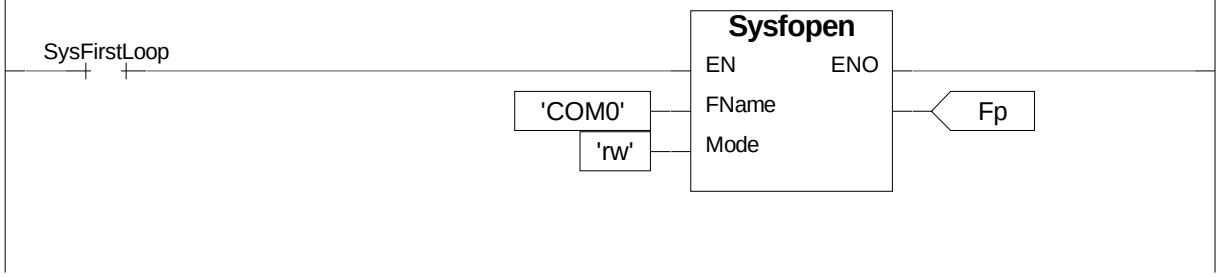
```
VAR_GLOBAL
Do00M00 AT %QX0.0 : BOOL;
Do01M00 AT %QX0.1 : BOOL;
TAGRdOk : BOOL; (* TAG read *)
TAGID : ARRAY[ 0..7 ] OF BYTE; (* TAG ID code *)
Do00CPU : BOOL; (* Output 00 CPU module *)
Do01CPU : BOOL; (* Output 01 CPU module *)
Di00CPU : BOOL; (* Input 00 CPU module *)
Di01CPU : BOOL; (* Input 01 CPU module *)
END_VAR
```

PROGRAM TAGReader

```
VAR
Fp : FILEP; (* File pointer *)
OWire : sOWireMng; (* One-Wire management FB *)
TAGRd : sOWRdIdentifier; (* One-Wire read identifier FB *)
OwID : UDINT; (* One-Wire ID *)
RdDone : BOOL; (* TAG read *)
END_VAR
```

