PROGRAM LCDMessages

```
VAR
LCD : MC21605LCDMng; (* LCD management FB *)
LCDMessage : STRING[ 32 ]; (* LCD message buffer *)
TimeBf : ARRAY[ 0..1 ] OF UDINT; (* Time buffer (uS) *)
MSelector : USINT; (* Selettore messaggi display *)
RVar : ARRAY[ 0..1 ] OF REAL := 125.23,15.5; (* Variabili REAL da visualizzare *)
i : UDINT; (* Variabile di appoggio *)
DTime : SysETimeToDate; (* Conversione in Data/Ora *)
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.DData:=ADR(LCDMessage); (* Display data buffer pointer *)
    LCD(Enable:=TRUE); (* Gestione FB *)
    LCD.Write:=FALSE; (* Display write command *)

    (* Gestione timer comando write, aggiorno dispaly ogni 100 mS. *)

    IF ((SysGetSysTime(TRUE)-TimeBf[0]) < 100000) THEN RETURN; END_IF;
    TimeBf[0]:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
    LCD.Write:=TRUE; (* Display write command *)

    (* ---------------------------------------------------------------- *)
    (* GESTIONE ROTAZIONE MESSAGGI DISPLAY                              *)
    (* ---------------------------------------------------------------- *)
    (* Ogni 2 secondi vario il messaggio visualizzato sul display. *)

    IF ((SysGetSysTime(TRUE)-TimeBf[1]) > 2000000) THEN
        TimeBf[1]:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
        MSelector:=MSelector+1; (* Selettore messaggi display *)
    END_IF;

    (* ---------------------------------------------------------------- *)
    (* GESTIONE MESSAGGI DISPLAY                                        *)
    (* ---------------------------------------------------------------- *)
    (* Gestione selettore messaggio. *)

    CASE (MSelector) OF

        (* ---------------------------------------------------------- *)
        (* Esempio messaggio solo testo. *)
        (* +----------------+ *)
        (* |Hello !         | *)
        (* |I am a SlimLine!| *)
        (* +----------------+ *)

        0:
```

	Project : LCDMessages	
	PROGRAM : LCDMessages	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:26/09/2013
	Note :	Page:1 of 2

```
48          LCDMessage:='Hello !         I am a SlimLine!'; (* LCD message buffer *)
49
50          (* ------------------------------------------------------------------ *)
51          (* Esempio messaggio con 2 variabili REAL. *)
52          (* +----------------+ *)
53          (* |Real 1:xxx.xxx  | *)
54          (* |Real 2:xx.x     | *)
55          (* +----------------+ *)
56
57          1:
58          i:=MemSet(ADR(LCDMessage), 0, SIZEOF(LCDMessage));
59          i:=SysVarsnprintf(ADR(LCDMessage), 16+1, 'Real 1:%7.3f  ', REAL_TYPE, ADR(RVar[0]));
60          i:=SysVarsnprintf(ADR(LCDMessage)+16, 16+1, 'Real 2:%4.1f     ', REAL_TYPE, ADR(RVar[1]));
61
62          (* ------------------------------------------------------------------ *)
63          (* Esempio messaggio con visualizzazione Data/Ora. *)
64          (* +----------------+ *)
65          (* |Date:xx/xx/xx   | *)
66          (* |Time:xx:xx:xx   | *)
67          (* +----------------+ *)
68
69          2:
70          DTime(EpochTime:=SysDateTime); (* Conversione in Data/Ora *)
71          DTime.Year:=DTime.Year-2000; (* Year *)
72
73          (* Eseguo report valori data/ora. *)
74
75          i:=MemSet(ADR(LCDMessage), 0, SIZEOF(LCDMessage));
76          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 8+1, 'Date:%02d/', USINT_TYPE, ADR(DTime.Day
    ));
77          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 3+1, '%02d/', USINT_TYPE, ADR(DTime.Month));
78          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 5+1, '%02d   ', USINT_TYPE, ADR(DTime.Year)
    ;
79
80          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 8+1, 'Time:%02d:', USINT_TYPE, ADR(DTime.Hou
    r));
81          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 3+1, '%02d:', USINT_TYPE, ADR(DTime.Minute)
    ;
82          i:=SysVarsnprintf(ADR(LCDMessage)+LEN(LCDMessage), 5+1, '%02d   ', USINT_TYPE, ADR(DTime.Second
    ));
83      ELSE
84          MSelector:=0; (* Selettore messaggi display *)
85      END_CASE;
86
87  (* [End of file] *)
88
89
```

	Project : LCDMessages	
	PROGRAM : LCDMessages	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:26/09/2013
	Note :	Page:2 of 2

```
VAR_INPUT
Enable : BOOL; (* FB enable *)
