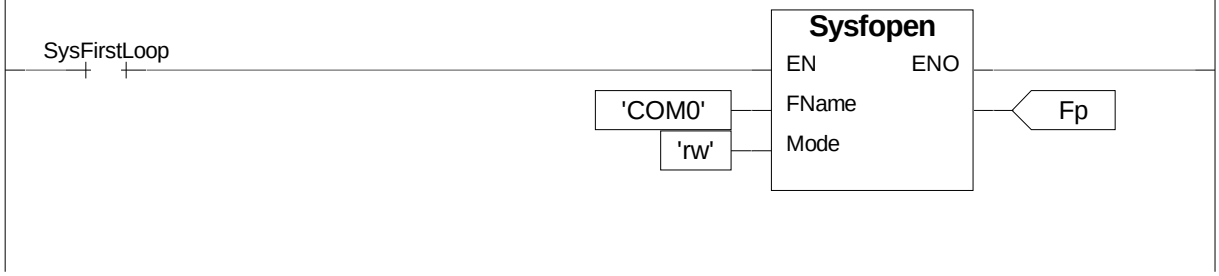
```
VAR_GLOBAL
Start AT %IX0.0 : BOOL;
END_VAR
```

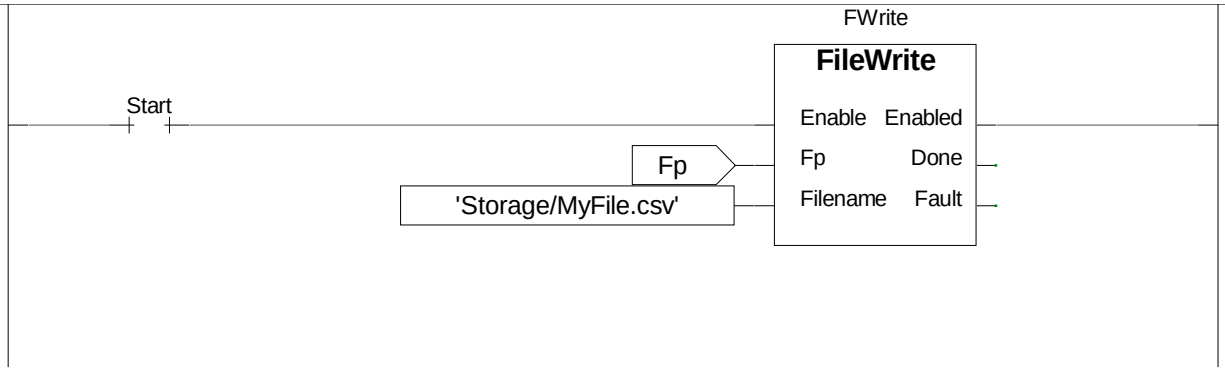
PROGRAM MProgram

```
VAR
FWrite : FileWrite; (* FileWrite FB instance *)
Fp : FILEP; (* Serial file pointer *)
END_VAR
```

1



2



	Project : SerialFileWrite	
	PROGRAM : MProgram	
	Release :	Ver :1.00
	Author :	Date:06/11/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK FileWrite

