```
VAR_GLOBAL
Day : USINT; (* Valore giorno *)
Month : USINT; (* Valore mese *)
Year : USINT; (* Valore anno *)
Hour : USINT; (* Valore ora *)
Minute : USINT; (* Valore minuti *)
Second : USINT; (* Valore secondi *)
END_VAR
```

PROGRAM LCDMessages

```
VAR
LCD : ArduinoLCD; (* LCD management FB *)
DTime : SysETimeToDate; (* Conversione in Data/Ora *)
TimeBf : UDINT; (* Time buffer (uS) *)
END_VAR
```

```
(* ********************************************************************** *)
(* PROGRAM "LCDMessages"                                                  *)
(* ********************************************************************** *)
(* Questo programma gestisce i messaggi del display LCD.                  *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------------- *)
    (* GESTIONE DISPLAY LCD                                                   *)
    (* ---------------------------------------------------------------------- *)
    (* Gestione blocco funzione display. *)

    LCD.I2CAddress:=16#3F; (* LCD I2C address *)
    LCD.HMIBuiltInID:=NewHMI.HMIBuiltInID; (* ID gestione messaggi *)
    LCD(Enable:=TRUE); (* Gestione FB *)

    (* ---------------------------------------------------------------------- *)
    (* GESTIONE ROTAZIONE MESSAGGI DISPLAY                                    *)
    (* ---------------------------------------------------------------------- *)
    (* Ogni 3 secondi vario il messaggio visualizzato sul display. *)

    IF ((SysGetSysTime(TRUE)-TimeBf) > 3000000) THEN
        TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
        NewHMI.ViewMID:=NewHMI.ViewMID+1; (* Displays the message with the defined ID *)

        IF (NewHMI.ViewMID > 2) THEN NewHMI.ViewMID:=1; END_IF;
    END_IF;

    (* ---------------------------------------------------------------------- *)
    (* GESTIONE DATA/ORA                                                      *)
    (* ---------------------------------------------------------------------- *)
    (* Gestione Data/Ora per visualizzazione. *)

    DTime(EpochTime:=SysDateTime); (* Conversione in Data/Ora *)
    Day:=DTime.Day; (* Valore giorno *)
    Month:=DTime.Month; (* Valore mese *)
    Year:=TO_USINT(DTime.Year-2000); (* Valore anno *)
    Hour:=DTime.Hour; (* Valore ora *)
    Minute:=DTime.Minute; (* Valore minuti *)
    Second:=DTime.Second; (* Valore secondi *)

(* [End of file] *)
```

	Project : ArduinoLCD	
	PROGRAM : LCDMessages	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:18/02/2014
	Note :	Page:1 of 1

FUNCTION_BLOCK ArduinoLCD

Arduino LCD with I2C interface

```
VAR_INPUT
Enable : BOOL; (* FB enable *)
HMIBuiltInID : @_HMIBUILTINDATA; (* HMI built in ID *)
I2CAddress : USINT; (* LCD I2C address *)
END_VAR

VAR_OUTPUT
Enabled : BOOL; (* FB enabled *)
Fault : BOOL; (* FB fault *)
END_VAR

VAR_EXTERNAL
SysActTaskID : USINT; (* Tasks in progress identification number *)
END_VAR

VAR
CaseNr : ARRAY[ 0..1 ] OF USINT; (* Case gestione *)
DWrBuffer : ARRAY[ 0..3 ] OF BYTE; (* Display write buffer *)
TimeBf : UDINT; (* Time buffer (uS) *)
DDIDx : USINT; (* Display data index *)
Ptr : @USINT; (* Auxiliary pointer *)
END_VAR
```

```
 1 (* ********************************************************************** *)
 2 (* FUNCTION BLOCK "ArduinoLCD"                                            *)
 3 (* ********************************************************************** *)
 4 (* Questo blocco funzione esegue la gestione del controller display SainSmart *)
 5 (* IIC/I2C/TWI 1602 Serial LCD Module Display For Arduino UNO MEGA R3.    *)
 6 (* Il controller utilizza un PCF8574 "Remote 8-bit I/O expander for I2C-bus"  *)
 7 (* per gestire i segnali di controllo di un display con interfaccia parallela *)
 8 (* QC2004A Systronix basato sul controller HD44780U della Hitachi.       *)
 9 (* ---------------------------------------------------------------------- *)
10 (* Gli 8 ports di uscita del I/O expander sono connessi al display nel modo.  *)
11 (*                                                                        *)
12 (* Pin  4 P0 -> Pin 4 Rs Display                                          *)
13 (* Pin  5 P1 -> Pin 5 R/W Display                                         *)
14 (* Pin  6 P2 -> Pin 6 En Display                                          *)
15 (* Pin  7 P3 -> Comando backlight                                         *)
16 (* Pin  9 P4 -> Pin 11 D4 Display                                         *)
17 (* Pin 10 P5 -> Pin 12 D5 Display                                         *)
18 (* Pin 11 P6 -> Pin 13 D6 Display                                         *)
19 (* Pin 12 P7 -> Pin 14 D7 Display                                         *)
20 (* ---------------------------------------------------------------------- *)
21 (* connesso su bus I2C. Viene gestita la connessione con la FB di gestione    *)
22 (* messaggi di cui è fornito indirizzo struttura "_HMIBUILTINDATA".       *)
23 (* ---------------------------------------------------------------------- *)
24
25     (* ---------------------------------------------------------------- *)
26     (* INIZIALIZZAZIONI                                                 *)
27     (* ---------------------------------------------------------------- *)
28     (* Eseguo reset bit di one shot. *)
29
30     IF (Fault) THEN Fault:=FALSE; CaseNr[0]:=0; CaseNr[1]:=0; END_IF;
31
32     (* Controllo se "HMIBuiltInID" definito, deve essere passato l'indirizzo *)
33     (* della FB creata da LogicLab. *)
```

