

```

VAR
  Tm : TIMERSDATASTRUCT; (* Struttura timers *)
  i : USINT; (* Auxiliary variable *)
  Ptr : @UDINT; (* Auxiliary pointer *)
  TReference : UDINT; (* Time reference (uS) *)
END_VAR

```

```

1 (* ***** *)
2 (* PROGRAM "ST" *)
3 (* ***** *)
4 (* Questo programma illustra l'utilizzo di una struttura di timers. *)
5 (* Per gestire le temporizzazioni si utilizza il valore di tempo di sistema *)
6 (* ritornato dalla funzione SysGetSysTime, è un un UDINT che si incrementa *)
7 (* ogni uS, al valore massimo esegue overflow a zero, non è possibile quindi *)
8 (* effettuare comparazioni dirette con il tempo di riferimento ma occorre *)
9 (* sempre eseguire la differenza. *)
10 (*)
11 (* In questo esempio specifico utilizzo una struttura dati che contiene tutti *)
12 (* i tempi di riferimento per mostrare come eseguire ciclo FOR di iterazione. *)
13 (* ----- *)
14
15 (* ----- *)
16 (* INIZIALIZZAZIONE *)
17 (* ----- *)
18 (* Al primo loop di programma inizializzo tutti i tempi di rferimento. *)
19
20 IF (SysFirstLoop) THEN
21   Ptr:=ADR(Tm); (* Auxiliary pointer *)
22   TReference:=SysGetSysTime(TRUE); (* Time reference (uS) *)
23
24   FOR i:=0 TO ((SIZEOF(Tm)/4)-1) DO
25     @Ptr:=TReference;
26     Ptr:=Ptr+4; (* Auxiliary pointer *)
27   END_FOR;
28 END_IF;
29
30 (* ----- *)
31 (* LAMPEGGIO USCITE *)
32 (* ----- *)
33 (* Eseguo lampeggio uscite logiche. *)
34
35 IF ((SysGetSysTime(TRUE)-Tm.TDo00) >= 500000) THEN
36   Tm.TDo00:=SysGetSysTime(TRUE); (* Tempo lampeggio (Do00) *)
37   Do00CPU:=NOT(Do00CPU); (* Output 00 modulo CPU *)
38 END_IF;
39
40 IF ((SysGetSysTime(TRUE)-Tm.TDo01) >= 1000000) THEN
41   Tm.TDo01:=SysGetSysTime(TRUE); (* Tempo lampeggio (Do01) *)
42   Do01CPU:=NOT(Do01CPU); (* Output 01 modulo CPU *)
43 END_IF;
44
45 (* [End of file] *)
46

```

Project : TimerStruct

PROGRAM : ST

Release : Xtarget

Ver :1.00

Author :

Date:15/03/2016

Note :

Page:1 of 1