(SFR065A100) Dumps and retrieves memory on file

```
VAR_INPUT
Enable : BOOL; (* Enable to execution *)
Fp : FILEP; (* File pointer *)
Filename : STRING[ 32 ]; (* Path and name of file where to log *)
END_VAR

VAR_OUTPUT
Enabled : BOOL; (* Function block enabled *)
Done : BOOL; (* File written *)
Fault : BOOL; (* Execution fault *)
END_VAR

VAR_EXTERNAL
SysActTaskID : USINT; (* Current task ID *)
END_VAR

VAR
IFp : FILEP; (* Internal file pointer *)
FPos : DINT; (* File position *)
CaseNr : USINT; (* Case number *)
i : INT; (* Auxiliary variable *)
Ch : INT; (* Received character *)
END_VAR
```

```
 1 (* ************************************************************************ *)
 2 (* FUNCTION BLOCK "FileWrite"                                               *)
 3 (* ************************************************************************ *)
 4 (* This FB writes all data received from a data stream (i.e. a serial port)   *)
 5 (* to a file. There are a trigger and a stop characters, that starts and      *)
 6 (* stops the data writing.                                                  *)
 7 (*                                                                          *)
 8 (* Input:                                                                   *)
 9 (* Enable: Enable FB to run. If FALSE, do nothing.                          *)
10 (* Read: Comando read file di dump in memoria.                              *)
11 (* Filename: Full path name of the file (es.: 'Storage/MyFile.txt').        *)
12 (*                                                                          *)
13 (* Output:                                                                  *)
14 (* Enabled: FB enabled to operate.                                          *)
15 (* Done: Set for a program loop at the file write end.                      *)
16 (* Fault: Set for a program loop on error.                                  *)
17 (* ------------------------------------------------------------------------ *)
18
19     (* -------------------------------------------------------------------- *)
20     (* INITIALIZATIONS                                                      *)
21     (* -------------------------------------------------------------------- *)
22     (* Fault management, if fault, aborts execution. *)
23
24     IF (Fault) THEN Fault:=FALSE; CaseNr:=0; END_IF;
25
26     (* Resets the one shot bits. *)
27
28     IF (Done) THEN Done:=FALSE; CaseNr:=0; END_IF;
29
30     (* This FB can only be executed in the background task. *)
31
32     IF (SysActTaskID <> ID_TASK_BACK) THEN Fault:=TRUE; RETURN; END_IF;
```

```
33
34      (* Check if the FB is enabled. *)
35
36      IF NOT (Enable) THEN Enabled:=FALSE; CaseNr:=0; RETURN; END_IF;
37      Enabled:=TRUE; (* Function block enabled *)
38
39      (* ---------------------------------------------------------------------- *)
40      (* PROGRAM CASES MANAGEMENT                                               *)
41      (* ---------------------------------------------------------------------- *)
42      (* Program cases management. *)
43
44      WHILE (TRUE) DO
45          CASE (CaseNr) OF
46
47              (* ------------------------------------------------------------- *)
48              (* WAITING THE TRIGGER CHARACTER                                 *)
49              (* ------------------------------------------------------------- *)
50              (* Waiting the trigger character "^". *)
51
52              0:
53              IF NOT(TO_BOOL(SysGetIChars(Fp))) THEN RETURN; END_IF;
54              IF (Sysfgetc(Fp) <> 16#5E) THEN RETURN; END_IF;
55              CaseNr:=10; (* Case number *)
56
57              (* ------------------------------------------------------------- *)
58              (* FILE WRITING MANAGEMENT                                       *)
59              (* ------------------------------------------------------------- *)
60              (* Checks the file length, if file is present it will be deleted. *)
61              (* We always start with an empty file that contains the data. *)
62
63              10:
64              IF (Sysfilelength(Filename) > 0) THEN
65                  IF NOT(Sysremove(Filename)) THEN
66                      Fault:=TRUE; (* Execution fault *)
67                      RETURN;
68                   END_IF;
69              END_IF;
70              CaseNr:=CaseNr+1; (* Case number *)
71
72              (* ------------------------------------------------------------- *)
73              (* Checks if any characters is been received. *)
74
75              11:
76              IF NOT(TO_BOOL(SysGetIChars(Fp))) THEN RETURN; END_IF;
77              Ch:=Sysfgetc(Fp); (* Received character *)
78
79              (* Check if is the end character "$". *)
80
81              IF (Ch = 16#24) THEN Done:=TRUE; RETURN; END_IF;
82
83              (* Opens the  file in append. *)
84
85              IFp:=Sysfopen(Filename, 'a'); (* Internal file pointer *)
86              IF (IFp = NULL) THEN Fault:=TRUE; RETURN; END_IF;
87
88              (* Write received character on a file. *)
89              (* On error close the file. *)
90
91              IF (Sysfputc(Ch, IFp) <> Ch) THEN
92                  Fault:=TRUE; (* Execution fault *)
```

```
 93                i:=Sysfclose(IFp); (* Eseguo chiusura file *)
 94                RETURN;
 95             END_IF;
 96
 97             (* Close the file. *)
 98
 99             IF (Sysfclose(IFp) = EOF) THEN Fault:=TRUE; RETURN; END_IF;
100
101             (* ------------------------------------------------------------ *)
102             (* CASE NUMBER ERROR                                            *)
103             (* ------------------------------------------------------------ *)
104             (* Manage the case number error. *)
105
106         ELSE
107             Fault:=TRUE; (* Execution fault *)
108         END_CASE;
109     END_WHILE;
110
111 (* [End of file] *)
112
113
```