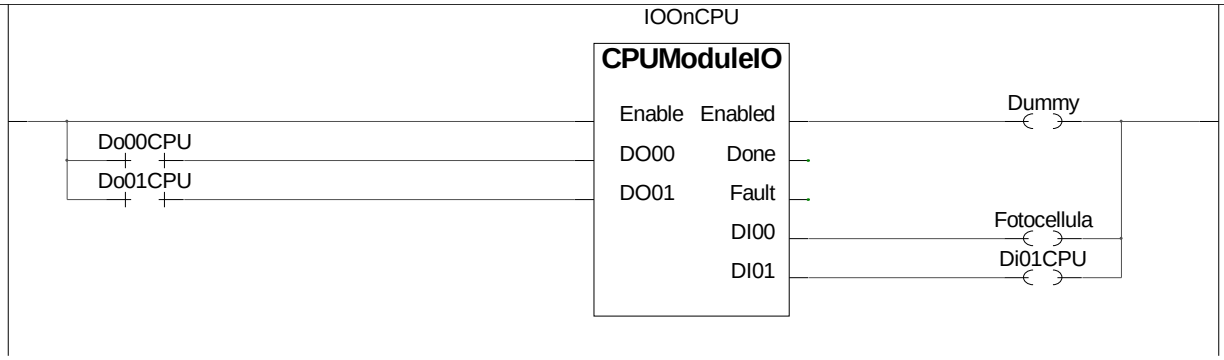
```
VAR_GLOBAL
Fotocellula : BOOL; (* Ingresso fotocellula *)
SacchiPiccoli AT %MW100.16 : UINT; (* Numero sacchi piccoli *)
SacchiGrandi AT %MW100.18 : UINT; (* Numero sacchi grandi *)
Di01CPU : BOOL; (* Di01 su modulo CPU *)
Do00CPU : BOOL; (* Do00 su modulo CPU *)
Do01CPU : BOOL; (* Do01 su modulo CPU *)
END_VAR
```

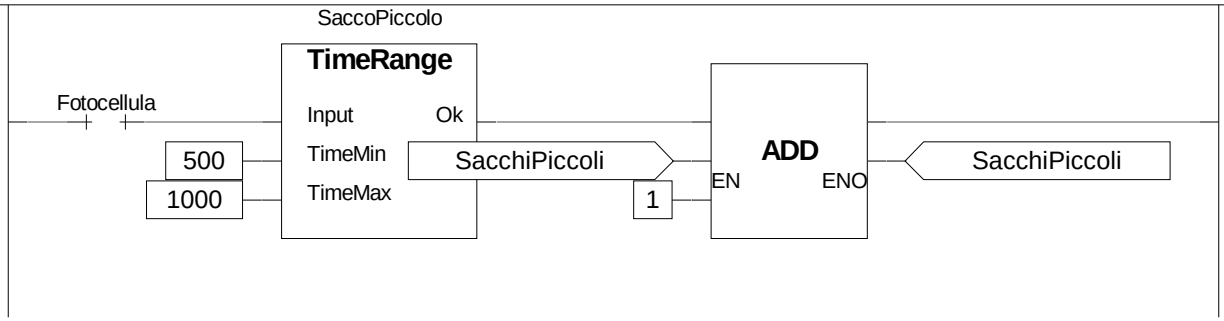
PROGRAM BagCounter

```
VAR
SaccoPiccolo : TimeRange; (* FB controllo sacco piccolo *)
SaccoGrande : TimeRange; (* FB controllo sacco grande *)
IOOnCPU : CPUModuleIO; (* FB acquisizione I/O su modulo CPU *)
Dummy : BOOL; (* Dummy variable *)
END_VAR
```

