HwSyst

ZÁKLADNÍ ADRESY A KONSTANTY PRO ŘÍDICÍ JEDNOTKY KIT

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 3.30

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: 23.09.2005
Datum posledního uložení dokumentu: 23.09.2005
(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	Hv		První vydání.
1.10	3.XX	Tu	20.05.2003	Úprava dokumentu dle ISO9000.
1.11	3.XX	Tu	03.06.2003	Změna velikosti konstanty IRQpSec.
3.00	4.XX	Hv	08.01.2004	Upřesnění popisu Přidány funkce pro práci s CS – GetCSAddr, GetCSSize, GetCSType dle typu procesoru (skryt přímý přístup k patřičným proměnným) Doplnění podpory pro procesor V40 ve standardním režimu Přidány převodní tabulka MemSizeKB Přidána konstanta hwsysReserved Přejmenována funkce GetDelay na GetAccessTime Opravena chyba ve funkci GetBaseAddr ve verzi pro DOS
3.10	5.XX	Wil	18.06.2004	Přidána funkce MemCMP. **Pozor! Zrušení konstanty IRQTime a nahrazení za konstanty DefIRQTime a ActIRQTime.** Toto může při překladu stávajících aplikací s touto verzí knihovny vyvolat chybu při překladu. Více viz popis konstanty ActIRQTime v tomto dokumentu. **Pozor! Zrušení konstanty IRQpSec a nahrazení za konstanty DefIRQpSec a ActIRQpSec.** **Pozor! Zrušení konstanty IRQpDay a nahrazení za konstanty DefIRQpDay a ActIRQpDay.** Přidány funkce SetTimer, GetTimer, GetRealIRQTimer a ReadTimer0. Úpravy konstant systémového časovače pro přesnější periodu IRQ0 55ms.
3.20	5.XX	Wil	13.08.2004	Úpravy pro nový procesor Kit188ER. **Pozor! Funkce GetOpMemory, GetChipSize a GetCSsize dříve vracely kód velikosti paměti, který se poté musel pomocí příslušné tabulky převést na skutečnou velikost. V nové verzi jednotky tyto funkce již vrací velikosti pamětí přímo v KB.** To může vést k jisté nekompatibilitě při překladu již napsaných starších aplikací s novými knihovnami (nutno upravit aplikaci v místě, kde volá jednu z výše uvedených funkcí). Díky výše zmíněným změnám byly odstraněny konstanty **hwsysMemSizeXxxKB** a pole **MemSizeKB**. Přidány funkce **PtrToAbsAddr** a **AbsAddrToPtr**. Přidány funkce pro obsluhu řadiče přerušení **NonSpecEndOfInt, EnableIRQ, DisableIRQ,**

				SetIRQroutine, GetIRQroutine
3.30	5.XX	Wil	23.09.2005	Přidána podpora IRQ>7 pro PC.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky definující základní adresy a konstanty pro řídící jednotky KIT.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

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

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

Při používání funkcí z kapitol „8 Funkce pro zjištění paměťového prostoru“ a „9 Funkce pro zjištění parametrů pamětí“ je vhodné se seznámit s paměťovými modely jednotlivých procesorů, které jsou popsány v dokumentu „KitAdr“.

2. Termíny a definice

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

3. Úvod

Tato programová jednotka slouží pro definici konstant, funkcí a proměnných obsahujících adresy periferií poplatných řídicí desce s různými procesory např. V40, 80386EXR, Am188ER. Tyto konstanty včetně proměnné SystIdent určující řídicí desku jsou nastavovány v inicializační sekci této jednotky.

Tento manuál neobsahuje podrobný popis konstant a proměnných, které jednotka obsahuje, poněvadž jejich popis již příliš zasahuje do prvotních inicializací desek na úrovni Biosu. Bližší informace lze proto nalézt v manuálech popisujících hardware řídicích desek.

4. Konstanty, proměnné a typy

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

Tyto konstanty udávají číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

4.1. Typ řídicí desky

```
SystIdent : Word;
```

Proměnná je nastavena na dále popsané konstanty v inicializační sekci a jsou v ní zakódovány vlastnosti řídicí desky.