	Project : ArduinoLCD	
	FUNCTION BLOCK : ArduinoLCD	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:18/02/2014
	Note :	

```
34
35      IF (TO_UDINT(HMIBuiltInID) = 0) THEN Fault:=TRUE; RETURN; END_IF;
36      IF (@HMIBuiltInID.UniqueID <> 16#15052013) THEN Fault:=TRUE; RETURN; END_IF;
37
38      (* Eseguo controllo se FB eseguita in task di background. *)
39
40      IF (SysActTaskID <> ID_TASK_BACK) THEN Fault:=TRUE; RETURN; END_IF;
41
42      (* ---------------------------------------------------------------------- *)
43      (* ESEGUO GESTIONE ABILITAZIONE                                           *)
44      (* ---------------------------------------------------------------------- *)
45      (* Gestione abilitazione blocco funzione. *)
46
47      IF NOT(Enable) THEN Enabled:=FALSE; RETURN; END_IF;
48
49      (* Gestione primo loop abilitazione. *)
50
51      IF NOT(Enabled) THEN
52          Enabled:=TRUE; (* FB enabled *)
53          CaseNr[0]:=16#00; (* Case gestione *)
54          CaseNr[1]:=16#FF; (* Case gestione *)
55      END_IF;
56
57      (* Per informare la FB "HMIBuiltInMessages" che la figlia è connessa si *)
58      (* deve azzerare il bit di conrollo. *)
59
60      @HMIBuiltInID.Heartbeat:=FALSE; (* Heartbeat (To/From child) *)
61
62      (* ---------------------------------------------------------------------- *)
63      (* GESTIONE ABILITAZIONE                                                  *)
64      (* ---------------------------------------------------------------------- *)
65      (* Gestione abilitazione blocco funzione. *)
66
67      IF NOT(Enable) THEN Enabled:=FALSE; RETURN; END_IF;
68
69      (* Gestione primo loop abilitazione. *)
70
71      IF NOT(Enabled) THEN
72          Enabled:=TRUE; (* FB enabled *)
73          CaseNr[0]:=16#00; (* Case gestione *)
74          CaseNr[1]:=16#FF; (* Case gestione *)
75      END_IF;
76
77      (* ---------------------------------------------------------------------- *)
78      (* TEMPORIZZAZIONE TRA CASES PROGRAMMA                                    *)
79      (* ---------------------------------------------------------------------- *)
80      (* Siccome non è possibile leggere dal display il segnale di busy, viene *)
81      (* eseguita una temporizzazione tra i vari cases pari al tempo necessario *)
82      (* al display per eseguire il comando più lento (Clear display 760 uS). *)
83
84      IF (CaseNr[0] <> CaseNr[1]) THEN
85          IF ((SysGetSysTime(TRUE)-TimeBf) < 1000) THEN RETURN; END_IF;
86          TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
87          CaseNr[1]:=CaseNr[0]; (* Case gestione *)
88      END_IF;
89
90      (* ---------------------------------------------------------------------- *)
91      (* GESTIONE CASE PROGRAMMA                                                *)
92      (* ---------------------------------------------------------------------- *)
93      (* Eseguo gestione case LCD. *)
```

```
 94
 95     CASE (CaseNr[0]) OF
 96
 97         (* ---------------------------------------------------------------- *)
 98         (* INIZIALIZZAZIONE DISPLAY                                         *)
 99         (* ---------------------------------------------------------------- *)
100         (* In questi cases, viene eseguita la sequenza di reset del display.  *)
101         (* In tutti questa cases occorre eseguire l'interfacciamento a 4 bit, *)
102         (* al termine della fase sarà possibile accedere al display ad 8 bit  *)
103         (* eseguendo due accessi consecutivi a 4 bit.                         *)
104         (* ---------------------------------------------------------------- *)
105         (* Eseguo comando "Default function set". *)
106
107         0, 1, 2:
108         DWrBuffer[0]:=16#30; (* *)
109         DWrBuffer[1]:=16#34; (* En:=TRUE *)
110         DWrBuffer[2]:=16#30; (* *)
111         IF NOT(SysI2CWrRd(I2CAddress, 3, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
112         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
113
114         (* ---------------------------------------------------------------- *)
115         (* Eseguo set interfaccia a 4 bit. Dopo questo comando è possibile *)
116         (* inviare comandi ad 8 bit eseguendo due accesi consecutivi a 4 bit. *)
117
118         3:
119         DWrBuffer[0]:=16#20; (* *)
120         DWrBuffer[1]:=16#24; (* En:=TRUE *)
121         DWrBuffer[2]:=16#20; (* *)
122         IF NOT(SysI2CWrRd(I2CAddress, 3, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
123         CaseNr[0]:=10; (* Case gestione *)
124
125         (* ---------------------------------------------------------------- *)
126         (* INIZIALIZZAZIONE DISPLAY                                         *)
127         (* ---------------------------------------------------------------- *)