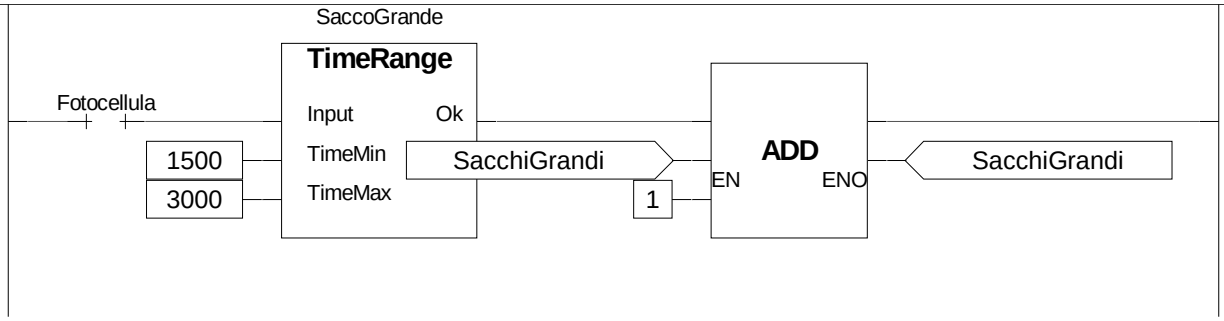
1



2



3



	Project : BagCounter	
	PROGRAM : BagCounter	
	Release : BagCounter	Ver :1.00
	Author :	Date:17/04/2012
	Note :	Page:1 of 1

```
VAR
Fp : FILEP; (* File pointer *)
SktLsn : SysSktListen; (* FB socket listen *)
WMsgPulse : BOOL; (* Welcome message pulse *)
i : INT; (* Auxiliary variable *)
CaseNr : USINT; (* Program cases *)
RxString : STRING[ 32 ]; (* Rx command string *)
Ptr : @USINT; (* Auxiliary pointer *)
Output : USINT; (* Output to set *)
aaa : UINT;
END_VAR
```

```
(* ********************************************************************** *)
(* GESTIONE INTERFACCIA SU CONNESSIONE TCP                                *)
(* ********************************************************************** *)
(* Questo programma esegue gestione di connessione su TCP.                *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------------- *)
    (* ESEGUO INIZIALIZZAZIONI                                                *)
    (* ---------------------------------------------------------------------- *)
    (* Eseguo apertura socket TCP. *)

    IF (SysFirstLoop) THEN
        Fp:=Sysfopen('TCPSKT', 'rw'); (* File pointer *)
    END_IF;

    (* ---------------------------------------------------------------------- *)
    (* GESTIONE CONTROLLO STATO SOCKET TCP                                    *)
    (* ---------------------------------------------------------------------- *)
    (* Forzo socket in condizione di listening. *)
    (* "LifeTm" piccolo, forza chiusura socket su mancanza comunicazione. *)
    (* "FlushTm" deve essere più grande del tempo di loop programma. *)

    SktLsn.File:=Fp; (* Flusso dati stream *)
    SktLsn.MyPort:=2000; (* Porta in ascolto su socket *)
    SktLsn.LifeTm:=60; (* Tempo di vita socket (S) *)
    SktLsn.FlushTm:=10; (* Tempo di flush socket (mS) *)
    SktLsn.RxSize:=200; (* Dimensione buffer Rx dati socket *)
    SktLsn.TxSize:=200; (* Dimensione buffer Tx dati socket *)
    SktLsn(Enable:=TRUE); (* Forzo socket in listening *)

    (* ---------------------------------------------------------------------- *)
    (* GESTIONE CONNESSIONE TCP                                               *)
    (* ---------------------------------------------------------------------- *)
    (* Su connessione TCP il sistema invia un messaggio di Welcome. *)

    IF (SktLsn.Connect <> WMsgPulse) THEN
        WMsgPulse:=SktLsn.Connect; (* Welcome message pulse *)

        IF (SktLsn.Connect) THEN
            i:=SysVarfprintf(Fp, '%s$r$n', STRING_TYPE, ADR('Welcome !'));
        END_IF;
    END_IF;

    (* Se nessun client TCP è connesso esco. *)

