Using Macro Commands to Do File I/O
By Chris Chiang (4/8/2008)

File I/O Control Block

A File I/O Control Block is required for some of the file related operations such as “read”, “write”, and “append”. The elements of the block are described in C language in the following:

```
struct FileIoControlBlock {
    DWORD handle;       // A 32-bit word to store the handle of an opened file
    DWORD size;         // A 32-bit word to store the size (in byte) of the file
    char filename[81];  // A byte array to store the filename and the extension name of
                        // the opened file; the maximum length of the filename and the extension
                        // name is 80 characters
};
```

Opening File

Before you can read or write a file, you have to open it. There are two ways to open a file:

1) A “Select File” function button can open a file for you if the purpose is “Open to read”, “Open/create to write”, or ”Open/create to append”. This button brings up a file selection box for the operator to select a desired file. It will open the selected file and fill the specified File I/O Control Block.

2) An **OpenFile** macro command as shown below can open a file for you too.

p1 = OpenFile(p2,p3)

p1: File I/O Control Block; this block requires 4 internal words; if the “handle” is zero, it indicates the operation failed.

p2: Filename or full pathname; only ASCII characters are allowed

p3: Purpose: 0-read; 1-write; 2-append

[Example]

$U10 = “test.txt”

$U100 = OpenFile($U10, 0) // Open the file “test.txt” for the read operation

Reading File

The **ReadFile** macro command reads a specified number of bytes from an opened file.

p1 = ReadFile(p2, p3, p4)
p1: The actual number of bytes that are read by this command.
p2: File I/O Control Block
p3: Internal memory to receive the read data
p4: Number of bytes to be read

[Example]
$U200 = OpenFile($U100, $U150, 20) // Read 20 bytes from the file and put them in $U150
// and the following words

Writing File

The **WriteFile** macro command writes a specified number of bytes to an opened file.

p1 = WriteFile(p2, p3, p4)
p1: Result; 0-OK; otherwise: failed
p2: File I/O Control Block
p3: Internal memory that stores the data to be written to the file
p4: Number of bytes to be written

[Example]
$U200 = WriteFile($U100, $U150, 30) // Write 30 bytes to the file. The data to be output is
// stored in $U150 and the following words

Closing File

The **CloseFile** macro command closes an opened file. It is important to close a file after you have finished any operation with it.

p1 = CloseFile(p2)
p1: Result; 0-OK; otherwise: failed
p2: File I/O Control Block

[Example]

$U200 = CloseFile($U100)

Deleting File

The **DeleteFile** macro command deletes a specified file.

p1 = DeleteFile(p2)

p1: Result; 0-OK; otherwise: failed

p2: Filename or full pathname; only ASCII characters are allowed

[Example]

$U10 = “test.txt”

$U100 = DeleteFile($U10) // Delete the file “test.txt”.

Renaming File

The **RenameFile** macro command renames a specified file.

p1 = RenameFile(p2, p3)

p1: Result; 0-OK; otherwise: failed

p2: Filename or full pathname of the file to be renamed; Only ASCII characters are allowed.

p3: New filename; Full pathname is not allowed; Only ASCII characters are allowed.

[Example]

$U10 = “test.txt”

$U50 = “new.txt”

$U100 = RenameFile($U10, $U50) // Rename the file “test.txt” to “new.txt”

Getting Volume Information

The **GetVolInfo** macro command gets the size, the free size, and the volume name of a specified disk drive (USB memory stick).

p1 = GetVolInfo(p2, p3)

p1: Result; 0-OK; otherwise: failed

p2: Drive ID; 3 for drive C; 4 for drive D; 5 for drive E; 0 for the current drive

p3: Volume information buffer; the buffer receives the volume information as described in the following C structure. The buffer requires 21 16-bit words.

```
struct VolumeInfo {
    WORD name[16]; // A 16-bit word array to store the volume name.
                   // The volume name is a null terminated ASCII character string.
                   // The name can be up to 31 characters.
    DWORD size; // A 32-bit word to store the size of the drive. The unit is K bytes.
    DWORD freeSize; // A 32-bit word to store the free size of the drive. The unit is K bytes.
    WORD drive; // A 16-bit word to store the drive ID.
                // 0: current drive; 3: drive C; 4: drive D; 5: drive E
};
```

[Example]

```
$U100 = GetVolInfo(0, $U0) // Get the volume information of the current drive.
                           // The volume name is stored in $U0 through $U15.
                           // The size of the drive is stored in $U16 and $U17.
                           // The free size of the drive is stored in $U18 and $U19.
                           // The ID of the current drive is stored in $U20.
```

Restrictions

1) Only ASCII characters are allowed for a full pathname or a filename.
2) The maximum length of a full pathname or a filename is 80 characters.
3) The file extension name can only be 3 characters.
4) No dot is allowed in the filename.
5) In the file selection dialog box, you do not enter the file extension name. The predefined file extension name will be used for any file operation. The maximum length of the filename you can enter is 48 characters.