Write : BOOL; (* Display write command *)
I2CAddress : USINT; (* LCD I2C address *)
DData : @USINT; (* Display data buffer pointer *)
END_VAR

VAR_OUTPUT
Enabled : BOOL; (* FB enabled *)
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 *)
IWrite : BOOL; (* Display write command (Internal) *)
END_VAR
```

```
 1 (* ********************************************************************** *)
 2 (* FUNCTION BLOCK "MC21605LCDMng"                                         *)
 3 (* ********************************************************************** *)
 4 (* Questo blocco funzione gestisce il display LCD MC21605 della Midas.    *)
 5 (* ---------------------------------------------------------------------- *)
 6
 7     (* ---------------------------------------------------------------- *)
 8     (* GESTIONE ABILITAZIONE                                            *)
 9     (* ---------------------------------------------------------------- *)
10     (* Gestione abilitazione blocco funzione. *)
11
12     IF NOT(Enable) THEN Enabled:=FALSE; RETURN; END_IF;
13
14     (* Gestione primo loop abilitazione. *)
15
16     IF NOT(Enabled) THEN
17         Enabled:=TRUE; (* FB enabled *)
18         CaseNr[0]:=16#00; (* Case gestione *)
19         CaseNr[1]:=16#FF; (* Case gestione *)
20     END_IF;
21
22     (* ---------------------------------------------------------------- *)
23     (* GESTIONE COMNDO WRITE                                            *)
24     (* ---------------------------------------------------------------- *)
25     (* Controllo se attivo il comando write. *)
26
27     IF (Write) THEN IWrite:=TRUE; END_IF;
28
29     (* ---------------------------------------------------------------- *)
30     (* TEMPORIZZAZIONE TRA CASES PROGRAMMA                              *)
31     (* ---------------------------------------------------------------- *)
32     (* Siccome non è possibile leggere dal display il segnale di busy, viene *)
33     (* eseguita una temporizzazione tra i vari cases pari al tempo necessario *)
34     (* al display per eseguire il comando più lento (Clear display 760 uS). *)
35
36     IF (CaseNr[0] <> CaseNr[1]) THEN
37         IF ((SysGetSysTime(TRUE)-TimeBf) < 1000) THEN RETURN; END_IF;
```

```
38          TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
39          CaseNr[1]:=CaseNr[0]; (* Case gestione *)
40      END_IF;
41
42      (* ---------------------------------------------------------------- *)
43      (* GESTIONE CASE PROGRAMMA                                          *)
44      (* ---------------------------------------------------------------- *)
45      (* Eseguo gestione case LCD. *)
46
47      CASE (CaseNr[0]) OF
48
49          (* ---------------------------------------------------------------- *)
50          (* INIZIALIZZAZIONE DISPLAY                                         *)
51          (* ---------------------------------------------------------------- *)
52          (* Eseguo comando "Default function set". *)
53
54          0, 1, 2:
55          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
56          DWrBuffer[1]:=16#30; (* Default function set *)
57          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
58          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
59
60          (* ---------------------------------------------------------------- *)
61          (* Eseguo comando "Function set DL, N, F". *)
62
63          3:
64          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
65          DWrBuffer[1]:=16#38; (* 8 bits, 2 lines, 5*7 dots *)
66          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
67          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
68
69          (* ---------------------------------------------------------------- *)
70          (* Eseguo comando "Display on/off control". *)
71
72          4:
73          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
74          DWrBuffer[1]:=16#08; (* Display off *)
75          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
76          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
77
78          (* ---------------------------------------------------------------- *)
79          (* Eseguo comando "Clear display". *)
80
81          5:
