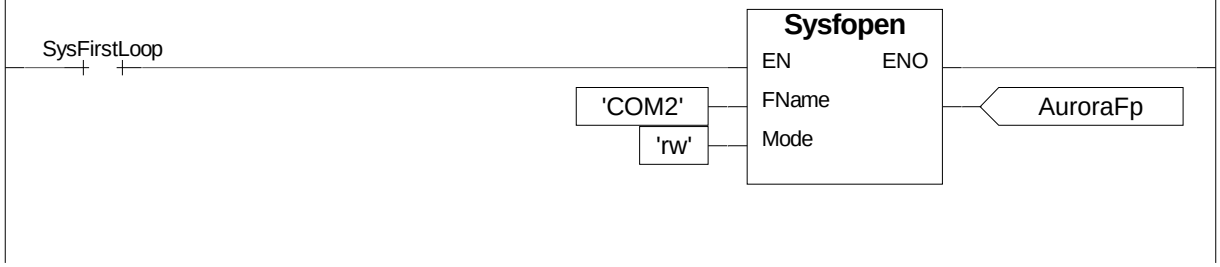
```
VAR_GLOBAL
TDEnergy AT %MD100.96 : REAL; (* Total daily energy (Kw) *)
TYEnergy AT %MD100.100 : REAL; (* Total year energy (Kw) *)
TTEnergy AT %MD100.104 : REAL; (* Total energy (Kw) *)
TGridPower AT %MD100.108 : REAL; (* Total grid power (W) *)
ICmErrors AT %MD100.112 : UDINT; (* Inverter communication errors *)
DEnergy AT %MD100.128 : ARRAY[ 0..15 ] OF REAL; (* Daily energy (Kw) *)
YEnergy AT %MD100.192 : ARRAY[ 0..15 ] OF REAL; (* Year energy (Kw) *)
TEnergy AT %MD100.256 : ARRAY[ 0..15 ] OF REAL; (* Total energy (Kw) *)
GridVoltage AT %MD100.320 : ARRAY[ 0..15 ] OF REAL; (* Grid voltage (V) *)
GridCurrent AT %MD100.384 : ARRAY[ 0..15 ] OF REAL; (* Grid current (A) *)
GridPower AT %MD100.448 : ARRAY[ 0..15 ] OF REAL; (* Grid power (W) *)
Frequency AT %MD100.512 : ARRAY[ 0..15 ] OF REAL; (* Frequency (Hz) *)
ITemperature AT %MD100.576 : ARRAY[ 0..15 ] OF REAL; (* Inverter temperature (°C) *)
I1Voltage AT %MD100.640 : ARRAY[ 0..15 ] OF REAL; (* Input 1 voltage (V) *)
I2Voltage AT %MD100.704 : ARRAY[ 0..15 ] OF REAL; (* Input 2 voltage (V) *)
I1Current AT %MD100.768 : ARRAY[ 0..15 ] OF REAL; (* Input 1 current (A) *)
I2Current AT %MD100.832 : ARRAY[ 0..15 ] OF REAL; (* Input 2 current (A) *)
AuroraFp : FILEP; (* Aurora inverter stream *)
END_VAR
```

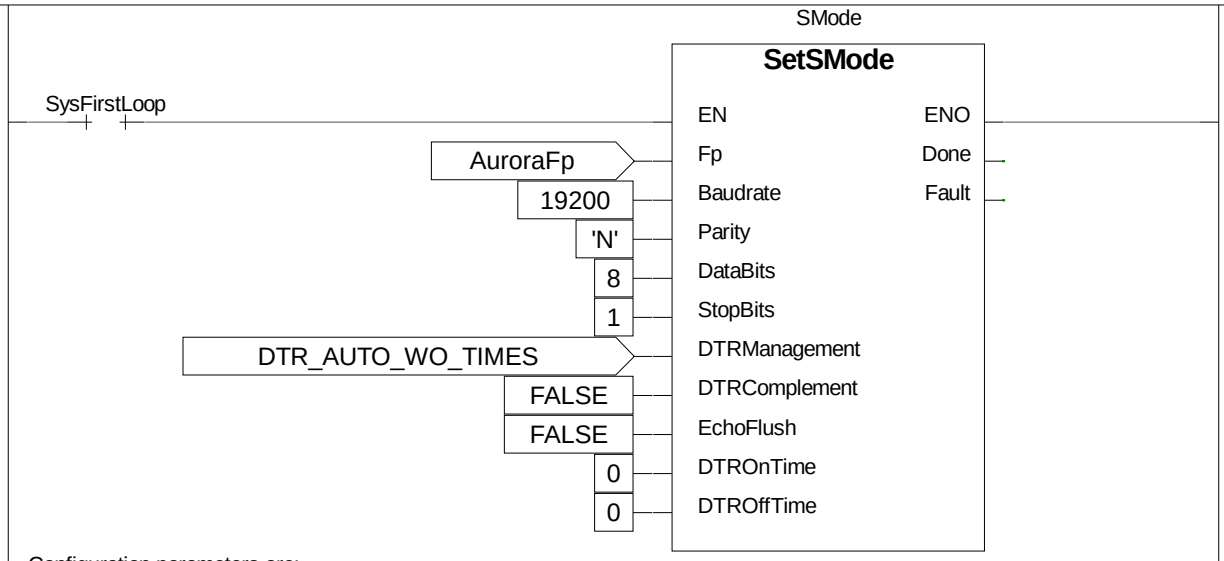
PROGRAM PortOpen

```
VAR
SMode : SetSMode;
END_VAR
```

