uTermT10

JEDNOTKA IMPLEMENTUJÍCÍ TERMINÁL TERM10

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: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Upozornění:
V dokumentu použité názvy výrobků, firem apod. mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Če		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky implementující terminál Term10.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem ChnVirt, uATerm, uTermGr, uKeybT10, uDispT10 a G240x128.

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 implementuje funkce konkrétního grafického terminálu TERM10. Terminál TERM10 je vybaven alfanumerickou membránovou klávesnicí a grafickým LCD displejem o rozměru 240 × 128 pixelů podsvětleným výbojkou s možností programového řízení jasu. Terminál je dále vybaven osmi signalizačními LED na předním panelu a piezoelektrickým měničem pro zvukovou signalizaci.

Funkce terminálu jsou implementovány v objektovém typu **tTermT10**. Tento objekt dědí veškeré funkce obecného grafického terminálu **tTermGr** a nově implementuje metody pro obsluhu specifických funkcí hardware terminálu (signalizační LED, jas podsvětlení, kontrast displeje, ovládání zvukové signalizace).

Kromě alfanumerických a funkčních kláves, které jsou obsluhovány standardním způsobem přes buffer klávesnice, je terminál vybaven tlačítky "START" a "STOP", jejichž obsluha je odlišná. Uživatel má možnost instalovat pro každé tlačítko vlastní proceduru, která je provedena po jeho stisku. Je tak umožněno přímo ovlivnit řídící algoritmus bez ohledu na stav v systému menu.

Pro šetření životnosti výbojky podsvětlující displej je terminál vybaven funkcí programového řízení jejího jasu a funkcí screen saver. Tato funkce zhasne výbojku, pokud po určitou dobu není stisknuta žádná klávesa. Při následujícím stisku klávesy je pak výbojka opět rozsvícena. Klávesa, kterou byla výbojka rozsvícena, není vložena do bufferu klávesnice. Funkce tlačítek "START" a "STOP" však zůstává i v tomto případě zachována. Prodlevu pro tuto funkci lze programově nastavit a případně lze funkci vypnout.

Zděděné metody zde nejsou popsány, jejich popis je možno najít v dokumentaci k jednotkám **tATerm, tTermChr** a **tTermGr**.

4. Popis konstant a typů

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

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

```
BaseGr    = $0;
```

BaseGr je adresa počátku využívané videopaměti displeje.

```
aa        = $ff;
```

aa je konstanta, kterou se plní videopaměť displeje pro prázdnou obrazovku.

```
PDataLCD  = $0;
```

PDataLCD je offset adresy datového registru displeje.

```
PCmdLCD   = $1;
```

PCmdLCD je offset adresy příkazového registru displeje.

```
PCtrlL    = $2;
```

PCtrlL je offset adresy dolních 4 bitů příkazového registru pro komunikaci s procesorem PIC.

```
PCtrlH    = $3;
```

PCtrlHje offset adresy horních 4 bitů příkazového registru pro komunikaci s procesorem PIC.

```
PLedL      = $4;
```
PLedL je offset adresy dolních 4 bitů výstupu signalizačních LED.

```
PLedH      = $5;
```
PLedH je offset adresy dolních 4 bitů výstupu signalizačních LED.

```
PChrL      = $4;
```
PChrL je offset adresy dolních 4 bitů datového registru pro komunikaci s procesorem PIC.

```
PChrH      = $5;
```
PChrH je offset adresy dolních 4 bitů datového registru pro komunikaci s procesorem PIC.

```
PINOUT     = $6;
```
PINOUT je offset adresy optovýstupů a optovstupů.

```
PEOUT      = $7;
```
Zápisem na offset **PEOUT** se připojí optovýstupy.

```
PDOUT      = $8;
```
Zápisem na offset **PDOUT** se odpojí optovýstupy.

```
PRESOff    = $7;
```
Zápisem na offset **PRESOff** je deaktivován reset hardware terminálu.

```
PRESOn     = $8;
```
Zápisem na offset **PRESOn** je aktivován reset hardware terminálu.

```
PJC      = $9;
```
Konstanta **PJC** je ofset adresy portu pro ovládání jasu a kontrastu displeje.

```
PEFL0      = $9;
PDFL0      = $a;
PEFL1      = $b;
PDFL1      = $c;
PEFL2      = $d;
PDFL2      = $e;
```
Konstanty **PEFL0, PDFL0, PEFL1, PDFL1, PEFL2, PDFL2** jsou ofsety adres, kterými se ovládá jas podsvětlující výbojky.

```
tOnSetupT10=procedure(P:pTermT10);
```
Typ **tOnSetupT10** je typ procedurální proměnné s ukazatelem na proceduru implementující setup terminálu.

```
tUserProcT10=procedure(P:pTermT10);
```
Typ **tUserProcT10** je typ procedurální proměnné s ukazatelem na proceduru implementující reakci na stisk tlačítka "START" a "STOP".

