FstTimer

JEDNOTKA PRO PŘESNÉ ODMĚŘOVÁNÍ ČASOVÝCH INTERVALŮ

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 17.06.2005
Datum posledního uložení dokumentu: 17.06.2005
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Wi	09.06.2003	První vydání.
1.10	1.XX	Wil	17.06.2005	Úpravy podle nové knihovny HwSyst: – některé globální proměnné o aktuálním nastavení systémového časovače byly přesunuty do knihovny HwSyst. – Zrušena funkce GetTimer0, kterou nahrazuje funkce ReadTimer0 z HwSyst. – Zrušena procedura RefreshIRQVals, její volání se může z aplikací odstranit.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro přesné odměřování časových intervalů pomocí procedury volané v **UserTick1** (procedura zrychleného časovače, který je spravován knihovnou **Tick**).

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je nutno seznámit se s popisem knihovny Tick a je vhodné seznámit se i s popisem knihovny Timer.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

Jednotka **FstTimer** definuje objekt **tFstTimer**, jehož instance slouží k přesnému odměřování časových intervalů. Jednotka je obdobou jednotky **Timer** s tím rozdílem, že pro odměřování časových intervalů nevychází ze standardního systémového časovače, který standardně „tiká“ s periodou 55ms, ale ze zrychleného časovače, který si uživatel nastaví prostřednictvím knihovny **Tick** a přímým čtením HW časovače. Tím lze dosáhnout podstatně přesnějšího odměřování časů (rozlišení 1ms), než při použití klasické jednotky **Timer**.

Při měření času se tedy vychází jednak z počtu případného zrychleného přerušení (zrychlené přerušení obsluhuje knihovna **Tick**) a jednak z přímého čtení HW časovače.

4. Popis konstant a typů

```
cVerNo = např. $0101; { BCD formát }
cVer   = např. '01.01,17.03.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

5. Objekt tFstTimer

```
pFstTimer = ^tFstTimer;
tFstTimer = object(TObject)
```

Objekt **tFstTimer** definuje metody, které zajišťují odměřování časových intervalů. Metody lze rozdělit podle způsobu použití do dvou skupin. Jednu skupinu tvoří metody **SaveTime**, **TstTime** a **ReadTime** a druhou skupinu tvoří metody **SetCount**, **TstCount** a **ReadCount**. Příklad použití metod z první skupiny: Voláním metody **SaveTime** je zaznamenána hodnota aktuálního času a periodickým voláním metody **TstTime** se otestuje, zda požadovaný interval již uplynul, popřípadě voláním metody **ReadTime** se zjistí již uplynulý čas od zaznamenání. Příklad použití metod z druhé skupiny: Voláním metody **SetCount** je zaznamenána hodnota aktuálního času a zároveň je nastaven požadovaný interval. Dále periodickým voláním metody **TstCount** se otestuje, zda požadovaný interval již uplynul, popřípadě voláním metody **ReadCount** se zjistí již uplynulý čas od zaznamenání. Metody obou skupin používají pro svou činnost odlišné lokální proměnné, proto je lze v aplikaci používat současně ve více odměřovacích algoritmech. V jednom odměřovacím algoritmu však musí být použity metody jen jedné skupiny.

5.1. Metody

5.1.1. SaveTime procedura

```
procedure SaveTime;
```

Metoda **SaveTime** zaznamená hodnotu aktuálního času. Hodnota takto zaznamenaného času je poté využívaná např. metodou **TstTime**.

5.1.2. TstTime funkce

```
function  TstTime(Ms:LongInt): Boolean;
```

Metoda **TstTime** provádí test, zda uplynulo **Ms** milisekund od času zaznamenaného metodou **SaveTime**. Je-li časový interval od volání metody **SaveTime** delší nebo roven **Ms**, vrátí metoda **TstTime** hodnotu True, v opačném případě vrátí hodnotu False.

5.1.3. ReadTime funkce

```
function  ReadTime: Longint;
```

Metoda **ReadTime** vrátí počet milisekund, které uběhly od zaznamenaného času metodou **SaveTime**.

5.1.4. SetCount procedura

```
procedure SetCount(Ms:Longint);
```

Metoda **SetCount** zaznamená hodnotu aktuálního času a zároveň provede nastavení časového intervalu na hodnotu **Ms** milisekund. Hodnota takto zaznamenaného času a nastavený interval je poté využívaná např. metodou **TstCount**.

5.1.5. TstCount funkce

```
function  TstCount: Boolean;
```

Metoda **TstCount** provádí test, zda nastavený interval od času zaznamenaného metodou **SetCount** již uplynul. Je-li časový interval od volání metody **SetCount** delší nebo roven nastavenému intervalu, vrátí metoda **TstCount** hodnotu True, v opačném případě vrátí hodnotu False.

5.1.6. ReadCount funkce

```
function  ReadCount: Longint;
```

Metoda **ReadCount** vrátí počet milisekund, které uběhly od zaznamenaného času metodou **SetCount**.

6. Příklad

```
uses
  Tick,
  FstTimer;

var
  MyTimer : tFstTimer;

begin
{práce s casovacem}
  MyTimer.SaveTime;
  Repeat until MyTimer.TstTime(100); {cekaci smycka 100ms s presnosti
                                      zhruba 0.001ms}
```

```
  {jiny zpusob}
  MyTimer.SetCount(100);
  Repeat until MyTimer.TstCount;
end.
```