Application Note

File I/O Operations in Relay Ladder Logic with ASIC-100/200

ASIC-100/200 now supports reading and writing data structures to data files on the hard disk. It also performs file operations, including copying files over a network, using Relay Ladder Logic function blocks.

These File I/O function blocks are as follows:

FNEW (open new empty file),
FOPEN (open existing file),
FREAD (read a line from file into a data structure),
FWRIT (write a data structure to a file),
FCLOS (close an open file),
FDEL (delete a file),
FCOPY (copy an existing file to another file),
FRWND (reposition read pointer to top of open file).

Each of the File function blocks requires the name of a File Control Block (fcb). This File Control Block controls the file operations and reports status as to the progress and outcome of the file operations. When the fcb name is first used in the program, a File Control Block is created with that name. You can have as many File Control Blocks as you wish, but each File Control Block performs only one file operation at a time. Make sure that the fcb name that you use does not conflict with other local or global variable names in your program.

The fcb provides several flags and variables that can be used in the program to determine the outcome of the file operation:

fcb.BUSY	is a BOOL that indicates that the fcb is busy performing an operation,
fcb.OPEN	is a BOOL that indicates that the file has been opened successfully,
fcb.RDN	is a BOOL that indicates that a read operation has been competed,
fcb.WDN	is a BOOL that indicates that a write operation has been completed,
fcb.CLSD	is a BOOL that indicates that the file has been closed,
fcb.EOF	is a BOOL that indicates the End Of File has been seen during reading,
fcb.EFLAG	is a BOOL that indicates a file operation error has occurred
fcb.ERR	is an INT that contains an error code for the operation.

To perform any reading or writing to a data file, the file must first be opened, the read and/or write operations performed, and then the file must be closed. The FNEW operation will create a new file, with the name specified in the File Name list box, if a file with that file name does not exist, or if one does exist it will open it and delete the contents. FOPEN will open an existing file with the specified File Name, or fail if a file with that name is not found. The File Name can be any string variable or constant. If the File Name does not contain the entire path to the file, the path to the current project directory will be prefixed to it automatically by the fcb. The FCLOS function block is used to close an open file. FCLOS does not require a filename because it takes it from the specified fcb.

FREAD and FWRIT take the name of a data structure (STRUCT) variable to read into or to write from. FREAD and FWRIT also specify strings of characters to be used between structure members (field separator), to delimit STRING values (string delimiter), and to be used at the end of the line (end of line terminator). These format strings are stored in the fcb and can be changed by the FREAD and FWRIT commands. The default values for these strings are: " " (space), "$"" (double quote), "$n" (newline) respectively. Other commonly used values are "$t" (tab), "," (comma), "$'" (single quote) and "$r" (carriage return).

The FRWND function block can be used to reposition the internal file position pointer to the beginning of the file, so that the next FREAD operation will begin from the start of the file. This operation is automatically performed when the file is opened.

A file can be deleted using the FDEL function block. The name of the file to be deleted must be specified.

A file can be copied to another file name or to another place on the network using the FCOPY function block. The name of the file to be copied from is specified in the Source File Name string, and the name of the file to be copied to is specified in the Destination File Name string. If you want to copy a file to another directory or somewhere else on the network, specify the path of the destination in the Destination File Name.

Each of the File I/O function blocks may fail for some reason or other. If this happens, the fcb.EFLAG boolean is set, and the fcb.ERR integer receives a value indicating the source of the problem before the fcb.BUSY flag is cleared. These values include the following:

FILE_NO_ERROR	0
FILE_FCB_BUSY	15
NO_FILENAME_SPECIFIED	16
FILE_NOT_OPENED	17
FILE_NOT_FOUND	18
DISK_FULL	19
READ_FAILED	20
FILE_COPY_FAILED	21
WRITE_FAILED	22
EOL_UNEXPECTED	23
EOF_UNEXPECTED	24

For more details on these error values, consult the Program Editor help file for File I/O functions.

Only one file operation can be performed for each File Control Block at a time. Each time a File I/O function block is activated, the fcb.ERR is set to 0, the fcb.EFLAG is set to FALSE and the fcb.BUSY bit is set to TRUE. When the file operation is complete, fcb.ERR value and the fcb.EFLAG will be set appropriately and the fcb.BUSY flag will be set to FALSE. It is up to you to make sure that a file operation is not activated on a fcb when the fcb.BUSY bit is TRUE, otherwise a program fault will occur.