5. Popis objektu tTermT10

Objektový typ **tTermT10** je potomkem typu **tTermGr**. Implementuje funkce konkrétního grafického terminálu TERM10 v návaznosti na jeho hardware. Terminál je vybaven grafickým LCD displejem o rozměru 240 × 128 pixelů podsvětleným výbojkou, jejíž jas lze programově řídit, membránovou klávesnicí, osmi signalizačními LED na čelním panelu, zvukovou signalizací, osmi opticky oddělenými digitálními vstupy a osmi opticky oddělenými digitálními výstupy. Navíc je v objektu implementována funkce screen saver pro automatické zhasnutí výbojky podsvětlující displej, není-li zadanou dobu stisknuto žádné tlačítko na terminálu.

5.1. Proměnné

`vAddr:Word;`

Proměnná **vAddr** uchovává bázovou adresu terminálu v IO prostoru.

`StOut:Byte;`

Proměnná **StOut** uchovává stav optovýstupů.

`StLight:Byte;`

Proměnná **StLight** uchovává velikost jasu displeje (0 až 4).

`StDelay:Byte;`

Proměnná **StDelay** obsahuje nastavenou délka prodlevy pro zhasínání přisvětlení v minutách, 0 = zhasínání přisvětlení vypnuto (funkce screen saver)

`FlDark:Boolean;`

Proměnná **FlDark** obsahuje true, je-li přisvětlení displeje zhasnuto funkcí screen saver.

`TimeLastKeyPressed:Integer;`

Proměnná **TimeLastKeyPressed** uchovává čas posledního stisku klávesy pro potřeby funkce screen saver.

`OnSetupTerminal:tOnSetupT10;`

Proměnná **OnSetupTerminal** obsahuje ukazatel na instalovanou proceduru implementující setup terminálu.

`UserStartProc:tUserProcT10;`

Proměnná **UserStartProc** obsahuje ukazatel na instalovanou proceduru implementující reakci na stisk tlačítka "START".

`UserStopProc:tUserProcT10;`

Proměnná **UserStopProc** obsahuje ukazatel na instalovanou proceduru implementující reakci na stisk tlačítka "STOP".

`FlRefreshOut:Boolean;`

Proměnná **FlRefreshOut** je true je-li povolena periodická obnova stavu optovýstupů.

5.2. Metody

5.2.1. Init

```
constructor Init(NewDisp:pADisp;NewKeyb:pAKeyb;
                 NAddr:Word;NewChnTerm:pChnVirt;
                 NewChnRecBuf:pointer);
```

Konstruktor **Init** inicializuje obecný grafický terminál s rozměry displeje 240 × 128 pixelů a inicializuje proměnné objektu implicitními hodnotami. Uživatelské procedury jsou nastaveny na nil, jas displeje na 2, prodleva funkce screen saver (**StDelay**) na 0 (funkce vypnuta), příznak zhasnutí **FlDark** na false a periodická obnova stavu optovýstupů je povolena (**FlRefreshOut**:=true). Proměnná **vAddr** je naplněna bázovou adresou terminálu v IO prostoru předanou v parametru **Naddr**.

5.2.2. Done

```
destructor Done;virtual;
```

Destruktor **Done** nastaví optovýstupy do neaktivního stavu a zruší grafický terminál.

5.2.3. Tick

```
procedure Tick;virtual;
```

Metoda **Tick** je rozšířením **tTermChr.Tick**, která volá metody **Tick** objektů klávesnice a displeje o následující činnosti:

Obnoví stav signalizačních LED voláním **LedSign(0,0)** a je-li povolena obnova stavu optovýstupů (**FlRereshOut**), obnoví stav optovýstupů voláním **WriteOut(0,0).**

Testuje příznaky stisku tlačítek "START" a "STOP" v instanci objektu klávesnice (**Keyb^.FlStart** a **Keyb^.FlStop**). Bylo-li některé z tlačítek stisknuto a je instalována příslušná uživatelská procedura (**UserStartProc, UserStopProc**), volá ji a nuluje příznak stisku příslušného tlačítka. Není-li uživatelská procedura instalována, příznak není nulován.

