DiskIO

JEDNOTKA PRO PRÁCI S RAM, ROM A FLASH DISKY

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 2.00

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: 08.01.2004
Datum posledního uložení dokumentu: 08.01.2004
(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	Hv		První vydání.
1.10	1.XX	Tu	19.5.2003	Úprava dokumentu dle ISO9000.
2.00	2.XX	Hv	05.01.2004	Zveřejněna proměnná cSectorSize Zpřesnění popisu

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky pro práci s RAM, ROM a FLASH disky.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu není potřeba číst žádný další manuál, ale je potřeba se orientovat v používání programového vybavení SofCon.

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.

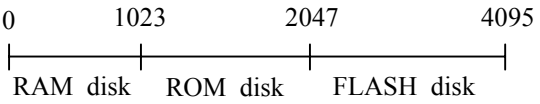
3. Úvod

Jednotka DiskIO poskytuje základní diskové služby pro RAM, ROM a FLASH disky. Názvy těchto disků jsou odvozeny od paměti, ve které jsou vytvořeny a souhrnně se nazývají rozšířenými disky, protože jsou vytvořeny především v paměti nad 1MB viz dále. Data z nich mohou být čtena i zapisována, např. archivy dat z výrobního procesu, definice obrazovek, konstant aj.

Poskytované služby, pracují s blokem dat, který se nazývá se sektor a má velikost 512B. Jednotlivé disky lze charakterizovat následujícím způsobem:

- ***RAM disk*** – je vytvořen v paměti RAM a jeho obsah je zálohován baterií a lze jej měnit bez omezení.
- ***FLASH disk*** – je vytvořen v paměti FLASH a jeho obsah lze měnit, ale počet zápisů do jednoho sektoru je zpravidla dán technologickými možnostmi paměti FLASH, max. 10000.
- ***ROM disk*** – je vytvořen v paměti ROM a je pouze pro čtení. Jeho obsah je určen pevně při programování paměti v programátoru.

Rozhraní rozšířených disků poskytuje pevně daný počet sektorů podle osazených pamětí a konfigurační tabulky v BIOS. Podle použitého čísla sektoru se přistupuje k patřičnému disk, viz obrázek.



Příklad sektorů poskytnutých rozhraním rozšířených disků

K základním službám rozhraní, které musí aplikace vždy volat, patří procedury InitDiskArea a GetDiskParam.

- Procedura InitDiskArea provede inicializaci řídicích struktur určujících počty sektorů pro jednotlivé disky.
- Procedura GetDiskParam vrátí strukturu, ve které jsou uloženy hodnoty prvního sektoru a počet sektorů pro jednotlivé typy disků. Podle těchto mezí se pak určuje přístup k jednotlivým diskům (RAM, ROM a FLASH) u služeb WriteSector, ReadSector a CopySector provádějících čtení, zápis a kopírování sektorů.

Vzhledem k omezené adresaci reálného módu na 1MB a rozdělením paměťového prostoru ve stavebnici KIT ve standardní konfiguraci na 512kB pro paměť RAM a 512kB pro paměť ROM(FLASH) je v diskovém prostoru této jednotky také adresována paměť RAM nebo ROM(FLASH), která přesahuje 512kB a paměti v patici HIGH při 8b přístupu. (Pozn.: Pojem 8b přístup je podrobně popsán v manuálu KIT386EXR) Základní rozdělení paměťového prostoru mezi paměť RAM a ROM(FLASH) lze změnit pomocí konfigurační tabulky. (Pozn.: Práce s konfigurační tabulkou je popsána v manuálu RTD a popis tabulky je v manuálu BIOS 386EXR)

Pozn.: V aplikaci, která používá rozšířené disky by se neměl používat chráněný režim a pracovat s A20 jinak může dojít k poškození dat, případně k chybovým hlášením při vykonávání výše uvedených služeb.

4. Popis konstant a proměnných

```
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.

```
cSectorSize = 512
 proměnná určuje velikost sektoru, která je dána typem paměti a
                            přístupu (8/16b). Tzn. tato jednotka
                            podporuje paměti FLASH - AT29C010,
                            AT29C020 a AT29C040A.

IOResultDiskIO : Byte = 0
 proměnná obsahuje stav poslední operace s disky.

