Sample FILEDEMO.4th File

This demonstration program is included in the CF Card Software distribution. For instructions on how to use the demo, see the comments at the end of this code listing. The distribution also contains a parallel file named FILEDEMO.C for C programmers.

```
\ FILEDEMO.4TH

\ See the "HOW TO RUN THE DEMO" section at the bottom of this file for instructions.
\ Be sure to define CF_MODULE_NUM so that it matches the hardware settings!

\ This file demonstrates how to use some of the common file manipulation functions
\ in the CF Card Software Package.
\ Both Forth (*.4th) and C (*.C) language versions of this FILEDEMO file exist.

\ The Demo function creates a file named "TESTER.QED" and writes to it a
\ sequence of bytes. It uses WPLUS_MODE to open the file, which means that if
\ the file already exists it is truncated to zero size, and that the file is
\ readable and writeable.  Using the pre-dimensioned File_Contents buffer,
\ we write increasing values from 0 to 255 in each of the first 256 bytes.
\ Then we store a message string in the file; it says:
\   " Alphabet in forward order: "
\ We then selectively read parts of the file using a buffer in
\ common RAM, printing the message followed by an upper case alphabetic listing.
\ Finally, we close the file.
\ This File I/O code demonstrates the following:
\ How to open and close a file;
\ How to use File_Set_Pos and File_Tell_Pos to control random file access;
\ How to use the pre-dimensioned File_Contents buffer in heap as a scratchpad;
\ How to use another buffer (we call it show_buffer) as a scratchpad;
\ How to use File_Write, File_Puts and File_Put_CRLF to put content into a file;
\ How to use File_Gets to fetch contents out of a file.

\ For clarity, very little error handling is performed in this code.
\ As an exercise for yourself, you can add more robust error handling
\ to the code.  In most cases, you can either report the returned error codes,
\ or simply check that the number of characters read or written
\ (as returned by the function) matches what you expect.
\ To report one overall code, you can logically OR the error codes of the
\ individual file operations.

\ Copyright 2002 Mosaic Industries, Inc.  All Rights Reserved.
\  Disclaimer: THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, WITHOUT ANY
\  WARRANTIES OR REPRESENTATIONS EXPRESS OR IMPLIED, INCLUDING, BUT NOT
\  LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS
\  FOR A PARTICULAR PURPOSE.

\ ****

\ You must load install.txt and LIBRARY.4TH containing the CF Card Software Drivers
\ before loading this file. Review the memory map statements at the top of the files
\ to ensure that there are no memory conflicts.


DECIMAL       \ interpret all numbers using decimal base
9 WIDTH !     \ avoid non-unique names

WHICH.MAP  0=
    IFTRUE  4 PAGE.TO.RAM      \ if in standard.map...
                               5 PAGE.TO.RAM
                               6 PAGE.TO.RAM
        DOWNLOAD.MAP
    ENDIFTRUE


ANEW DEMO_CODE  \ a marker to simplify reloading of this code
```

```
\ ****** Constants, Variables and Buffers ************

0   CONSTANT CF_MODULE_NUM      \ MUST match hardware jumper settings! Inits cf_module
256 CONSTANT NUM_ASCENDING_BYTES
60  CONSTANT SHOW_BUF_SIZE          \ bigger than we need
ASCII Z 1+ ASCII A - CONSTANT LETTERS_PER_ALPHABET

1024 32 * CONSTANT BYTES_PER_PAGE   \ each page is 32 Kbytes; see Make_Image_File

INTEGER: demo_file_id       \ used to save the file_id
DOUBLE:  string1_start      \ holds 32bit offset of the message string in file

SHOW_BUF_SIZE V.INSTANCE: show_buffer   \ allocate RAM buffer for Show_File below

\ ********* Demo code showing how to use the file I/O functions ***********

: Init_Contents_Buffer  ( file_id -- )
\ writes an ascending pattern starting at zero into the File_Contents buffer
   LOCALS{ &file_id }
   NUM_ASCENDING_BYTES 0
   DO  I
      I &file_id File_Contents C!
   LOOP
;

: Make_File ( -- )
\ Opens a file named TESTER.QED, transfers an ascending sequence of 256
\ bytes to the file, followed by a message string.