You can use the fcb.EFLAG to test to determine if a file operation error has occurred and fcb.ERR to determine what kind of error is indicated. The fcb.EFLAG and fcb.ERR are reset whenever a file function block is activated.

You also should construct your RLL program so that the File I/O operations are called one at a time and in the proper sequence. Also, make sure that all the File I/O function blocks that operate on the same file use the same Function Control Block name. An example of File I/O function block usage and error reporting and recovery are shown in the following figures.

Example Program

Assume that you have used the Symbol Manager to define two data structure types similar to those shown in Figure 1. Assume also that you have defined two data structure variables a and b as shown. If you create an RLL program as shown in Figure 2 and run it, the resulting contents of file test.dat will be as follows:

```
1995-09-18 16:24:03 TRUE 5 "hello"
1995-09-18 16:24:03 FALSE 123 "hello there" 234
```

This sample program uses the default file format strings, which you can modify if you wish**.**

```
TYPE
        struct1 : STRUCT
                d : DATE;
                t : TOD;
                x : BOOL;
                i : INT;
                s : STRING;
        END_STRUCT;
        struct2 : STRUCT
                c : struct1;
                i : INT;
        END_STRUCT;
END_TYPE
VAR_GLOBAL
        a : struct1;
        b : struct2;
END_VAR
```

Figure 1

Detailed Description of Figure 2

Rung 1: On the **rising edge** of **doit** the **done** flag is reset, the data structures values are loaded using **MOVE** function blocks and the **initialized** flag is set.

Rung 2: When **doit** goes low the **initialized**, **written1** and **written2** flags are reset. If **fcb1.OPEN** is true indicating that the file was somehow left open, when .**BUSY** is low the file will be closed. If the .**EFLAG** is TRUE indicating that an error occurred somewhere, the **override** flag is set so that the FNEW function block can be called even though the .EFLAG is set in order to clear the error and retry the operation.

Rung 3: If the data structures have been **initialized** and the file is not **open** and the task is not **done**, if the .**EFLAG** is not set or the **override** is set and the fcb is not **busy** then a new file is opened and **written1** and **written2** are cleared. If the **override** flag is reset, it will be cleared.

Rung 4: If the file is **open** and **written1** is not set, if no error has occurred (.**EFLAG**) and the fcb is not **busy**, then data structure a will be written to the file and **written1** will be set.

