```
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 : CustomLCDwI2C; (* 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#3C; (* 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 *)
    DTime.Year:=DTime.Year-2000; (* Year *)
    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 : LCDCustom	
	PROGRAM : LCDMessages	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:13/12/2013
	Note :	Page:1 of 1

FUNCTION_BLOCK CustomLCDwI2C

Custom 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..1 ] OF BYTE; (* Display write buffer *)
TimeBf : UDINT; (* Time buffer (uS) *)
DDIDx : USINT; (* Display data index *)
Ptr : @USINT; (* Auxiliary pointer *)
END_VAR
```

```
(* ********************************************************************** *)
(* FUNCTION BLOCK "CustomLCDwI2C"                                         *)
(* ********************************************************************** *)
(* Questo blocco funzione esegue la gestione del display LCD MC21605 Midas    *)
(* connesso su bus I2C. Viene gestita la connessione con la FB di gestione    *)
(* messaggi di cui è fornito indirizzo struttura "_HMIBUILTINDATA".           *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------------- *)
    (* INIZIALIZZAZIONI                                                       *)
    (* ---------------------------------------------------------------------- *)
    (* Eseguo reset bit di one shot. *)

    IF (Fault) THEN Fault:=FALSE; CaseNr[0]:=0; CaseNr[1]:=0; END_IF;

    (* Controllo se "HMIBuiltInID" definito, deve essere passato l'indirizzo *)
    (* della FB creata da LogicLab. *)

    IF (TO_UDINT(HMIBuiltInID) = 0) THEN Fault:=TRUE; RETURN; END_IF;
    IF (@HMIBuiltInID.UniqueID <> 16#15052013) THEN Fault:=TRUE; RETURN; END_IF;

    (* Eseguo controllo se FB eseguita in task di background. *)

    IF (SysActTaskID <> ID_TASK_BACK) THEN Fault:=TRUE; RETURN; END_IF;

    (* ---------------------------------------------------------------------- *)
    (* ESEGUO GESTIONE ABILITAZIONE                                           *)
    (* ---------------------------------------------------------------------- *)
    (* Gestione abilitazione blocco funzione. *)

    IF NOT(Enable) THEN Enabled:=FALSE; RETURN; END_IF;

    (* Gestione primo loop abilitazione. *)
```

```
34
35      IF NOT(Enabled) THEN
36          Enabled:=TRUE; (* FB enabled *)
37          CaseNr[0]:=16#00; (* Case gestione *)
38          CaseNr[1]:=16#FF; (* Case gestione *)
39      END_IF;
40
41      (* Per informare la FB "HMIBuiltInMessages" che la figlia è connessa si *)
42      (* deve azzerare il bit di conrollo. *)
43
44      @HMIBuiltInID.Heartbeat:=FALSE; (* Heartbeat (To/From child) *)
45
46      (* ------------------------------------------------------------------- *)
47      (* GESTIONE ABILITAZIONE                                               *)
48      (* ------------------------------------------------------------------- *)
49      (* Gestione abilitazione blocco funzione. *)
50
51      IF NOT(Enable) THEN Enabled:=FALSE; RETURN; END_IF;
52
53      (* Gestione primo loop abilitazione. *)
54
55      IF NOT(Enabled) THEN
56          Enabled:=TRUE; (* FB enabled *)
57          CaseNr[0]:=16#00; (* Case gestione *)
58          CaseNr[1]:=16#FF; (* Case gestione *)
59      END_IF;
60
61      (* ------------------------------------------------------------------- *)
62      (* TEMPORIZZAZIONE TRA CASES PROGRAMMA                                 *)
63      (* ------------------------------------------------------------------- *)
64      (* Siccome non è possibile leggere dal display il segnale di busy, viene *)
65      (* eseguita una temporizzazione tra i vari cases pari al tempo necessario *)
66      (* al display per eseguire il comando più lento (Clear display 760 uS). *)
67
68      IF (CaseNr[0] <> CaseNr[1]) THEN
69          IF ((SysGetSysTime(TRUE)-TimeBf) < 1000) THEN RETURN; END_IF;
70          TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
71          CaseNr[1]:=CaseNr[0]; (* Case gestione *)
72      END_IF;
73
74      (* ------------------------------------------------------------------- *)
75      (* GESTIONE CASE PROGRAMMA                                             *)
76      (* ------------------------------------------------------------------- *)
77      (* Eseguo gestione case LCD. *)
78
79      CASE (CaseNr[0]) OF
80
81          (* ------------------------------------------------------------- *)
82          (* INIZIALIZZAZIONE DISPLAY                                      *)
83          (* ------------------------------------------------------------- *)
84          (* Eseguo comando "Default function set". *)
85
86          0, 1, 2:
87          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
88          DWrBuffer[1]:=16#30; (* Default function set *)
89          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
90          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
91
92          (* ------------------------------------------------------------- *)
93          (* Eseguo comando "Function set DL, N, F". *)
```

```
 94
 95          3:
 96          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
 97          DWrBuffer[1]:=16#38; (* 8 bits, 2 lines, 5*7 dots *)
 98          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
 99          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
100
101          (* ---------------------------------------------------------------- *)
102          (* Eseguo comando "Display on/off control". *)
103
104          4:
105          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
106          DWrBuffer[1]:=16#08; (* Display off *)
107          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
108          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
109
110          (* ---------------------------------------------------------------- *)
111          (* Eseguo comando "Clear display". *)
112
113          5:
114          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
115          DWrBuffer[1]:=16#01; (* Clear display e cursor home *)
116          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
117          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
118
119          (* ---------------------------------------------------------------- *)
120          (* Eseguo comando "Cursor and display shift". *)
121
122          6:
123          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
124          DWrBuffer[1]:=16#06; (* Cursor increment *)
125          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
126          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
127
128          (* ---------------------------------------------------------------- *)
129          (* Eseguo comando "Display on/off control". *)
130
131          7:
132          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
133          DWrBuffer[1]:=16#0C; (* Display on *)
134          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
135          CaseNr[0]:=100; (* Case gestione *)
136
137          (* ---------------------------------------------------------------- *)
138          (* SCRITTURA DATI SU RIGA SUPERIORE                                 *)
139          (* ---------------------------------------------------------------- *)
140          (* Eseguo comando "Set DDRAM address". *)
141
142          100:
143          IF NOT(@HMIBuiltInID.DUpdated) THEN RETURN; END_IF;
144          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
145          DWrBuffer[1]:=16#80; (* DDRAM address:=0 *)
146          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
147          DDIDx:=0; (* Display data index *)
148          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
149
150          (* ---------------------------------------------------------------- *)
151          (* Eseguo scrittura dato sul display. Il tempo necessario è 18.5 uS *)
152          (* quindi non eseguo temporizzazione, il tempo di loop del programma *)
153          (* PLC sarà sicuramente maggiore. *)
```

```
154
155         101:
156         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
157         DWrBuffer[0]:=16#40; (* Co:=FALSE, A0:=TRUE *)
158         DWrBuffer[1]:=@Ptr; (* Dato display *)
159         IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
160
161         (* Eseguo controllo se terminato tutta la riga superiore. *)
162
163         DDIDx:=DDIDx+1; (* Display data index *)
164         IF (DDIDx > 15) THEN CaseNr[0]:=110; END_IF;
165
166         (* ---------------------------------------------------------------- *)
167         (* SCRITTURA DATI SU RIGA INFERIORE                                 *)
168         (* ---------------------------------------------------------------- *)
169         (* Eseguo comando "Set DDRAM address". *)
170
171         110:
172         DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
173         DWrBuffer[1]:=16#C0; (* DDRAM address:=0 *)
174         IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
175         DDIDx:=16; (* Display data index *)
176         CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
177
178         (* ---------------------------------------------------------------- *)
179         (* Eseguo scrittura dato sul display. Il tempo necessario è 18.5 uS *)
180         (* quindi non eseguo temporizzazione, il tempo di loop del programma *)
181         (* PLC sarà sicuramente maggiore. *)
182
183         111:
184         Ptr:=TO_UDINT(@HMIBuiltInID.DData)+DDIDx; (* Auxiliary pointer *)
185         DWrBuffer[0]:=16#40; (* Co:=FALSE, A0:=TRUE *)
186         DWrBuffer[1]:=@Ptr; (* Dato display *)
187         IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
188
189         (* Eseguo controllo se terminato tutta la riga inferiore. *)
190
191         DDIDx:=DDIDx+1; (* Display data index *)
192         IF (DDIDx > 31) THEN @HMIBuiltInID.DUpdated:=FALSE; CaseNr[0]:=100; END_IF;
193     ELSE
194         CaseNr[0]:=0; (* Case gestione *)
195     END_CASE;
196
197 (* [End of file] *)
198
```

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 : LCDCustom	
	FUNCTION : MemSet	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:13/12/2013
	Note :	Page:1 of 1