Popis kódů řídicí desky

```
SystKitV40_8     = $0101;
```

Řídicí deska KitV40 s 8 MHz procesorem.

```
SystKitV40_16    = $0102;
```

Řídicí deska KitV40 s 16 MHz procesorem.

```
SystJT283_8      = $0201;
```

Zákaznická deska analyzátoru plynů JT283.

```
SystTERM03_8     = $0301;
```

Řídicí deska terminálu Term03 s 8 MHz procesorem V40.

```
SystTERM03_16    = $0302;
```

Řídicí deska terminálu Term03 s 16 MHz procesorem V40.

```
SystTERM10_8     = $0401;
```

Terminál Term10 s 8 MHz řídicí deskou KitV40.

```
SystTERM10_16    = $0402;
```

Terminál Term10 s 16 MHz řídicí deskou KitV40.

```
SystKit386EXR_25 = $0421;
```

Řídicí deska Kit386EXR s 25 MHz procesorem.

```
SystKit386EXR_33 = $0422;
```

Řídicí deska Kit386EXR s 33 MHz procesorem.

```
SystKit188ER_40  = $0611;
```

Řídicí deska Kit188ER s 40 MHz procesorem.

```
SystKit188ER_44  = $0612;
```

Řídicí deska Kit188ER s 44,2368 MHz procesorem.

```
SystKit188ER_48  = $0613;
```
Řídicí deska Kit188ER s 48 MHz procesorem.

```
SystKit188ER_49  = $0614;
```
Řídicí deska Kit188ER s 49,152 MHz procesorem.

```
SystDOS          = $FF00;
```
Program běží pod DOS na PC (Real Mode nebo DOS Protected Mode).

Pozn: Terminál Term03 je vždy osazen procesorem V40.

Terminál Term10 může být připojen k libovolné řídicí desce, která umožňuje přístup ke sběrnici IOBUS, tj. KitV40, Kit386EXR nebo Kit188ER. Jelikož BIOS pro řídicí desku Kit386EXR bez nebo s Term10 je stejný, platí hodnota `SystKit386EXR_xx` jak pro samotný Kit386EXR tak i pro připojený k Term10. Stejně tak platí i v případě řídicí desky Kit188ER.

4.2. Typ paměti

```
hwsysReserved = 0;
```
paměť není osazena nebo je vyhrazena, rezervována, pro připojené zařízení

```
hwsysRAM      = 1;
```
aplikační paměť RAM

```
hwsysFLASH    = 2;
```
aplikační paměť FLASH

```
hwsysROM      = 3;
```
aplikační paměť ROM

4.3. Adresy a inicializace periferií

Pozor! Aplikace nesmí měnit hodnotu žádné z níže uvedených konstant. Tyto konstanty jsou nadeklarovány jako typové konstanty (tj. proměnné) pouze pro jejich správné nastavení v inicializační sekci jednotky podle řídicí desky. Tj. pro aplikaci jsou pouze „ReadOnly“.

Pozn: Některé konstanty či proměnné jsou společné pro všechny řídicí desky, některé jsou poplatné pouze pro konkrétní řídicí desku.

```
ATCU      : Word = $40; { Pro standardní PC a Kit386EXR }
                = $3C; { Pro řídicí desku KitV40 }
```
Proměnná obsahuje bázovou IO adresu systémového časovače a je nastavována v inicializační sekci jednotky podle řídicí desky a prostředí (DOS/KIT).

```
A188PCB   = $FF00;
```
Bázová IO adresa všech interních periferií (Peripheral Control Block) procesoru Kit188ER

```
ATCU188   = A188PCB+$50;
```
Bázová IO adresa systémového časovače Kit188ER.

```
ARTC     = $8210;
```
Proměnná obsahuje bázovou IO adresu obvodu reálného času, který je obsluhován v BIOS a dále přístupný pomocí rozhraní v jednotce RTCTime.

```
cBIOSBeginAddr : LongInt = $FE000; { na KitV40 }
                         = $FD000; { na Kit386EXR a Kit188ER }
```

Proměnná obsahuje lineární adresu začátku BIOS v paměťovém prostoru řídicí desky a je nastavována v inicializační sekci jednotky podle řídicí desky.

4.3.1. Funkce WatchDog na KitV40 a Kit386EXR

```
ALedOn   = $4210;
```

Proměnná obsahuje IO adresu, při jejímž čtení se rozsvítí LED dioda na řídicí desce. Současně dojde k obnovení monostabilního obvodu WatchDog.

```
ALedOff  = $6210;
```

Proměnná obsahuje IO adresu, při jejímž čtení zhasne LED dioda na řídicí desce. Současně dojde k obnovení monostabilního obvodu WatchDog.

```
AWDogL   = ALedOn;
```

V současné době je stejná s adresou ALedOn a čtení této adresy způsobuje obnovení monostabilního obvodu WatchDog.

```
AWDogH   = ALedOff;
```

V současné době je stejná s adresou ALedOff a čtení této adresy způsobuje obnovení monostabilního obvodu WatchDog.

Pozn: Správná obsluha WatchDog se provádí periodickým čtením střídavě z adresy AWDogL a AWDogH. Tuto obsluhu zajišťuje BIOS automaticky v přerušení od systémového časovače, takže aplikace se o tuto obsluhu starat nemusí.

4.3.2. Funkce WatchDog na Kit188ER

```
AWDog    = $FF7A;
```

Adresa funkce Watch-Dog v IO prostoru.

```
MWDog    = $2000;
```

Maska pro nastavení/nulování 13.bitu Watch-Dog na adrese AWDog.