Podle stavu funkce sreen saver nastaví voláním metody **FlLight** jas výbojky podsvětlující displej.

5.2.4. DTick_GetDisplayData

```
function  DTick_GetDisplayData:Boolean;virtual;
```

Metoda **DTick_GetDisplayData** slouží k předání dat pro zobrazení instanci objektu displeje. Rozšiřuje metodu **tTermChr.DTick_GetDisplayData** o předání grafických dat obrazovky prostřednictvím instance objektu **tGraphicDataMan**. Metoda vrací true, byla-li data k dispozici a byla předána.

5.2.5. FLLight

```
procedure FLLight(B: Byte);virtual;
```

Metoda **FlLight** ovládá jas výbojky podsvětlující displej terminálu. V parametru **B** je předávána velikost jasu od 0 (tma) do 4 (nejvyšší jas). Ostatní hodnoty jsou ignorovány.

5.2.6. LedSign

```
procedure LedSign(On,Off: Byte);virtual;
```

Metodou **LedSign** jsou ovládány signalizační LED na terminálu. Nastavením jednotlivých bitů parametru **On** je definováno, které LED mají být rozsvíceny a nastavením jednotlivých bitů parametru **Off** , které LED mají být zhasnuty. Bit s nejvyšší vahou (MSB) přísluší LED nejvíce vpravo (v tlačítku "START"), bit s nejnižší vahou (LSB) přísluší LED nejvíce vlevo. Stav LED je uchováván v objektu displeje. Volání metody modifikuje proměnnou uchovávající stav LED a provede výstup na fyzický hardware. Volání **LedSign(0,0)** tedy nezmění stav LED, ale odešle uchovávaný stav na fyzické výstupy (refresh). Uvedenou činnost metoda zajišťuje prostřednictvím instance objektu displeje voláním **Disp^.mLedSign**.

Příklad: Voláním LedSign(3,64) rozsvítíme dvě LED nejvíce vlevo a zhasneme LED v tlačítku "STOP". Ostatní LED zůstanou nezměněny.

5.2.7. DispContr

```
procedure DispContr(B: Byte);virtual;
```

Metoda **DispContr** ovládá kontrast displeje. V parametru **B** je předávána velikost kontrastu od 0 (nejnižší kontrast) do 6 nebo 15 * (nejvyšší kontrast). Ostatní hodnoty jsou ignorovány. Nastavená hodnota kontrastu je uchována v instanci objektu displeje. Činnost metody je zajištěna voláním **Disp^.mDispCont**.

** podle typu displeje v terminálu - 6 pro displej Seiko (odpovídá objektový typ tDispT10), 15 pro displej Toshiba (odpovídá objektový typ tDispT10A).*

5.2.8. SetDelay

```
procedure SetDelay(B:Byte);virtual;
```

Metoda **SetDelay** slouží k nastavení délky prodlevy pro funkci screen saver. Parametr **B** udává délku prodlevy v minutách, hodnota 0 má význam vypnutí funkce screen saver (nezhasíná nikdy). Přípustné hodnoty parametru **B** jsou od 0 do 30, ostatní hodnoty jsou ignorovány. Nastavená délka prodlevy je uchována v proměnné **StDelay**.

5.2.9. WriteOut

```
procedure WriteOut(On,Off: Byte);virtual;
```

Metodou **WriteOut** je možné ovládat optovýstupy. Nastavením jednotlivých bitů parametru **On** je definováno, které z bitů mají být nastaveny a nastavením jednotlivých bitů parametru **Off** , které z bitů mají být nulovány. Metoda modifikuje uchovaný stav optovýstupů v proměnné **StOut** a provede výstup na fyzický hardware. Volání **WriteOut(0,0)** nezmění stav optovýstupů, ale provede odeslání uchovávaného stavu na fyzické výstupy (refresh).

5.2.10. Readln

```
function  Readln: Byte;virtual;
```

Metodou **Readln** lze číst stav optovstupů terminálu.

5.2.11. BellOn

```
procedure BellOn;virtual;
```

Metoda **BellOn** zapíná zvukovou signalizaci terminálu (pípání). Tato činnost je zprostředkována objektem klávesnice voláním **Keyb^.mBellOn**.

