)
(address-of-parameter-table)
(execution returns to here when finished)
```

If an error happens, the carry flag is set and the accumulator contains the error code. MacQForth returns 0 if no error or the absolute value of the corresponding Macintosh file error number if an error happened.

The contents of the parameter table vary from command to command, but a general form is:

```
(number-of-parameters)
(parameters)
.
.
```

So, a possible assembly language calling sequence to read some data from an open file (file 0) might be:

```
ldx #$00
ldy #$10
sty params+4
stx params+5    ; setup number of bytes to read (16)
jsr $BF00       ; call ProDOS
.BYTE $CA       ; ProDOS command number = CA (read)
.WORD params    ; address of parameter table, lo/hi
```

```
          bcs error       ; carry set, error
          .
          .

  params .BYTE $04        ; number of parameters for a read
          .BYTE $00       ; file reference number, 0, 1, 2 in MacQForth
          .WORD BUFFER    ; pointer to data buffer
          .WORD $0000     ; requested number of bytes to read, fill in
          .WORD $0000     ; number actually read, returned by ProDOS
```

The following ProDOS commands are available and their parameters are outlined below:

```
$C0  =  create a new file
$C1  =  destroy an existing file
$C4  =  get file info (a dummy command, do not use)
$C8  =  open a file (reference numbers 0, 1, or 2)
$CA  =  read from a file
$CB  =  write to a file
$CC  =  close a file
$CE  =  position file marker
$65  =  bye, cause MacQForth to quit, use to quit the application
        from within an assembly language program
```

Note: Do not use "/" as the directory separator. Instead, use ":" which is the normal Mac separator. Remember that pathnames are stored as length/text. So, the pathname for the file "ABC" is stored (in decimal) as 3,65,66,67. (Minus commas, of course!)

Detailed command descriptions

Command parameters marked as _required_ are necessary for MacQForth, those marked as _ignored_ are not. Any value can be in the _ignored_ field. If a parameter is returned it is indicated as a (result) and space must be made for the value.

** Create a new file

command number $C0

parameters:

0	(number-of-parameters) (7)	_required_
+1	(pointer to pathname)	_required_
+3	(access code)	_ignored_
+4	(file type code)	_ignored_
+5	(auxilliary type code)	_ignored_
+7	(storage type)	_ignored_
+8	(date of creation)	_ignored_
+10	(time of creation)	_ignored_

** Destroy an existing file

command number $C1

parameters:

0	(number-of-parameters) (1)	_required_
+1	(pointer to pathname)	_required_

** Open an existing file

command number $C8

parameters:

0	(number-of-parameters) (3)	_required_
+1	(pointer to pathname)	_required_
+3	(pointer to i/o buffer)	_required_**
+5	(reference number, 0, 1, 2)	_required_ (result)

** MacQForth is trailored to running QForth. Therefore, you can use at most three files corresponding to reference numbers 0, 1, and 2. MacQForth determines which file QForth wants to use by the address of this buffer. The buffer itself is unused but it _must_ be one of the following addresses, in lo/hi format,

File 0 = 00:A6
File 1 = 00:A2
File 2 = 00:9E

** Read from an open file

command number $CA

parameters:

0	(number-of-parameters) (4)	_required_
+1	(file reference number)	_required_
+2	(pointer to data buffer)	_required_
+4	(requested number of bytes)	_required_
+6	(number actually read)	_required_ (result)

** Write to an open file

command number $CB

parameters:

0	(number-of-parameters) (4)	_required_
+1	(file reference number)	_required_
+2	(pointer to data buffer)	_required_
+4	(requested number of bytes)	_required_
+6	(number actually written)	_required_ (result)

** Close an open file

command number $CC

parameters:

0	(number-of-parameters) (1)	_required_
+1	(file reference number)	_required_

** Position file marker within an open file

command number $CE

parameters:

0	(number-of-parameters) (2)	_required_
+1	(file reference number)	_required_
+2	(file position, *3* bytes)	_required_

** Bye

command number $65

parameters:

0	(number-of-parameters) (4)	_ignored_
+1	(quit type code)	_ignored_
+2	(pointer to quit code)	_ignored_
+4	(a reserved value)	_ignored_
+5	(a reserved pointer)	_ignored_

Programming example

A simple programming example to create a new file named "ABC" and write some text to it. Also in FILE.S in the DEMO folder.

```
;
; MakeFile -- creates a file and writes some data. Ignores errors.
;

 *= $300

; create the file "ABC"

 lda #$01           ; setup for 'create'
 sta PARAMS
 lda #<NAME          ; low byte of name address
 sta PARAMS+1
 lda #>NAME         ; high byte of name address
 sta PARAMS+2
 lda #$C0           ; create command
 sta MLI+3          ; put in table
 jsr MLI            ; create the file

; open the file

 lda #$03
 sta PARAMS         ; adjust number of parameters, name already set
 lda #$00
```

```
 sta PARAMS+3
 lda #$A6
 sta PARAMS+4       ; use file 0
 lda #$C8           ; open command
 sta MLI+3
 jsr MLI            ; open the file

; write to the file

 lda #$04
 sta PARAMS
 lda PARAMS+5       ; get reference number returned by open
 sta PARAMS+1       ; and put in for write
 sta REF            ; and save for close
 lda #<STRING
 sta PARAMS+2       ; pointer to data
 lda #>STRING
 sta PARAMS+3
 lda #$05           ; number of bytes to write
 sta PARAMS+4
 lda #$00
 sta PARAMS+5
 lda #$CB           ; write command
 sta MLI+3
 jsr MLI            ; write the data

; close the file

 lda #$01
 sta PARAMS
 lda REF            ; put in reference number
 sta PARAMS+1
 lda #$CC           ; close command
 sta MLI+3
 jsr MLI            ; close the file
 rts                ; and end

; call MLI

MLI jsr $BF00       ; call ProDOS
 .byte $00          ; command number
 .word PARAMS       ; address of parameter table
 rts

; data

NAME .byte 3,"ABC"        ; name of the file with length
STRING .byte "Hello",0    ; data to write
REF .byte $00             ; ProDOS reference number
PARAMS .dbyt $0000        ; ProDOS parameter table
 .dbyt $0000
 .dbyt $0000
 .dbyt $0000
```