128         (* Eseguo comando "Function set DL, N, F", 8 bits, 2 lines, 5*7 dots *)
129
130         10:
131         IF NOT(LCDOut8Bit(I2CAddress, 16#28, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
132         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
133
134         (* ---------------------------------------------------------------- *)
135         (* Eseguo comando "Display on/off control". *)
136
137         11:
138         IF NOT(LCDOut8Bit(I2CAddress, 16#04, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
139         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
140
141         (* ---------------------------------------------------------------- *)
142         (* Eseguo comando "Clear display". *)
143
144         12:
145         IF NOT(LCDOut8Bit(I2CAddress, 16#01, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
146         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
147
148         (* ---------------------------------------------------------------- *)
149         (* Eseguo comando "Cursor and display shift". *)
150
151         13:
152         IF NOT(LCDOut8Bit(I2CAddress, 16#06, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
153         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
```

```
154
155         (* ---------------------------------------------------------------- *)
156         (* Eseguo comando "Display on/off control". *)
157
158         14:
159         IF NOT(LCDOut8Bit(I2CAddress, 16#0C, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
160         CaseNr[0]:=100; (* Case gestione *)
161
162         (* ---------------------------------------------------------------- *)
163         (* SCRITTURA DATI SU DISPLAY                                        *)
164         (* ---------------------------------------------------------------- *)
165         (* Eseguo scrittura dato sul display. Il tempo necessario è 37 uS   *)
166         (* quindi non eseguo temporizzazione, il tempo di loop del programma  *)
167         (* PLC sarà sicuramente maggiore.                                   *)
168         (* ---------------------------------------------------------------- *)
169         (* Eseguo comando "Set DDRAM address". *)
170
171         100:
172         IF NOT(@HMIBuiltInID.DUpdated) THEN RETURN; END_IF;
173         IF NOT(LCDOut8Bit(I2CAddress, 16#80, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
174         DDIDx:=0; (* Display data index *)
175         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
176
177         (* ---------------------------------------------------[1a Riga]-- *)
178
179         101:
180         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
181         IF NOT(LCDOut8Bit(I2CAddress, @Ptr, TRUE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
182         DDIDx:=DDIDx+1; (* Display data index *)
183         IF (DDIDx < 20) THEN RETURN; END_IF;
184
185         IF NOT(LCDOut8Bit(I2CAddress, 16#C0, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
186         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
187
188         (* ---------------------------------------------------[2a Riga]-- *)
189
190         102:
191         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
192         IF NOT(LCDOut8Bit(I2CAddress, @Ptr, TRUE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
193         DDIDx:=DDIDx+1; (* Display data index *)
194         IF (DDIDx < 40) THEN RETURN; END_IF;
195
196         IF NOT(LCDOut8Bit(I2CAddress, 16#94, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
197         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
198
199         (* ---------------------------------------------------[3a Riga]-- *)
200
201         103:
202         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
203         IF NOT(LCDOut8Bit(I2CAddress, @Ptr, TRUE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
204         DDIDx:=DDIDx+1; (* Display data index *)
205         IF (DDIDx < 60) THEN RETURN; END_IF;
206
207         IF NOT(LCDOut8Bit(I2CAddress, 16#D4, FALSE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
208         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
209
210         (* ---------------------------------------------------[4a Riga]-- *)
211
212         104:
213         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
```

```
214         IF NOT(LCDOut8Bit(I2CAddress, @Ptr, TRUE, TRUE)) THEN CaseNr[0]:=0; RETURN; END_IF;
215         DDIDx:=DDIDx+1; (* Display data index *)
216         IF (DDIDx < 80) THEN RETURN; END_IF;
217
218         @HMIBuiltInID.DUpdated:=FALSE; (* Display updated *)
219         CaseNr[0]:=100; (* Case gestione *)
220     ELSE
221         CaseNr[0]:=0; (* Case gestione *)
222     END_CASE;
223
224 (* [End of file] *)
225
226
```