Konstanty popisující stav poslední operace s rozšířenými disky:
      DiskIO_OK = 0;         operace proběhla bez chyby
      DiskIO_FatalErr = 255; operace proběhla s chybou, kterou nelze
                            blíže určit

      DiskIO_MemoryErr   = 1;
                              struktury pro práci s rozšířenými disky
                              nemohly být vytvořeny pro nedostatek
                              paměti.
      DiskIO_A20Err      = 2;
                              při provádění operace s rozšířenými
                              disky bylo zjištěno nastavení A20.
      DiskIO_CPUErr      = 3;
                              jednotka je určena pouze pro procesor
                              typu 386 a vyšší.
      DiskIO_SectorErr   = 4;
                              neplatné číslo sektoru. Číslo sektoru je
                              mimo rozsah rozšířených disků.
      DiskIO_WriteErr    = 5;
                              sektor nebyl zapsán. (možné poškození
                              paměti)
      DiskIO_ReadErr     = 6;
                              sektor nebyl přečten. (možné poškození
                              paměti)
      DiskIO_CopyErr     = 7;
                              sektor nebyl přečten. (možné poškození
                              paměti)
      DiskIO_FlashManErr = 8;
                              neplatná konfigurační tabulka.
      DiskIO_FlashTypeErr = 9;
                              neplatná konfigurační tabulka.
      DiskIO_GetParamErr = 10;
                              poškozená nebo neinicializovaná
                              struktura pro práci s rozšířenou pamětí.
      DiskIO_DskStrucErr = 11;
                              poškozená nebo neinicializovaná
                              struktura pro práci s rozšířenými disky.
```

5. Popis typů

Typ **tSectParam** slouží k předávání informace o prvním sektoru a počtu sektorů, který přísluší k danému typu disku.

```
tSectParam = record
  FirstSector: LongInt;
  SectorCount: LongInt;
end;
```

Typ **tDiskRecord**, **pDiskRecord** slouží k předávání informace o všech typech disků. Jednotlivé položky pak obsahují informace o prvním sektoru a počtu sektorů pro daný disk. Pomocí těchto mezí se při práci s disky (zadává se sektor) určuje typ disku tj. paměti.

```
pDiskRecord = ^tDiskRecord;
tDiskRecord = record
    RAM    : tSectParam;
    ROM    : tSectParam;
    FLASH  : tSectParam;
end;
```

6. Funkce

6.1. InitDiskArea

Popis:
Procedura provede inicializaci struktur pro práci s rozšířenými disky. Při jejich inicializaci je použita konfigurační tabulka v BIOSu, která lze v případě uložení BIOS v paměti FLASH modifikovat pomocí RTD.
Pokud je konfigurační tabulka neplatná pro osazené typy pamětí (použit Default mode při zapnutí KIT386EXR), může při pokusu o zápis nebo čtení z rozšířeného disku dojít k restartu celého systému. Proto je důležité vždy, před vlastním spuštěním aplikace pracující s rozšířenými disky, nastavit správně všechny propojky.
Syntaxe:
procedure InitDiskArea;
Vstup:
žádný
Výstup:
V proměnné IOResultDiskIO je uložen stav operace.

6.2. DestroyDiskArea

Popis:
Procedura provede uvolnění všech struktur, které jsou používány při práci s rozšířenými disky.
Syntaxe:
procedure DestroyDiskArea;
Vstup:
žádný
Výstup:
Nastaví IOResultDiskIO na DiskIO_OK.

6.3. GetDiskParam

Popis:
Procedura nastaví do zadané struktury aDiskParam popis rozšířených disků, viz 5, Popis typů. Tento popis poté slouží k určení prvního a posledního sektoru vybraného disku, viz výše.
Syntaxe:
procedure GetDiskParam(aDiskParam:pDiskRecord);
Vstup:
žádný
Výstup:
aDiskParam je ukazatel na strukturu tDiskRecord obsahující čísla sektorů určujících rozdělení disků.
V proměnné IOResultDiskIO je uložen stav operace.

6.4. WriteSector

Popis:
Procedura provede zápis bloku dat z adresy aAddr o délce 512B a na sektor aSectNumber. Typ disku je určen číslem sektoru.
Pozn.: Rozdělení čísel sektoru mezi jednotlivé disky se provede v proceduře InitDiskArea.
Syntaxe:
procedure WriteSector(aAddr:Pointer;aSectNumber:LongInt);
Vstup:
aAddr adresa bloku dat o délce 512B, který je zapsán do paměti spravované touto jednotkou (zpravidla nad 1MB).
aSectNumber číslo sektoru, na který je zapsán blok dat.
Výstup:
V proměnné IOResultDiskIO je uložen stav operace.

6.5. ReadSector

Popis:
Procedura přečte blok dat o velikosti 512B ze sektoru aSectNumber a data zapíše na adresu aAddr. Typ disku je určen číslem sektoru.
Pozn.: Rozdělení čísel sektoru mezi jednotlivé disky se provede v proceduře InitDiskArea.
Syntaxe:
procedure ReadSector(aSectNumber:LongInt; aAddr:Pointer);
Vstup:
aSectNumber číslo sektoru, z kterého je přečten blok dat.
aAddr adresa, na kterou jsou zapsána přečtená data o velikosti 512B.
Výstup:
V proměnné IOResultDiskIO je uložen stav operace.

6.6. CopySector

Popis:
Procedura přečte blok dat ze sektoru aSectSource a zapíše ho na sektor aSectDest. Procedura přenáší data pomocí pomocného bufferu, což jí umožní provádět operace čtení a zápisu sektorů na stejném disku.
Pozn.: Rozdělení čísel sektoru mezi jednotlivé disky se provede v proceduře InitDiskArea.
Syntaxe:
procedure CopySect(aSectSource:LongInt; aSectDest:LongInt);
Vstup:

aSectSource	číslo sektoru, z kterého je přečten blok dat.
aSectDest	číslo sektoru, do kterého se provede zápis přečteného bloku dat.

Výstup:
V proměnné IOResultDiskIO je uložen stav operace.

7. Příklad

Příklad ukazuje použití jednotky pro práci s rozšířenými disky