\ Initializes the global self-fetching variables demo_file_id and string1_start
   " TESTER.QED" COUNT WPLUS_MODE File_Open        ( -- file_id)
   DUP TO demo_file_id         \ save in global variable
   0<                         \ negative file_id means file_open failed
   IF CR ." File open failed!" CR
   ELSE                        \ if no error, continue...
      demo_file_id Init_Contents_Buffer   \ put ascending byte pattern
      0 demo_file_id File_Contents
      NUM_ASCENDING_BYTES U>D
      demo_file_id           ( xsrc_addr\d.numbytes\file_id -- )
      File_Write 2DROP       \ write buffer to file; ignore d.numbytes_written

      demo_file_id File_Tell_Pos TO string1_start     \ save string offset
      " Alphabet in forward order: " COUNT
      demo_file_id              ( xsource\max_chars\fileid -- )
      File_Puts DROP             \ write string -> file; ignore numchars_written
      demo_file_id File_Put_CRLF DROP     \ mark line end; drop error code
   ENDIF
;

: Show_File ( -- )
\ this function prints some of the contents of the file whose id
\ is in the demo_file_id variable.
\ First we print the message that starts at d.offset = string1_start,
\ and then we print the upper case alphabet which starts at
\ an offset equal to ASCII A (promoted to 32 bits).
   demo_file_id string1_start  File_Set_Pos    \ get "ascending" label
   DROP                               \ drop error flag
   show_buffer SHOW_BUF_SIZE demo_file_id  ( xdest\bufsize\fileid -- )
   File_Gets          ( -- numchars_read) \ terminates at linefeed character
   show_buffer ROT CR TYPE                \ type string1 (ends in CRLF)
   demo_file_id ASCII A U>D File_Set_Pos
   DROP                               \ drop error flag
   show_buffer LETTERS_PER_ALPHABET demo_file_id   ( xdest\bufsize\fileid -- )
   File_Gets              ( -- numchars_read ) \ ends when 26 letters are read
   show_buffer ROT TYPE    \ type ascending alphabet
   CR                   \ we need to explicitly add carriage return
      \ as an excercise: add a line showing the lower case alphabet!
   ;
```

```
: Demo  ( -- )  \ this is the demonstration function
\ Opens TESTER.QED, initializes it, and reports selected contents, then closes the file.
\ The optional first line relinks the names up to this point in this file.
\   [ LATEST ] 2LITERAL VFORTH X!  \ this line is optional; it makes names accessible
    CR ." Initializing File System..."
    CF_MODULE_NUM Set_CF_Module        \ must init before calling Init_File_System
    Init_File_System        ( error -- )
    IF CR ." Couldn't initialize file system!" CR
    ELSE
       CR ." Creating and initializing the TESTER.QED file..." CR
       Make_File
       Show_File
       demo_file_id File_Close DROP    \ drop error flag
    ENDIF
    ;

: Startup   ( -- )     \ calls the Demo function
\ this top-level function can be declared as a priority.autostart routine.
    CF_MODULE_NUM Set_CF_Module        \ must init before calling Init_File_System
\           Do_Autoexec    \ un-comment this if you want to use the autoexec.qed capability
    Demo
    ;

\ We can also set up a PRIORITY.AUTOSTART routine.
\ We'll make the Demo Program run on each subsequent startup:
\ CFA.FOR Startup PRIORITY.AUTOSTART

\ To remove, type NO.AUTOSTART from your terminal.

4 PAGE.TO.FLASH
5 PAGE.TO.FLASH
6 PAGE.TO.FLASH
STANDARD.MAP
SAVE           \ even after a cold restart, you can still call the demo function by typing  RESTORE

0 1 DP X!       \ this puts dictionary pointer in RAM so that interactive use
              \ of strings (as in file_open, file_type, etc.) works!

\  ************* HOW TO RUN THE DEMO USING FORTH ********************
\ Make sure that your QED Board or Panel-Touch Controller is communicating
\ with your PC, that the Wildcard Carrier Board is mounted, and that the
\ CF Module is installed.  Check that the module port and jumper settings
\ match the CF_MODULE_NUM defined in this program.  Plug a formatted
\ CF Card into the socket on the CF Module.

\ Follow these steps:
\ 1. Use QEDTERM to send to the QED Board
\    the INSTALL.TXT file and the LIBRARY.4TH file that contain the CF Card Software
\    drivers.  The former file need not be re-sent each time the application program
\    is downloaded.  The LIBRAY.4TH file must be sent with each download.
\ 2. Use QEDTERM to send this file to the QED Board.
\ 3. Type
\        Startup
\    from your terminal to run the demo program from this file.
\    You should see the following:
\        Initializing File System...
\        Creating and initializing the TESTER.QED file...
\
\        Alphabet in forward order:
\        ABCDEFGHIJKLMNOPQRSTUVWXYZ
\
\ 4. To automatically run the program upon each powerup and restart, type from the  terminal:
\        CFA.FOR STARTUP PRIORITY.AUTOSTART
\    This places the autostart pattern near the top of page 4 flash.
\    To remove the autostart, simply type
\        NO.AUTOSTART
\     from your terminal.  If you remove the autostart, and
\     if power has cycled and you want to re-run the demo, you will have to type
\        RESTORE
\    before typing Startup.  This relinks the names so that the operating
\    system can interactively recognize interactively typed names.
```