1

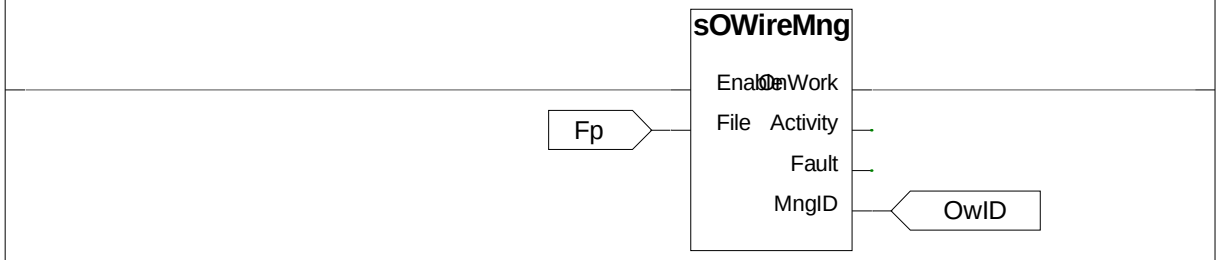
Open the serial port




2

Manage the Serial/One-Wire converter

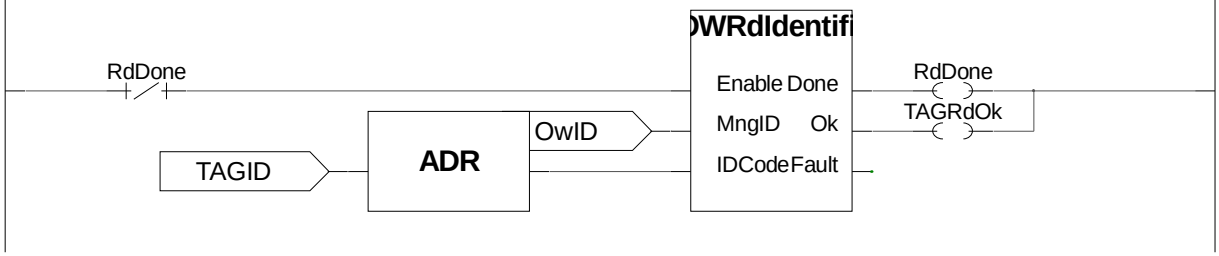
OWire




3

Read the TAG ID code

TAGRd



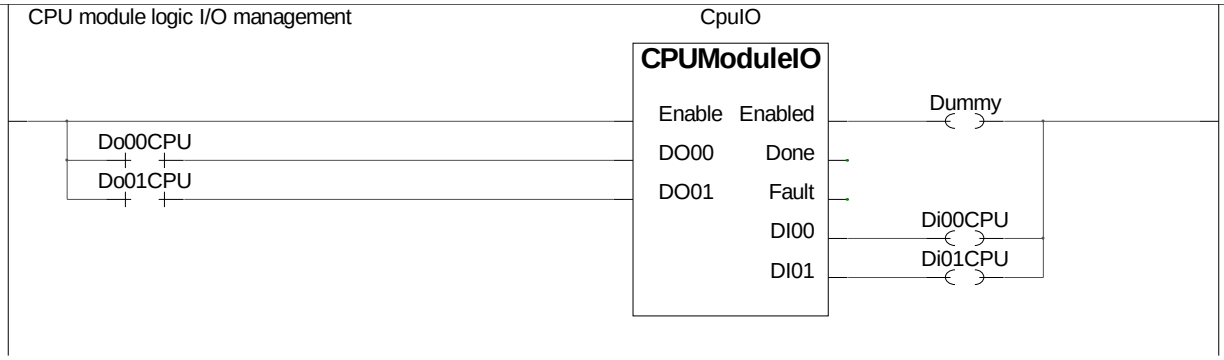

| | | |
|---|---|---|
| | Project : TagReader | |
| | PROGRAM : TAGReader | |
| | Release : TagReader | Ver :1.00 |
| | Author : | Date:19/05/2012 |
| | Note : | Page:1 of 1 |

PROGRAM Logic

```
VAR
CpuIO : CPUModuleIO; (* CPU logic I/O management FB *)
Dummy : BOOL; (* Dummy variable *)
SearchOn : BOOL; (* Search CLIP on a file *)
TAGFound : BOOL; (* TAG found in the file *)
DoorNr : USINT; (* Door to open *)
DoorCmdDelay : eTON;
TAGFind : FindTAGOnFile;
END_VAR
```

1

CPU module logic I/O management



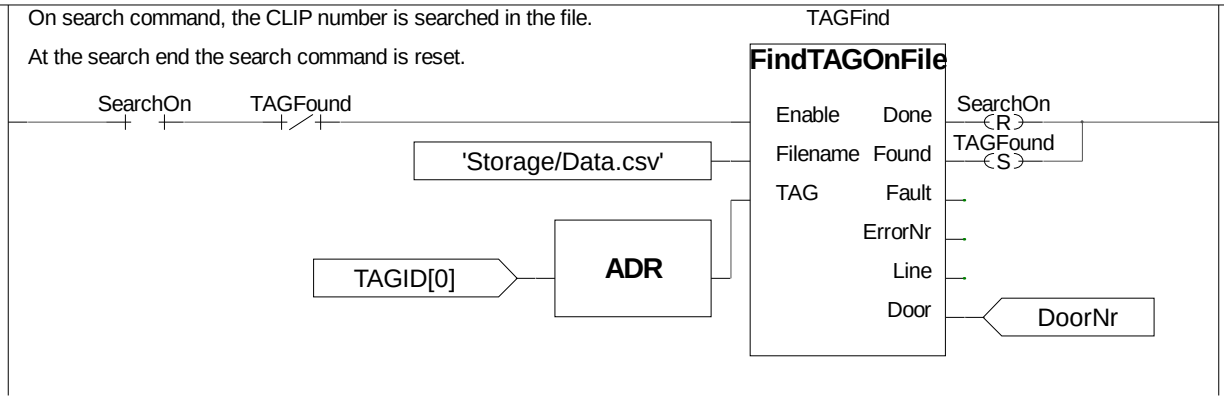

2

On CLIP detection (A call is arrived) the search sequence is started.


3

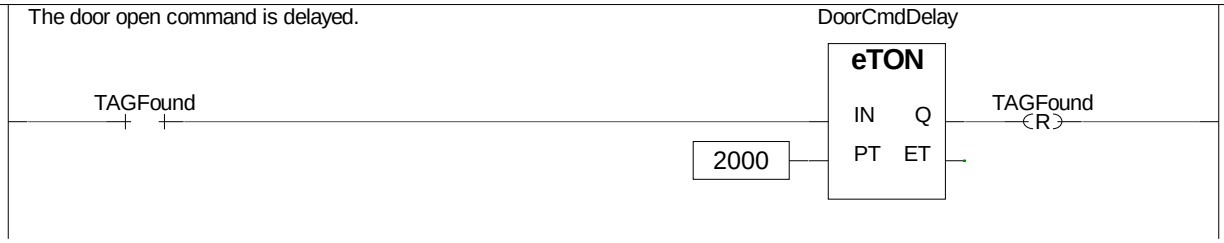
On search command, the CLIP number is searched in the file.

