FUNCTION_BLOCK WT2xxAcquisition

```
VAR_INPUT
File : FILEP; (* File pointer *)
SpyOn : BOOL; (* Spy On *)
Timeout : UDINT; (* Timeout time (mS) *)
END_VAR

VAR_OUTPUT
Voltage : REAL; (* Voltage (V) *)
Current : REAL; (* Current (A) *)
Power : REAL; (* Power (W) *)
Frequency : REAL; (* Frequency (Hz) *)
END_VAR

VAR
CaseNr : USINT; (* Case programma *)
TimeBf : UDINT; (* Time buffer (uS) *)
RxData : STRING[ 256 ]; (* Buffer ricezione *)
Ptr : @BYTE; (* Auxiliary pointer *)
i : INT; (* Auxiliary counter *)
Fault : BOOL; (* Execution fault *)
Done : BOOL; (* Execution done *)
END_VAR
```

```
 1 (* ********************************************************************** *)
 2 (* FUNCTION BLOCK "WT2xxAcquisition"                                      *)
 3 (* ********************************************************************** *)
 4 (* Questo programma esegue l'acqisizione dei dati da un misuratore di energia *)
 5 (* VT210 Yokogawa connesso alla porta seriale.                            *)
 6 (* I parametri di comunicazione sono: 9600, N, 8, 1                       *)
 7 (*                                                                        *)
 8 (* Ecco il printout della comunicazione. Ogni riga termina con un <CR>.   *)
 9 (* V  1N  0388.91E+0,V  3N  0223.64E+0,V  4N  0306.27E+0                  *)
10 (* A  1N  0000.00E-3,A  3N  0097.89E-3,A  4N  0048.94E-3                  *)
11 (* W  1N  000.063E+0,W  3N  021.294E+0,W  4N  0021.36E+0                  *)
12 (* HzV3N  050.002E+0,A  3N  0097.89E-3                                    *)
13 (* END                                                                    *)
14 (*                                                                        *)
15 (* I valori da acquisire sono:                                            *)
16 (* 3N  0223.67E+0,V                                                       *)
17 (* 3N  0097.94E-3,A                                                       *)
18 (* 3N  021.299E+0,W                                                       *)
19 (* HzV3N  050.002E+0                                                      *)
20 (* ---------------------------------------------------------------------- *)
21
22     (* ---------------------------------------------------------------- *)
23     (* INIZIALIZZAZIONI                                                 *)
24     (* ---------------------------------------------------------------- *)
25     (* Eseguo Spy per averlo attivo da Telnet. *)
26
27     IF (SpyOn) THEN IF NOT(SysSpyData(0, 0, NULL, NULL))THEN RETURN; END_IF; END_IF;
28
29     (* Eseguo gestione variabili one shot. *)
30
31     Done:=FALSE; (* Execution done *)
32     IF (Fault) THEN Fault:=FALSE; CaseNr:=0; END_IF;
33
34     (* ---------------------------------------------------------------- *)
```

```
35      (* GESTIONE PROTOCOLLO                                                  *)
36      (* ---------------------------------------------------------------------- *)
37      (* Eseguo gestione chiamate subroutines di case. *)
38
39      CASE (CaseNr) OF
40
41          (* ---------------------------------------------------------------------- *)
42          (* Attesa nessun carattere in ricezione per sincronismo su pausa. *)
43
44          0:
45          IF (TO_BOOL(SysGetIChars(File))) THEN
46              i:=SysFIBfClear(File); (* Clear input buffer *)
47              TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
48              RETURN;
49          END_IF;
50
51          (* Temporizzazione pausa. *)
52
53          IF ((SysGetSysTime(TRUE)-TimeBf) < 100000) THEN RETURN; END_IF;
54          i:=TO_INT(Sysmemset(ADR(RxData), 0, SIZEOF(RxData)));
55          Ptr:=ADR(RxData); (* Auxiliary pointer *)
56          CaseNr:=10; (* Case programma *)
57
58          (* ---------------------------------------------------------------------- *)
59          (* Gestione ricezione caratteri. *)
60
61          10:
62          WHILE (TO_BOOL(SysGetIChars(File))) DO
63              TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
64
65              @Ptr:=TO_USINT(Sysfgetc(File));
66              Ptr:=Ptr+1; (* Auxiliary pointer *)
67
68              (* Controllo lunghezza stringa. *)
69
70              IF (Ptr > (ADR(RxData)+SIZEOF(RxData))) THEN
71                  IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Stringa troppo
    lunga')); END_IF;
72                  Fault:=TRUE; (* Execution fault *)