82          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
83          DWrBuffer[1]:=16#01; (* Clear display e cursor home *)
84          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
85          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
86
87          (* ---------------------------------------------------------------- *)
88          (* Eseguo comando "Cursor and display shift". *)
89
90          6:
91          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
92          DWrBuffer[1]:=16#06; (* Cursor increment *)
93          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
94          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
95
96          (* ---------------------------------------------------------------- *)
97          (* Eseguo comando "Display on/off control". *)
```

```
 98
 99          7:
100          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
101          DWrBuffer[1]:=16#0C; (* Display on *)
102          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
103          CaseNr[0]:=100; (* Case gestione *)
104
105          (* ------------------------------------------------------------------ *)
106          (* SCRITTURA DATI SU RIGA SUPERIORE                                   *)
107          (* ------------------------------------------------------------------ *)
108          (* Eseguo comando "Set DDRAM address". *)
109
110          100:
111          IF NOT(IWrite) THEN RETURN; END_IF;
112          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
113          DWrBuffer[1]:=16#80; (* DDRAM address:=0 *)
114          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
115          DDIDx:=0; (* Display data index *)
116          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
117
118          (* ------------------------------------------------------------------ *)
119          (* Eseguo scrittura dato sul display. Il tempo necessario è 18.5 uS *)
120          (* quindi non eseguo temporizzazione, il tempo di loop del programma *)
121          (* PLC sarà sicuramente maggiore. *)
122
123          101:
124          Ptr:=TO_UDINT(DData)+DDIDx; (* Auxiliary pointer *)
125          DWrBuffer[0]:=16#40; (* Co:=FALSE, A0:=TRUE *)
126          DWrBuffer[1]:=@Ptr; (* Dato display *)
127          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
128
129          (* Eseguo controllo se terminato tutta la riga superiore. *)
130
131          DDIDx:=DDIDx+1; (* Display data index *)
132          IF (DDIDx > 15) THEN CaseNr[0]:=110; END_IF;
133
134          (* ------------------------------------------------------------------ *)
135          (* SCRITTURA DATI SU RIGA INFERIORE                                   *)
136          (* ------------------------------------------------------------------ *)
137          (* Eseguo comando "Set DDRAM address". *)
138
139          110:
140          DWrBuffer[0]:=16#80; (* Co:=TRUE, A0:=FALSE *)
141          DWrBuffer[1]:=16#C0; (* DDRAM address:=0 *)
142          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
143          DDIDx:=16; (* Display data index *)
144          CaseNr[0]:=CaseNr[0]+1; (* Case gestione *)
145
146          (* ------------------------------------------------------------------ *)
147          (* Eseguo scrittura dato sul display. Il tempo necessario è 18.5 uS *)
148          (* quindi non eseguo temporizzazione, il tempo di loop del programma *)
149          (* PLC sarà sicuramente maggiore. *)
150
151          111:
152          Ptr:=TO_UDINT(DData)+DDIDx; (* Auxiliary pointer *)
153          DWrBuffer[0]:=16#40; (* Co:=FALSE, A0:=TRUE *)
154          DWrBuffer[1]:=@Ptr; (* Dato display *)
155          IF NOT(SysI2CWrRd(I2CAddress,02, ADR(DWrBuffer), 0, 0)) THEN CaseNr[0]:=0; RETURN; END_IF;
156
157          (* Eseguo controllo se terminato tutta la riga inferiore. *)
```

```
158
159         DDIDx:=DDIDx+1; (* Display data index *)
160         IF (DDIDx > 31) THEN IWrite:=FALSE; CaseNr[0]:=100; END_IF;
161     ELSE
162         CaseNr[0]:=0; (* Case gestione *)
163     END_CASE;
164
165 (* [End of file] *)
166
167
```

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 : LCDMessages	
	FUNCTION : MemSet	
	Release : LCDMessage	Ver :1.00
	Author : Sergio Bertana	Date:26/09/2013
	Note :	Page:1 of 1