At the search end the search command is reset.




4

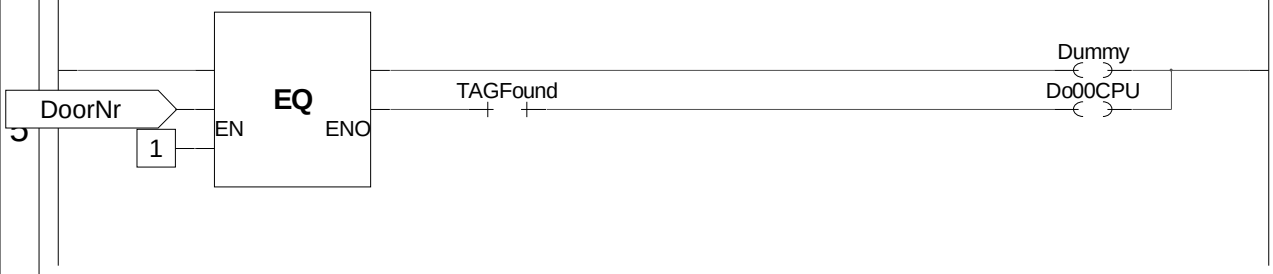
The door open command is delayed.




| | | |
|---|---|---|
| | Project : TagReader | |
| | PROGRAM : Logic | |
| | Release : TagReader | Ver :1.00 |
| | Author : | Date:19/05/2012 |
| | Note : | Page:1 of 2 |

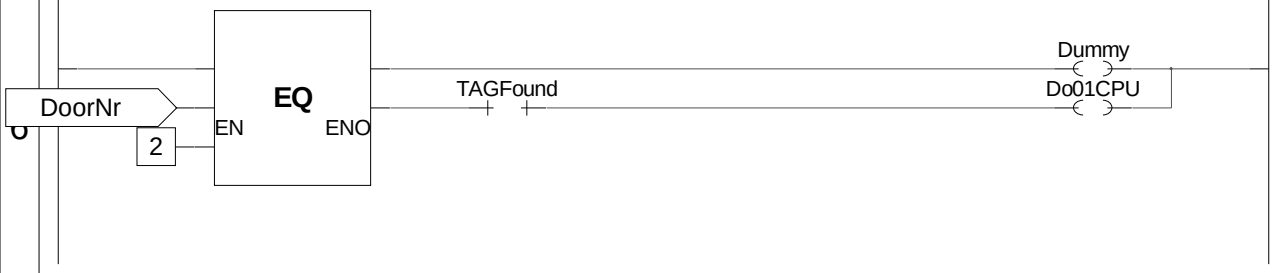
5

Command the logic output to open door Nr 1.




6

Command the logic output to open door Nr 2.