1



2



Configuration parameters are:

19200 baud (default value)

1 stop bit

no parity

	Project : PowerOneCm	
	PROGRAM : PortOpen	
	Release :	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1

```
VAR
ACe : AuroraCEnergy; (* FB Aurora cumulated energy *)
AcqOn : BOOL; (* Acquisizione attiva *)
RegIDx : USINT; (* Register index *)
DPtr : @REAL; (* DataPointer *)
ADm : AuroraDSPMeasure; (* FB Aurora DSP measure *)
TimBuff : UDINT; (* Time buffer *)
IIDx : USINT; (* Inverter index *)
IAOfs : USINT; (* Inverter address offset *)
INumber : USINT; (* Number of inverters *)
i : USINT; (* Auxiliary variable *)
AuxErrors : UDINT; (* Auxiliary error counter *)
END_VAR
```

```
(* ********************************************************************** *)
(* LETTURA REGISTRI ENERGIA DA INVERTER AURORA                            *)
(* ********************************************************************** *)
(* Questo programma esegue lettura registri energia da inverter Aurora.   *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------- *)
    (* DEFINIZIONE LIMITI RETE DI INVERTERS                             *)
    (* ---------------------------------------------------------------- *)
    (* Occorre definire primo indirizo dell'inverter della rete ed il numero *)
    (* di inverters presenti nella rete. *)

    IAOfs:=2; (* Inverter address offset *)
    INumber:=3; (* Number of inverters *)

    (* ---------------------------------------------------------------- *)
    (* INIZIALIZZAZIONE                                                 *)
    (* ---------------------------------------------------------------- *)
    (* Eseguo inizializzazione gestione. *)

    IF (SysFirstLoop) THEN
        IIDx:=0; (* Inverter index *)
        RegIDx:=10-1; (* Register index *)
        TimBuff:=SysGetSysTime(TRUE); (* Time buffer *)
    END_IF;

    (* ---------------------------------------------------------------- *)
    (* GESTIONE BLOCCHI FUNZIONE COMUNICAZIONE CON AURORA               *)
    (* ---------------------------------------------------------------- *)
    (* Eseguo gestione lettura registri energia. *)

    ACe.Address:=IIDx+IAOfs; (* Inverter address *)
    ACe(Start:=AcqOn, File:=AuroraFp); (* FB Aurora cumulated energy *)
    IF (ACe.Done) THEN @DPtr:=TO_REAL(ACe.Value)/1000.0; END_IF;

    (* Eseguo gestione lettura registri DSP. *)

    ADm.Address:=IIDx+IAOfs; (* Inverter address *)
    ADm(Start:=AcqOn, File:=AuroraFp); (* FB Aurora DSP measure *)
    IF (ADm.Done) THEN @DPtr:=ADm.Value; END_IF;

    (* Gestione conteggio errori di comunicazione. *)