FUNCTION MemSet

(SFR058A200) Fills memory with value
ENCRYPTED CODE

```
VAR_INPUT
Buffer : @USINT; (* Buffer address *)
Value : USINT; (* Value to set *)
Size : UDINT; (* Buffer size *)
END_VAR
```

1

	Project : ArduinoLCD	
	FUNCTION : MemSet	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:18/02/2014
	Note :	Page:1 of 1

FUNCTION LCDOut8Bit

```
VAR_INPUT
I2CAddress : USINT; (* I2C Address *)
OData : BYTE; (* Output data to display *)
Rs : BOOL; (* RS command *)
Backlight : BOOL; (* Backlight command *)
END_VAR

VAR
OBuffer : ARRAY[ 0..2 ] OF BYTE; (* Output buffer *)
END_VAR
```

```
(* ********************************************************************** *)
(* FUNCTION "BOOL LCDOut8Bit(BYTE OData, BOOL Rs, BOOL Backlight)"        *)
(* ********************************************************************** *)
(* Questa funzione esegue l'uscita del dato sul display.                  *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------- *)
    (* GESTIONE USCITA MS NIBBLE                                        *)
    (* ---------------------------------------------------------------- *)
    (* Gestisco segnali di comando verso display. *)

    OBuffer[0]:=OData AND 16#F0; (* Output buffer *)
    IF (Rs) THEN OBuffer[0]:=OBuffer[0] OR 16#01; END_IF; (* Rs:=TRUE *)
    IF (Backlight) THEN OBuffer[0]:=OBuffer[0] OR 16#08; END_IF; (* Backlight:=TRUE *)

    (* Copio output buffer in tutti i bytes di uscita dato display. *)

    OBuffer[1]:=OBuffer[0] OR 16#04; (* En:=TRUE *)
    OBuffer[2]:=OBuffer[0]; (* Output buffer *)
    LCDOut8Bit:=SysI2CWrRd(I2CAddress, 3, ADR(OBuffer), 0, 0); (* Function result *)
    IF NOT(LCDOut8Bit) THEN RETURN; END_IF;

    (* ---------------------------------------------------------------- *)
    (* GESTIONE USCITA LS NIBBLE                                        *)
    (* ---------------------------------------------------------------- *)
    (* Gestisco segnali di comando verso display. *)

    OBuffer[0]:=(OData*16) AND 16#F0; (* Output buffer *)
    IF (Rs) THEN OBuffer[0]:=OBuffer[0] OR 16#01; END_IF; (* Rs:=TRUE *)
    IF (Backlight) THEN OBuffer[0]:=OBuffer[0] OR 16#08; END_IF; (* Backlight:=TRUE *)

    (* Copio output buffer in tutti i bytes di uscita dato display. *)

    OBuffer[1]:=OBuffer[0] OR 16#04; (* En:=TRUE *)
    OBuffer[2]:=OBuffer[0]; (* Output buffer *)
    LCDOut8Bit:=SysI2CWrRd(I2CAddress, 3, ADR(OBuffer), 0, 0); (* Function result *)

(* [End of file] *)
```

	Project : ArduinoLCD	
	FUNCTION : LCDOut8Bit	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:18/02/2014
	Note :	Page:1 of 1