| | | |
|---|---|---|
| | Project : TagReader | |
| | PROGRAM : Logic | |
| | Release : TagReader | Ver :1.00 |
| | Author : | Date:19/05/2012 |
| | Note : | Page:2 of 2 |

```
VAR_INPUT
Enable : BOOL := FALSE; (* FB enable *)
Filename : STRING[ 32 ]; (* Path and name of file *)
TAG : @BYTE; (* TAG to find *)
END_VAR

VAR_OUTPUT
Done : BOOL; (* Execution done *)
Found : BOOL := FALSE; (* Code found *)
Fault : BOOL; (* Execution fault *)
ErrorNr : USINT; (* Error number *)
Line : UDINT; (* File line where FB ends *)
Door : USINT; (* Door number *)
END_VAR

VAR
Enabled : BOOL; (* FB enabled *)
FPos : UDINT; (* File position *)
FString : STRING[ 64 ]; (* Find string *)
Fp : FILEP; (* File pointer *)
i : INT; (* Auxiliary var *)
j : UDINT; (* Auxiliary var *)
Ptr : @USINT; (* General pointer *)
AString : STRING[ 32 ]; (* Auxiliary string *)
BString : STRING[ 32 ]; (* Auxiliary string *)
END_VAR
```

```
 1 (* ********************************************************************** *)
 2 (* FUNCTION BLOCK "FindTAGOnFile"                                         *)
 3 (* ********************************************************************** *)
 4 (* Questo blocco funzione esegue ricerca in un file CSV del TAG indicato.     *)
 5 (* Viene ritornato il numero di riga in cui il TAG è stato trovato ed il      *)
 6 (* numero di porta che può aprire.                                        *)
 7 (*                                                                        *)
 8 (* Il file CSV deve essere del tipo:                                      *)
 9 (* TAG;              Door<CR><LF>                                         *)
10 (* 1234567812345678; 1<CR><LF>                                            *)
11 (* 8765432187654321; 2<CR><LF>                                            *)
12 (*                                                                        *)
13 (* I parametri in ingresso sono:                                          *)
14 (* Enable: Attivandolo viene eseguita la ricerca del TAG.                 *)
15 (* Filename: Percorso e nome del file in cui eseguire la ricerca.         *)
16 (* TAG: TAG da ricercare.                                                 *)
17 (*                                                                        *)
18 (* I parametri in uscita sono:                                            *)
19 (* Done: Ricerca terminata.                                               *)
20 (* Found: TAG trovato (One shot).                                         *)
21 (* Fault: Errore nella ricerca (One shot).                                *)
22 (* ErrorNr: Numero errore riscontrato.                                    *)
23 (* Line: Linea del file in cui termina controllo.                         *)
24 (* Door: Numero di porta abilitata.                                       *)
25 (* ---------------------------------------------------------------------- *)
26
27     (* ---------------------------------------------------------------- *)
28     (* ABILITAZIONE                                                     *)
29     (* ---------------------------------------------------------------- *)
30     (* Eseguo controllo se FB abilitata. *)
```

```
31
32      IF NOT(Enable) THEN Enabled:=FALSE; Done:=FALSE; Found:=FALSE; Fault:=FALSE; RETURN; END_IF;
33
34      (* Eseguo inizializzazioni su abilitazione. *)
35
36      IF NOT(Enabled) THEN
37          Enabled:=TRUE; (* FB enabled *)
38          Line:=0; (* File line where FB ends *)
39          ErrorNr:=0; (* Error number *)
40          Door:=0; (* Door number *)
41          FPos:=0; (* File position *)
42      END_IF;
43
44      (* Se attivo Done esco, occorre disabilitare e poi riabilitare l'FB per *)
45      (* eseguire una nuova ricerca. *)
46
47      IF (Done) THEN RETURN; END_IF;
48
49      (* ------------------------------------------------------------------- *)
50      (* INIZIALIZZAZIONI                                                    *)
51      (* ------------------------------------------------------------------- *)
52      (* Apertura file in read. Se non esiste errore, il file rimane aperto *)
53      (* solo durante l'esecuzione della FB, uscendo viene sempre chiuso. *)
54
55      Fp:=Sysfopen(Filename, 'r'); (* File pointer *)
56      IF (Fp = NULL) THEN
57          Fault:=TRUE; (* Execution fault *)
58          Done:=TRUE; (* Execution done *)
59          ErrorNr:=10; (* Error number *)
60          RETURN;
61      END_IF;
62
63      (* ------------------------------------------------------------------- *)
64      (* ESEGUO LETTURA RIGA DA FILE                                         *)
65      (* ------------------------------------------------------------------- *)
66      (* Imposto posizione in cui leggere nel file. *)
67
68      IF (Sysfseek(Fp, TO_DINT(FPos), ID_SEEK_SET) = EOF) THEN
69          Fault:=TRUE; (* Execution fault *)
70          Done:=TRUE; (* Execution done *)
71          ErrorNr:=20; (* Error number *)
72          i:=Sysfclose(Fp); (* Eseguo chiusura file *)
73          RETURN;
74      END_IF;
75
76      (* Leggo dal file stringa di dimensione sufficente per contenere almeno *)
77      (* una riga, su fine file funzione torna meno caratteri dei richiesti. *)
78
79      i:=Sysfread(ADR(FString), 1, 64, Fp); (* Received characters *)
80      j:=Sysfclose(Fp); (* Eseguo chiusura file *)
81
82      (* Se non sono letti dati il file è terminato. *)
83
84      IF (i = 0) THEN Done:=TRUE; RETURN; END_IF;
85
86      (* Ho letto sicuramente almeno una riga. La riga termina con <CR><LF> *)
87      (* controllo se presenti i terminatori di riga. *)
88
89      j:=FIND(FString, '$r$n');
90
```

```
 91     IF (j = 0) THEN
 92         Fault:=TRUE; (* Execution fault *)
 93         Done:=TRUE; (* Execution done *)
 94         ErrorNr:=30; (* Error number *)
 95         RETURN;
 96     END_IF;
 97
 98     (* Inserisco codice tappo dopo ultimo carattere di riga, sul <CR>. *)
 99
100     Ptr:=ADR(FString)+j; (* General pointer *)
101     @Ptr:=0; (* Inserisco codice tappo *)
102
103     (* Posiziono file su inizio riga seguente, +1 per puntare dopo <CR><LF>. *)
104
105     FPos:=FPos+j+1; (* File position *)
106     Line:=Line+1; (* File line where FB ends *)
107
108     (* ---------------------------------------------------------------------- *)
109     (* ESEGUO CONTROLLO TAG                                                   *)
110     (* ---------------------------------------------------------------------- *)
111     (* Eseguo acquisizione TAG. Se funzione non trova stringa di TAG non *)
112     (* segnalo errore, per eseguire skip di campi vuoti o non numerici. *)
113
114     IF NOT(SysVarsscanf(ADR(FString), '%s', STRING_TYPE, ADR(AString))) THEN RETURN; END_IF;
115
116     (* Converto valore TAG in stringa. *)
117
118     FOR j:=0 TO 7 DO
119         i:=SysVarsnprintf(ADR(BString)+(j*2), 2+1, '%02X', USINT_TYPE, TAG+j);
120     END_FOR;
121
122     (* Eseguo verifica se CLIP cercato. *)
123
124     IF (FIND(AString, BString) = 0) THEN RETURN; END_IF;
125
126     (* TAG trovato, ritorno numero porta, cerco separatore. *)
127
128     j:=FIND(FString, ';');
129
130     IF (j = 0) THEN
131         Fault:=TRUE; (* Execution fault *)
132         Done:=TRUE; (* Execution done *)
133         ErrorNr:=40; (* Error number *)
134         RETURN;
135     END_IF;
136
137     (* Copio numero porta su variabile di ritorno. *)
138
139     IF NOT(SysVarsscanf(ADR(FString)+j, '%d', USINT_TYPE, ADR(Door))) THEN
140         Fault:=TRUE; (* Execution fault *)
141         Done:=TRUE; (* Execution done *)
142         ErrorNr:=50; (* Error number *)
143         RETURN;
144     END_IF;
145
146     (* Trovata voce in file per comandare l'apertura porta. *)
147
148     Found:=TRUE; (* Code found *)
149     Done:=TRUE; (* Execution done *)
150
```

```
151 (* [End of file] *)
152
153
```