```

```
44      IF ((ACe.Errors+ADm.Errors) <> AuxErrors) THEN
45          AuxErrors:=ACe.Errors+ADm.Errors; (* Auxiliary error counter *)
46          IF (AuxErrors > 0) THEN ICmErrors:=ICmErrors+1; END_IF;
47      END_IF;
48
49      (* ---------------------------------------------------------------------- *)
50      (* GESTIONE COMANDO ACQUISIZIONE                                          *)
51      (* ---------------------------------------------------------------------- *)
52      (* Comando acquisizione, si attiva su fine lettura o su errore. *)
53      (* Il timeout delle FB Aurora è 2 Sec metto un tempo maggiore. *)
54
55      AcqOn:=FALSE; (* Acquisizione attiva *)
56      IF NOT(((SysGetSysTime(TRUE)-TimBuff) >= 3000000)
57             OR (ACe.Done) OR (ACe.Fault)
58             OR (ADm.Done) OR (ADm.Fault)) THEN RETURN;
59      END_IF;
60
61      TimBuff:=SysGetSysTime(TRUE); (* Time buffer *)
62      AcqOn:=TRUE; (* Acquisizione attiva *)
63      RegIDx:=RegIDx+1; (* Register index *)
64      ACe.Enable:=FALSE; (* FB enable *)
65      ADm.Enable:=FALSE; (* FB enable *)
66
67      (* ---------------------------------------------------------------------- *)
68      (* GESTIONE REGISTRI INVERTER                                             *)
69      (* ---------------------------------------------------------------------- *)
70
71      CASE (RegIDx) OF
72
73          (* ---------------------------------------------------------------- *)
74          (* GESTIONE LETTURA ENERGIA                                         *)
75          (* ---------------------------------------------------------------- *)
76          (* Lettura energia giornaliera prodotta. *)
77
78          10: ACe.Enable:=TRUE; ACe.Parameter:=0; DPtr:=ADR(DEnergy[IIDx]); (* Daily energy (Kw) *)
79              TDEnergy:=0.0; (* Total daily energy (Kw) *)
80              FOR i:=0 TO 15 DO TDEnergy:=TDEnergy+DEnergy[i]; END_FOR;
81
82          (* ---------------------------------------------------------------- *)
83          (* Lettura energia annuale prodotta. *)
84
85          11: ACe.Enable:=TRUE; ACe.Parameter:=4; DPtr:=ADR(YEnergy[IIDx]); (* Year energy (Kw) *)
86              TYEnergy:=0.0; (* Total year energy (Kw) *)
87              FOR i:=0 TO 15 DO TYEnergy:=TYEnergy+YEnergy[i]; END_FOR;
88
89          (* ---------------------------------------------------------------- *)
90          (* Lettura energia totale prodotta. *)
91
92          12: ACe.Enable:=TRUE; ACe.Parameter:=5; DPtr:=ADR(TEnergy[IIDx]); (* Total energy (Kw) *)
93              TTEnergy:=0.0; (* Total energy (Kw) *)
94              FOR i:=0 TO 15 DO TTEnergy:=TTEnergy+TEnergy[i]; END_FOR;
95              RegIDx:=50-1; (* Register index *)
96
97          (* ---------------------------------------------------------------- *)
98          (* GESTIONE LETTURA REGISTRI DSP                                    *)
99          (* ---------------------------------------------------------------- *)
100         (* Lettura grid voltage (For three-phases systems is the mean). *)
101
102         50: ADm.Enable:=TRUE; ADm.Measure:=1; DPtr:=ADR(GridVoltage[IIDx]); (* Grid voltage (V) *)
103
```

```
104         (* ---------------------------------------------------------------- *)
105         (* Lettura grid current (For three-phases systems is the mean). *)
106
107         51: ADm.Enable:=TRUE; ADm.Measure:=2; DPtr:=ADR(GridCurrent[IIDx]); (* Grid current (A) *)
108
109         (* ---------------------------------------------------------------- *)
110         (* Lettura grid power (For three-phases systems is the mean). *)
111
112         52: ADm.Enable:=TRUE; ADm.Measure:=3; DPtr:=ADR(GridPower[IIDx]); (* Grid power (W) *)
113             TGridPower:=0.0; (* Total grid power (W) *)
114             FOR i:=0 TO 15 DO TGridPower:=TGridPower+GridPower[i]; END_FOR;
115
116         (* ---------------------------------------------------------------- *)
117         (* Lettura frequency (For three-phases systems is the mean). *)
118
119         53: ADm.Enable:=TRUE; ADm.Measure:=4; DPtr:=ADR(Frequency[IIDx]); (* Frequency (Hz) *)
120
121         (* ---------------------------------------------------------------- *)
122         (* Lettura inverter temperature. *)
123
124         54: ADm.Enable:=TRUE; ADm.Measure:=21; DPtr:=ADR(ITemperature[IIDx]); (* Inverter temperature (
    °C) *)
125
126         (* ---------------------------------------------------------------- *)
127         (* Lettura input 1 voltage. *)
128
129         55: ADm.Enable:=TRUE; ADm.Measure:=23; DPtr:=ADR(I1Voltage[IIDx]); (* Input 1 voltage (V) *)
130
131         (* ---------------------------------------------------------------- *)
132         (* Lettura input 2 voltage. *)
133
134         56: ADm.Enable:=TRUE; ADm.Measure:=26; DPtr:=ADR(I2Voltage[IIDx]); (* Input 2 voltage (V) *)
135
136         (* ---------------------------------------------------------------- *)
137         (* Lettura input 1 current. *)
138
139         57: ADm.Enable:=TRUE; ADm.Measure:=25; DPtr:=ADR(I1Current[IIDx]); (* Input 1 current (A) *)
140
141         (* ---------------------------------------------------------------- *)
142         (* Lettura input 2 current. *)
143
144         58: ADm.Enable:=TRUE; ADm.Measure:=27; DPtr:=ADR(I2Current[IIDx]); (* Input 2 current (A) *)
145
146             (* Controllo se terminato loop lettura di tutti gli inverters. *)
147
148             IIDx:=IIDx+1; (* Inverter index *)
149             IF (IIDx >= INumber) THEN IIDx:=0; END_IF;
150             RegIDx:=10-1; (* Register index *)
151
152     END_CASE;
153
154 (* [End of file] *)
155
156
```