```
uses
  DiskIO,
...;

var
 aPole:array [0..cSectorSize-1] of Byte; { datova oblast }
 aDiskArea:tDiskRecord;

...
 { definice vlastnich typu, konstant, promennych, funkci a procedur }

begin
 InitDiskArea; { inicializace struktur jednotky }
 if IOResultDiskIO<>DiskIO_OK then
 begin
  ... { obsluha chyby }
 end;
 GetDiskParam(@DiskArea);
 if IOResultDiskIO<>DiskIO_OK then
 begin
  ... { obsluha chyby}
 end;
 { vypis cisel prvniho sektoru a poctu sektoru jednotlivych typu
   rozsirenych disku }
 if IOResultDiskIO=DiskIO_OK then
 begin
   with aDiskArea.RAM do
   begin
     Writeln('RAM:  ');
     Writeln('........ First sector : ',FirstSector);
     Writeln('........ Sector count : ',SectorCount);
   end;
   with aDiskArea.ROM do
   begin
     Writeln('ROM:  ');
     Writeln('........ First sector : ',FirstSector);
     Writeln('........ Sector count : ',SectorCount);
   end;
```

```
   with aDiskArea.FLASH do
   begin
     Writeln('FLASH:  ');
     Writeln('........ First sector : ',FirstSector);
     Writeln('........ Sector count : ',SectorCount);
   end;
 end;

 { priklad zapisu bloku dat na sektoru cislo 100 }
 {
   Pri rozdelení cisel sektoru mezi jednotlive typy rozsirenych disku
   nasledujicim zpusobem:
            RAM : 0 .. 1023   ( FirstSector = 0; SectorCount = 1024)
            ROM : 0 .. 0      ( FirstSector = 0; SectorCount = 0)
            FLASH : 1024 .. 2047
                              ( FirstSector = 0; SectorCount = 1024 }
   Pak bude zapis proveden na RAM disk, v pripade zapisu na sektor
   cislo 1050 by doslo k zapisu na FLASH disk.
 }
 WriteSector(@aPole,100);
 Writeln('Stav zapisu : ',IOResultDiskIO);
 if IOResultDiskIO<>DiskIO_OK then
 begin
  ... { obsluha chyby }
 end;
{ priklad cteni sektoru cislo 100 }
 ReadSector(100, @aPole);
 Writeln('Stav cteni : ',IOResultDiskIO);
 if IOResultDiskIO<>DiskIO_OK then
 begin
  ... { obsluha chyby }
 end;
{ priklad kopirovani mezi sektory 0 a 300 na RAM disku}
 CopySector(0,300);
 Writeln('Stav kopirovani : ',IOResultDiskIO);
 if IOResultDiskIO<>DiskIO_OK then
 begin
  ... { obsluha chyby }
 end;
 ...
end;
```