```

```
46      IF NOT(SktLsn.Connect) THEN RETURN; END_IF;
47
48      (* ---------------------------------------------------------------- *)
49      (* GESTIONE PROTOCOLLO COMUNICAZIONE                                *)
50      (* ---------------------------------------------------------------- *)
51      (* Viene gestito un semplice protocollo ascii per la lettura contatori e  *)
52      (* per il reset. Il protocollo può essere modificato secondo le esigenze. *)
53      (*                                                                  *)
54      (* Inviando la stringa ?C<CR>, viene ritornato valore dei due counters.   *)
55      (* Inviando la stringa !C<CR>, viene azzerato valore dei due counters.    *)
56      (* ---------------------------------------------------------------- *)
57      (* Eseguo loop su ricezione caratteri dal client TCP. *)
58
59      CASE (CaseNr) OF
60
61          (* ---------------------------------------------------------------- *)
62          (* Inizializzo ricezione stringa comando. *)
63
64          0:
65          Ptr:=ADR(RxString); (* Auxiliary pointer *)
66          CaseNr:=CaseNr+1; (* Program cases *)
67
68          (* ---------------------------------------------------------------- *)
69          (* Ricezione stringa fino al <CR>. *)
70
71          1:
72          WHILE (TO_BOOL(SysGetIChars(Fp))) DO
73              @Ptr:=TO_USINT(Sysfgetc(Fp)); (* Character read *)
74              IF (@Ptr = 16#0D) THEN i:=TO_INT(SysFIBfClear(Fp)); CaseNr:=10; RETURN; END_IF;
75
76              (* Incremento puntatore stringa ricezione e controllo range. *)
77
78              Ptr:=Ptr+1; (* Auxiliary pointer *)
79              IF (Ptr >= ADR(RxString)+32) THEN
80
81                  (* Errore stringa ricevuta troppo lunga. *)
82
83                  i:=SysVarfprintf(Fp, '%s$r$n', STRING_TYPE, ADR('Command too long'));
84                  i:=TO_INT(SysFIBfClear(Fp)); (* svuoto buffer ricezione *)
85                  CaseNr:=0; (* Program cases *)
86                  RETURN;
87              END_IF;
88          END_WHILE;
89
90          (* ---------------------------------------------------------------- *)
91          (* Controllo su comando ricevuto. *)
92
93          10:
94          IF (FIND(RxString, '?C$r') > 0) THEN CaseNr:=20; RETURN; END_IF;
95          IF (FIND(RxString, '!C$r') > 0) THEN CaseNr:=30; RETURN; END_IF;
96          i:=SysVarfprintf(Fp, '%s$r$n', STRING_TYPE, ADR('Command error'));
97          CaseNr:=0; (* Program cases *)
98
99          (* ---------------------------------------------------------------- *)
100         (* GESTIONE LETTURA VALORE CONTATORI                                *)
101         (* ---------------------------------------------------------------- *)
102         (* Ritorno valore dei due contatori. *)
103
104         20:
105         i:=SysVarfprintf(Fp, 'Piccoli:%d, ', UINT_TYPE, ADR(SacchiPiccoli));
```

```
106         i:=SysVarfprintf(Fp, 'Grandi:%d$r$n', UINT_TYPE, ADR(SacchiGrandi));
107         CaseNr:=0; (* Program cases *)
108 
109         (* ---------------------------------------------------------------- *)
110         (* Eseguo reset valore contatori. *)
111 
112         30:
113         SacchiPiccoli:=0; (* Numero sacchi grandi *)
114         SacchiGrandi:=0; (* Numero sacchi grandi *)
115         i:=SysVarfprintf(Fp, '%s$r$n', STRING_TYPE, ADR('Reset Ok'));
116         CaseNr:=0; (* Program cases *)
117 
118         (* ---------------------------------------------------------------- *)
119         (* Errore cases gestione. *)
120 
121         ELSE
122             i:=SysVarfprintf(Fp, '%s$r$n', STRING_TYPE, ADR('Case error'));
123             CaseNr:=0; (* Program cases *)
124     END_CASE;
125 
126 (* [End of file] *)
127 
```

```
VAR_INPUT
Input : BOOL; (* Input to detect *)
TimeMin : UINT; (* Minimum time (S) *)
TimeMax : UINT; (* Maximum time (S) *)
END_VAR

VAR_OUTPUT
Ok : BOOL; (* Time in range *)
END_VAR

VAR_EXTERNAL
SysTime : UDINT; (* System time (mS) *)
END_VAR

VAR
Pulse : BOOL; (* Pulse detection *)
TimeBf : UDINT; (* Buffer gestione tempo *)
TimeActive : UDINT; (* Tempo attivazione ingresso (mS) *)
END_VAR
```

```
(* ********************************************************************** *)
(* FUNCTION BLOCK "TimeRange"                                             *)
(* ********************************************************************** *)
(* Questo blocco funzione controlla se l'ingresso rimane attivo per un tempo  *)
(* compreso tra i valori "Min" e "Max". Se compreso si attiva per un loop     *)
(* l'uscia di "Ok".                                                       *)
(* ---------------------------------------------------------------------- *)
    (* Eseguo reset uscite impulsive. *)

    Ok:=FALSE; (* Time in range *)

    (* Eseguo controllo se variazione stato ingresso. *)

    IF (Input = Pulse) THEN RETURN; END_IF;
    Pulse:=Input; (* Pulse detection *)

    (* Controllo attivazione ingresso, su attivazione salvo tempo. *)

    IF (Input) THEN
        TimeBf:=SysTime; (* Buffer gestione tempo *)
    END_IF;

    (* Controllo disattivazione ingresso, su disattivazione controllo *)
    (* tempo trascorso e gestisco counter relativo. *)

    IF NOT(Input) THEN
        TimeActive:=SysTime-TimeBf; (* Tempo attivazione ingresso (mS) *)

        (* Controllo se tempo di attivazione compreso tra valori di soglia. *)

        IF (TimeActive >= TimeMin) AND (TimeActive <= TimeMax)THEN
            Ok:=TRUE; (* Time in range *)
        END_IF;
    END_IF;

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

38

FUNCTION_BLOCK CPUModuleIO

(SFR054B000) Manages the logic I/O on the CPU module
ENCRYPTED CODE

```
VAR_INPUT
Enable : BOOL; (* Function enable *)
DO00 : BOOL; (* Digital output 0 *)
DO01 : BOOL; (* Digital output 1 *)
END_VAR

VAR_OUTPUT
Enabled : BOOL; (* Function enabled *)
Done : BOOL; (* Function done *)
Fault : BOOL; (* Function fault *)
DI00 : BOOL; (* Digital input 0 *)
DI01 : BOOL; (* Digital input 0 *)
END_VAR
```

1

	Project : BagCounter	
	FUNCTION BLOCK : CPUModuleIO	
	Release : BagCounter	Ver :1.00
	Author :	Date:17/04/2012
	Note :	Page:1 of 1