FUNCTION_BLOCK DB100AddOffset

Returns the DB100 address offset
ENCRYPTED CODE

```
VAR_INPUT
Offset : UINT; (* Address offset *)
END_VAR

VAR_OUTPUT
Address : UDINT; (* Address value *)
END_VAR
```

1

	Project : PowerOneCm	
	FUNCTION BLOCK : DB100AddOffset	
	Release : PowerOneCm	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK SetSMode

Set the serial communication mode
ENCRYPTED CODE

```
VAR_INPUT
Fp : FILEP;
Baudrate : UDINT; (* Baudrate *)
Parity : STRING[ 1 ]; (* Parity type *)
DataBits : USINT; (* Nr of data bits *)
StopBits : USINT; (* Nr of stop bits *)
DTRManagement : USINT; (* DTR management type *)
DTRComplement : BOOL; (* Complement the DTR signal *)
EchoFlush : BOOL; (* Flush the echo *)
DTROnTime : UINT; (* DTR On wait time *)
DTROffTime : UINT; (* DTR Off wait time *)
END_VAR

VAR_OUTPUT
Done : BOOL; (* Execution done *)
Fault : BOOL; (* Execution fault *)
END_VAR
```

1

	Project : PowerOneCm	
	FUNCTION BLOCK : SetSMode	
	Release : PowerOneCm	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK CRCPolinomial

ENCRYPTED CODE

```
VAR_INPUT
Data : @USINT; (* Stringa record dati *)
Length : USINT; (* Lunghezza record dati *)
END_VAR

VAR_OUTPUT
CRC : UINT; (* CRC *)
END_VAR
```

1

	Project : PowerOneCm	
	FUNCTION BLOCK : CRCPolinomial	
	Release : PowerOneCm	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK AuroraCEnergy

Manages the modbus RTU master communication
ENCRYPTED CODE

```
VAR_INPUT
Enable : BOOL; (* FB enable *)
Start : BOOL; (* Start command *)
File : FILEP; (* Terminal I/O pointer *)
Address : USINT; (* Inverter address *)
Parameter : USINT; (* Parameter code *)
END_VAR

VAR_OUTPUT
Done : BOOL; (* Command done *)
Fault : BOOL; (* Command fault *)
Value : UDINT; (* Parameter value *)
Errors : UDINT; (* Error counter *)
END_VAR
```

1

	Project : PowerOneCm	
	FUNCTION BLOCK : AuroraCEnergy	
	Release : PowerOneCm	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1

FUNCTION_BLOCK AuroraDSPMeasure

Manages the modbus RTU master communication
ENCRYPTED CODE

```
VAR_INPUT
Enable : BOOL; (* FB enable *)
Start : BOOL; (* Start command *)
File : FILEP; (* Terminal I/O pointer *)
Address : USINT; (* Inverter address *)
Measure : USINT; (* Measure code *)
END_VAR

VAR_OUTPUT
Done : BOOL; (* Command done *)
Fault : BOOL; (* Command fault *)
Value : REAL; (* Measure value *)
Errors : UDINT; (* Error counter *)
END_VAR
```

1

	Project : PowerOneCm	
	FUNCTION BLOCK : AuroraDSPMeasure	
	Release : PowerOneCm	Ver :1.00
	Author :	Date:27/06/2012
	Note :	Page:1 of 1