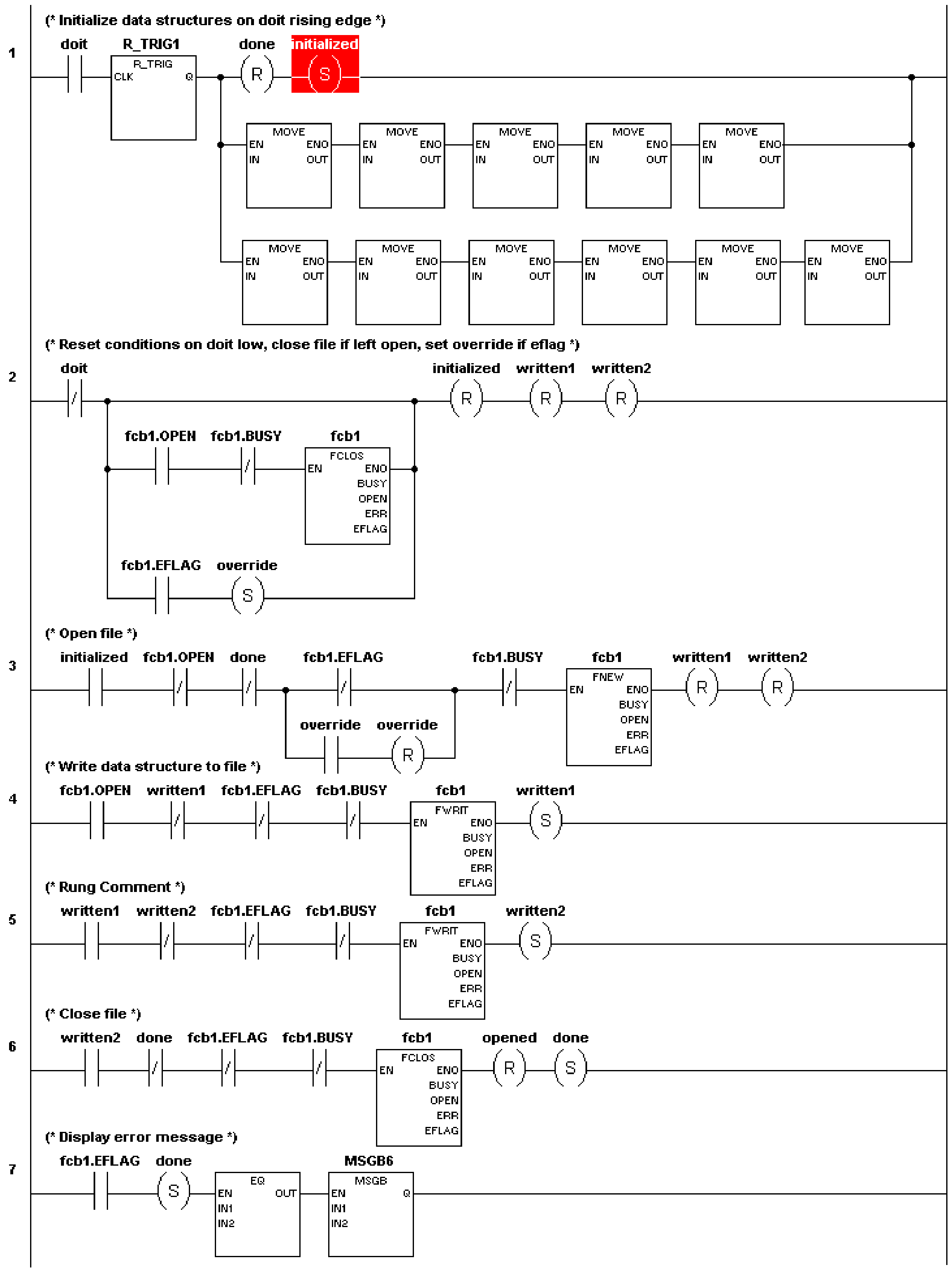
Rung 5: If **written1** is set and **written2** is not set, if no error has occurred (.**EFLAG**) and the fcb is not **busy**, then data structure b will be written to the file and **written2** will be set.

Rung 6: If **written2** is set and **done** is not set, if no error has occurred (.**EFLAG**) and the fcb is not **busy** the file will be closed, the **done** will be set. The **fcb1.OPEN** flag will be reset by the **FCLOS** operation.

Rung 7: If the **EFLAG** is set, the **done** flag will be set. If **fcb.ERR** is equal to 17, then a message box is displayed to indicate that the file could not be opened. Other error messages could be added to this rung.

Rung 3 is disabled by **.EFLAG** so that rung 7 can be run before the **FNEW** is performed again, which would clear the .**EFLAG**. Rung 7 sets **done** to disable rung 3 until rung 2 is executed when **doit** goes low and rung 1 is executed when **doit** goes high again and the **done** flag is cleared.

Notice that **all** File I/O function blocks check if the .**BUSY** flag is low before activating. Notice also that almost all File I/O function blocks check and set flags (e.g. written1) to make sure that the operation is done only once and in the proper sequence relative to other file operations. Further note that most file operations use the .EFLAG to disable the function if an error has occurred.



(* Initialize data structures on doit rising edge *)
1
doit
R_TRIG1
R_TRIG
CLK
Q
done
initialized
R
S
MOVE
EN
ENO
IN
OUT
(* Reset conditions on doit low, close file if left open, set override if eflag *)
2
doit
initialized
written1
written2
fcb1.OPEN
fcb1.BUSY
fcb1
FCLOS
BUSY
OPEN
ERR
EFLAG
fcb1.EFLAG
override
(* Open file *)
3
initialized
fcb1.OPEN
done
fcb1.EFLAG
fcb1.BUSY
fcb1
FNEW
written1
written2
override
override
(* Write data structure to file *)
4
fcb1.OPEN
written1
fcb1.EFLAG
fcb1.BUSY
fcb1
FWRIT
written1
(* Rung Comment *)
5
written1
written2
fcb1.EFLAG
fcb1.BUSY
fcb1
FWRIT
written2
(* Close file *)
6
written2
done
fcb1.EFLAG
fcb1.BUSY
fcb1
FCLOS
opened
done
(* Display error message *)
7
fcb1.EFLAG
done
EQ
IN1
IN2
MSGB6
MSGB

Figure 2