5.2.12. BellOff

```
procedure BellOff;virtual;
```

Metoda **BellOff** vypíná zvukovou signalizaci terminálu (pípání). Tato činnost je zprostředkována objektem klávesnice voláním **Keyb^.mBellOff**.

5.2.13. BeepKeyOn

```
procedure BeepKeyOn;virtual;
```

Metoda **BeepKeyOn** zapíná zvukovou signalizaci stisku klávesy terminálu (krátké pípnutí). Tato činnost je zprostředkována objektem klávesnice voláním **Keyb^.mBeepKeyOn**.

5.2.14. BeepKeyOff

```
procedure BeepKeyOff;virtual;
```

Metoda **BeepKeyOff** vypíná zvukovou signalizaci stisku klávesy terminálu (krátké pípnutí). Tato činnost je zprostředkována objektem klávesnice voláním **Keyb^.mBeepKeyOff**.

5.2.15. KeyPressed

```
function  KeyPressed:Boolean;virtual;
```

Metoda **Keypressed** slouží ke zjištění přítomnosti znaku klávesy ve vyrovnávacím bufferu klávesnice. Tuto funkci zajišťuje volání metody objektu klávesnice **Keyb^.KeyPressed** v předcích objektového typu tTermT10. Metoda je však doplněna o implementaci funkce screen saver. Funkce screen saver je tedy závislá na pravidelném volání této metody (standardně je volána z kontextu procesu "Menu").

5.2.16. SetUserStart

```
procedure SetUserStart(P:pointer);virtual;
```

Metoda **SetUserStart** slouží k instalaci uživatelské procedury volané v případě stisku tlačítka "START". V parametru **P** je předávána adresa uživatelské procedury. Uživatelská procedura musí mít hlavičku shodnou s typem **tUserProcT10** a musí používat vzdálený model volání (far).

5.2.17. SetUserStop

```
procedure SetUserStop(P:pointer);virtual;
```

Metoda **SetUserStop** slouží k instalaci uživatelské procedury volané v případě stisku tlačítka "STOP". V parametru **P** je předávána adresa uživatelské procedury. Uživatelská procedura musí mít hlavičku shodnou s typem **tUserProcT10** a musí používat vzdálený model volání (far).

5.2.18. SetupTerminal

```
procedure SetupTerminal;virtual;
```

Metoda **SetupTerminal** je volána z kontextu procesu "Menu", je-li stisknuta příslušná klávesa (implicitně SHIFT-ENTER). Metoda zjistí, zda je instalována uživatelská procedura OnSetupTerminal a pokud je zavolá ji.

Tato uživatelská procedura dovoluje implementovat setup terminálu podle konkrétních potřeb a přání uživatele. Příklad uživatelského setupu je implementován v jednotce **SetupT10**. Příklad dovoluje v setupu nastavovat kontrast displeje, jas

podsvětlující výbojky, prodlevu pro funkci screen saver a zapínat nebo vypínat zvukovou signalizaci stisku tlačítek. Nastavené hodnoty jsou uchovávány v zálohované paměti RWM. Jednotka SetupT10 je určena k modifikaci uživatelem podle jeho potřeb.

5.2.19. SetOnSetupTerminal

```
procedure SetOnSetupTerminal(P : Pointer);virtual;
```

Metoda **SetOnSetupTerminal** slouží k instalaci uživatelské procedury, která implementuje konkrétní setup terminálu. V parametru **P** je předávána adresa uživatelské procedury. Uživatelská procedura musí mít hlavičku shodnou s typem **tOnSetupT10** a musí používat vzdálený model volání (far).

5.2.20. EnableRefreshOut

```
procedure EnableRefreshOut;virtual;
```

Metoda **EnableRefreshOut** povolí odesílání uchovaného stavu optovýstupů na fyzické výstupy v metodě **Tick** nastavením **FlRefreshOut**:=true.

5.2.21. DisableRefreshOut

```
procedure DisableRefreshOut;virtual;
```

Metoda **DisableRefreshOut** zakáže odesílání uchovaného stavu optovýstupů na fyzické výstupy v metodě **Tick** nastavením **FlRefreshOut**:=false. Zákaz obnovy stavu optovýstupů dovoluje při ovládání optovýstupů obejít služby objektu terminálu. Implicitně je obnova stavu výstupů konstruktorem **Init** povolena.