FUNCTION_BLOCK CPUModuleIO

(SFR054B000) Manages the logic I/O on the CPU module
ENCRYPTED CODE

```
VAR_INPUT
Enable : BOOL; (* Function enable *)
DO00 : BOOL; (* Digital output 0 *)
DO01 : BOOL; (* Digital output 1 *)
END_VAR

VAR_OUTPUT
Enabled : BOOL; (* Function enabled *)
Done : BOOL; (* Function done *)
Fault : BOOL; (* Function fault *)
DI00 : BOOL; (* Digital input 0 *)
DI01 : BOOL; (* Digital input 0 *)
END_VAR
```

1

| | | |
|---|---|---|
| | Project : TagReader | |
| | FUNCTION BLOCK : CPUModuleIO | |
| | Release : TagReader | Ver :1.00 |
| | Author : | Date:19/05/2012 |
| | Note : | Page:1 of 1 |

FUNCTION_BLOCK eTON

(SFR053A500) On delay timer
ENCRYPTED CODE

```
VAR_INPUT
IN : BOOL; (* Timer input *)
PT : UDINT; (* Preset time value (mS) *)
END_VAR

VAR_OUTPUT
Q : BOOL; (* Delayed output *)
ET : UDINT; (* Executing time (mS) *)
END_VAR

VAR_EXTERNAL
SysTime : UDINT; (* System time (mS) *)
END_VAR
```

1

| | | |
|---|---|---|
| | Project : TagReader | |
| | FUNCTION BLOCK : eTON | |
| | Release : TagReader | Ver :1.00 |
| | Author : | Date:19/05/2012 |
| | Note : | Page:1 of 1 |