73                  RETURN;
74              END_IF;
75          END_WHILE;
76
77          (* Se non ricevuto caratteri resto in questo case. *)
78
79          IF (Ptr = ADR(RxData)) THEN RETURN; END_IF;
80
81          (* Gestione e temporizzazione pausa. *)
82
83          IF ((SysGetSysTime(TRUE)-TimeBf) < 100000) THEN RETURN; END_IF;
84          IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#00000001, ADR('Rx'), ADR(RxData)); END_IF;
85          CaseNr:=100; (* Case programma *)
86
87          (* ---------------------------------------------------------------------- *)
88          (* Controllo dati ricevuti. *)
89
90          100:
91          IF (SysStrFind(ADR(RxData), ADR('END$r'), FIND_DEFAULT) = NULL) THEN
92              IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore ricezione s
    tringa')); END_IF;
```

```
 93            Fault:=TRUE; (* Execution fault *)
 94            RETURN;
 95        END_IF;
 96
 97        Ptr:=ADR(RxData); (* Auxiliary pointer *)
 98        CaseNr:=CaseNr+1; (* Case programma *)
 99
100        (* ---------------------------------------------------------------- *)
101        (* Controllo ricezione tag misura "3N  ". *)
102
103        101, 103, 105, 107:
104        Ptr:=SysStrFind(Ptr, ADR('3N  '), FIND_GET_END); (* Auxiliary pointer *)
105        IF (Ptr = NULL) THEN
106            IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore Tag misura'
    )); END_IF;
107            Fault:=TRUE; (* Execution fault *)
108            RETURN;
109        END_IF;
110        CaseNr:=CaseNr+1; (* Case programma *)
111
112        (* ---------------------------------------------------------------- *)
113        (* Acquisisco il valore della tensione, dopo primo "3N  ". *)
114
115        102:
116        IF NOT(SysVarsscanf(Ptr, '%f', REAL_TYPE, ADR(Voltage))) THEN
117            IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore acquisizion
    e tensione')); END_IF;
118            Fault:=TRUE; (* Execution fault *)
119            RETURN;
120        END_IF;
121        CaseNr:=CaseNr+1; (* Case programma *)
122
123        (* ---------------------------------------------------------------- *)
124        (* Acquisisco il valore della corrente, dopo secondo "3N  ". *)
125
126        104:
127        IF NOT(SysVarsscanf(Ptr, '%f', REAL_TYPE, ADR(Current))) THEN
128            IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore acquisizion
    e corrente')); END_IF;
129            Fault:=TRUE; (* Execution fault *)
130            RETURN;
131        END_IF;
132        CaseNr:=CaseNr+1; (* Case programma *)
133
134        (* ---------------------------------------------------------------- *)
135        (* Acquisisco il valore della potenza, dopo terzo "3N  ". *)
136
137        106:
138        IF NOT(SysVarsscanf(Ptr, '%f', REAL_TYPE, ADR(Power))) THEN
139            IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore acquisizion
    e potenza')); END_IF;
140            Fault:=TRUE; (* Execution fault *)
141            RETURN;
142        END_IF;
143        CaseNr:=CaseNr+1; (* Case programma *)
144
145        (* ---------------------------------------------------------------- *)
146        (* Acquisisco il valore della potenza, dopo terzo "3N  ". *)
147
148        108:
```

```
149         IF NOT(SysVarsscanf(Ptr, '%f', REAL_TYPE, ADR(Frequency))) THEN
150             IF (SpyOn) THEN i:=SysSpyData(SPY_ASCII, 16#40000000, ADR('Er'), ADR('Errore acquisizion
    e frequenza')); END_IF;
151             Fault:=TRUE; (* Execution fault *)
152             RETURN;
153         END_IF;
154         CaseNr:=CaseNr+1; (* Case programma *)
155
156         (* ---------------------------------------------------------------- *)
157         (* Eseguo report dati acquisisti. *)
158
159         109:
160         IF (SpyOn) THEN
161             i:=SysVarsnprintf(ADR(RxData), SIZEOF(RxData), 'Voltage=%.3f', REAL_TYPE, ADR(Voltage));
162             i:=SysLWVarsnprintf(ADR(RxData), SIZEOF(RxData), ', Current=%.3f', REAL_TYPE, ADR(Current)
    ;
163             i:=SysLWVarsnprintf(ADR(RxData), SIZEOF(RxData), ', Power=%.3f', REAL_TYPE, ADR(Power));

164             i:=SysLWVarsnprintf(ADR(RxData), SIZEOF(RxData), ', Frequency=%.3f', REAL_TYPE, ADR(Frequen
    cy));
165             i:=SysSpyData(SPY_ASCII, 16#00000002, ADR('Lg'), ADR(RxData));
166         END_IF;
167         CaseNr:=10; (* Case programma *)
168     END_CASE;
169
170 (* [End of file] *)
171
```