Sample FILEDEMO.C 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.4th for Forth programmers.

```
// FILEDEMO.C

// 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, only simple error handling is performed in this code.
// As an exercise for yourself, you can add more robust error handling.
// 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.

// ****

// We recommend that you type COLD before loading the compiled *.txt file.

#include <\mosaic\allqed.h>     // include all of the qed and C utilities
#include "library.c"    // this is a kernel extension file;
// assume it's in the source code directory; if not, edit the file specification


// ****** Constants, Variables and Buffers ************

#define CF_MODULE_NUM  0    // MUST match hardware jumper settings! Inits cf_module
#define NUM_ASCENDING_BYTES  256
#define SHOW_BUF_SIZE        60           // bigger than we need
#define LETTERS_PER_ALPHABET ('Z' + 1 - 'A')

#define BYTES_PER_PAGE       (1024 * 32)    // 1 page= 32 Kb; see Make_Image_File

static int demo_file_id;    // used to save the file_id
static long string1_start;  // holds 32bit offset of the message string in file
static long numbytes_accessed; // value returned by file_read or file_write; not checked
static int f_error;         // holds logical or of error codes; not checked

static char show_buffer[SHOW_BUF_SIZE];  // allocate buffer for Show_File below

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

_Q void Init_Contents_Buffer( int file_id )
// writes an ascending pattern starting at zero into the File_Contents buffer
// note that we can't embed the call to File_Contents in the parameter list of
//    StoreChar, as nested calls of _forth functions are not permitted.
{   int i;
    xaddr buf_xaddress;
    for( i=0; i < NUM_ASCENDING_BYTES; i++ )
    {   buf_xaddress = File_Contents( i, file_id);
        StoreChar( (char) i, buf_xaddress);
    }
}

_Q void Make_File( void )
// Opens a file named TESTER.QED, transfers an ascending sequence of 256
// bytes to the file, followed by a message string.
// Initializes the global variables demo_file_id and string1_start
{   xaddr source_xaddr;
    char* string_ptr = "Alphabet in forward order: ";
    int string_size = strlen(string_ptr);
    int demo_file_id = File_Open("TESTER.QED",THIS_PAGE,-1,WPLUS_MODE);
    if(demo_file_id < 0)        // negative file_id means file_open failed
        printf("\nFile open failed!\n");
    else                        // if no error, continue...
    {   Init_Contents_Buffer(demo_file_id);     // put ascending byte pattern
        source_xaddr = File_Contents(0, demo_file_id);
        numbytes_accessed =     // write buffer to file; we discard numbytes written
            File_Write( XADDR_TO_ADDR(source_xaddr),XADDR_TO_PAGE(source_xaddr),
                        (long) NUM_ASCENDING_BYTES, demo_file_id);

        string1_start = File_Tell_Pos(demo_file_id);  // save string offset
        numbytes_accessed =       // write string -> file; ignore numchars written
            File_Puts( string_ptr, THIS_PAGE, string_size, demo_file_id);

        f_error |= File_Put_CRLF(demo_file_id);    // mark line end; or error code
    }
}

_Q void Show_File( void )
// 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).
{   int numchars_read;
    f_error |= File_Set_Pos( demo_file_id, string1_start);  // get "ascending" label
    numchars_read = File_Gets(show_buffer,THIS_PAGE,SHOW_BUF_SIZE,demo_file_id);
    show_buffer[ numchars_read ] = 0;   // put terminating null
    printf("\n%s",show_buffer);           // type string1 (ends in CRLF)
    f_error |= File_Set_Pos(demo_file_id,(long) 'A');
    numchars_read =                  // ends when 26 letters are read
            File_Gets(show_buffer,THIS_PAGE,LETTERS_PER_ALPHABET,demo_file_id);
    show_buffer[ numchars_read ] = 0;   // put terminating null
    printf("%s\n",show_buffer);         // type ascending alphabet, add newline
        // as an excercise: add a line showing the lower case alphabet!
}
```

```
_Q void Demo( void )    // this is the demonstration function
// Initializes file system, Opens and initializes TESTER.QED file,
// reports selected contents, then closes the file.
{   f_error = 0;
    printf("\nInitializing File System...");
    Set_CF_Module( CF_MODULE_NUM );  // must init before calling Init_File_System
    if(Init_File_System())      // if error...
        printf("\nCouldn't initialize file system!\n");
    else                        // if no error...
    {   printf("\nCreating and initializing the TESTER.QED file...\n");
        Make_File();
        Show_File();            // print announcement and alphabet
        File_Close(demo_file_id);   // ignore error flag
    }
}


void main( void )   // sets up automated file system init and calls the Demo function
// this top-level function can be declared as a priority.autostart routine.
{  Set_CF_Module( CF_MODULE_NUM );   // must init before calling Init_File_System
//   Do_Autoexec();  // un-comment this if you want to use the autoexec.qed capability
   Demo();
}

/* ************* HOW TO RUN THE DEMO USING C ********************

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. Drag the LIBRARY.C and LIBRARY.H files that contain the CF Card software drivers
   to the same directory as your source code (or, if you choose to put these files
   in another location, edit the #include directives at the top of this file
   accordingly).
2. Use QEDTERM to send the INSTALL.TXT file from the CF Card software driver
   to the QED Board.  This loads the drivers onto the board.  You need not
   repeat this step when recompiling your C application program.
3. Compile this demo program by opening the file from WinEdit
   and pressing the hammer (make) icon.
4. Use QEDTERM to send the resulting FILEDEMO.TXT file to the QED Board.
5. Type
       MAIN
   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

6. You can optionally type
       CFA.FOR  MAIN  PRIORITY.AUTOSTART
   from your terminal to automatically run the program after every startup.

   To remove the autostart, type NO.AUTOSTART from your terminal.

*/
```