Pozn: Správná obsluha WatchDog se provádí periodickým nastavováním a nulováním 13.bitu na adrese AWDog. Tuto obsluhu zajišťuje BIOS automaticky v přerušení od systémového časovače, takže aplikace se o tuto obsluhu starat nemusí.

4.3.3. Řadič přerušení na PC, KitV40 a Kit386EXR

4.3.3.1. Indexy hardwarového přerušení (IRQ)

```
IRQ0    = 0;
```

Konstanta **IRQ0** definuje přerušení 0 (INT 08h), které je využíváno systémovým časovačem Timer0.

```
IRQ1    = 1;
```

Konstanta **IRQ1** definuje přerušení 1 (INT 09h).

```
IRQ2    = 2;
```

Konstanta **IRQ2** definuje přerušení 2 (INT 71h).

```
IRQ3    = 3;
```

Konstanta **IRQ3** definuje přerušení 3 (INT 0Bh).

```
IRQ4    = 4;
```

Konstanta **IRQ4** definuje přerušení 4 (INT 0Ch).

```
IRQ5    = 5;
```
Konstanta **IRQ5** definuje přerušení 5 (INT 0Dh).

```
IRQ6    = 6;
```
Konstanta **IRQ6** definuje přerušení 6 (INT 0Eh).

```
IRQ7    = 7;
```
Konstanta **IRQ7** definuje přerušení 7 (INT 0Fh).

```
IRQ8    = $8;
```
Konstanta **IRQ8** definuje přerušení 8 (INT 70h). Pouze pro PC.

```
IRQ9    = $9;
```
Konstanta **IRQ9** definuje přerušení 9 (INT 71h). Pouze pro PC.

```
IRQAh   = $A;
```
Konstanta **IRQAh** definuje přerušení 10 (INT 72h). Pouze pro PC.

```
IRQBh   = $B;
```
Konstanta **IRQBh** definuje přerušení 11 (INT 73h). Pouze pro PC.

```
IRQCh   = $C;
```
Konstanta **IRQCh** definuje přerušení 12 (INT 74h). Pouze pro PC.

```
IRQDh   = $D;
```
Konstanta **IRQDh** definuje přerušení 13 (INT 75h). Pouze pro PC.

```
IRQEh   = $E;
```
Konstanta **IRQEh** definuje přerušení 14 (INT 76h). Pouze pro PC.

```
IRQFh   = $F;
```
Konstanta **IRQFh** definuje přerušení 15 (INT 77h). Pouze pro PC.

```
type
  tIrq      = Irq0..Irq7;  {pro KitV40 a Kit386EXR}
            = Irq0..IrqFh; {pro PC}
```
Intervalový typ možných IRQ.

4.3.3.2. Adresy řadiče přerušení

```
AICUM    = $0020;
```
Proměnná obsahuje IO adresu Master řadiče přerušení typu 8259.

```
AICUS    = $00A0;
```
Proměnná obsahuje IO adresu Slave řadiče přerušení typu 8259.

```
M59_IMR  = $21;
```
Konstanta definuje IO adresu masky Master řadiče přerušení typu 8259.

```
S59_IMR  = $A1;
```
Konstanta definuje IO adresa masky Slave řadiče přerušení typu 8259.

Pozn: Pro povolování či zakazování jednotlivých přerušení IRQ je výhodné použít procedur uvedených v kapitole „5 Funkce pro obsluhu řadiče přerušení“.

4.3.4. Řadič přerušení na Kit188ER

4.3.4.1. Indexy hardwarového přerušení (IRQ)

```
IRQ8        = $08;
```
Konstanta **IRQ8** definuje přerušení 8 (INT 08h), které je využíváno systémovým časovačem Timer0.

```
IRQ9        = $09;
```
Konstanta **IRQ9** definuje přerušení 9 (INT 09h), které je rezervováno pro budoucí použití.

```
IRQ0Ah      = $0A;
```
Konstanta **IRQ0Ah** definuje přerušení A (INT 0Ah), které je využíváno řadičem DMA0.

```
IRQ0Bh      = $0B;
```
Konstanta **IRQ0Bh** definuje přerušení B (INT 0Bh), které je využíváno řadičem DMA1.

```
IRQ0Ch      = $0C;
```
Konstanta **IRQ0Ch** definuje přerušení C (INT 0Ch), na které je přiveden signál INT3 ze sběrnice IOBUS.

```
IRQ0Dh      = $0D;
```
Konstanta **IRQ0Dh** definuje přerušení D (INT 0Dh), na které je přiveden signál INT4 ze sběrnice IOBUS.

```
IRQ0Eh      = $0E;
```
Konstanta **IRQ0Eh** definuje přerušení E (INT 0Eh), které je využíváno UART 8250 A (COM A).

```
IRQ0Fh      = $0F;
```
Konstanta **IRQ0Fh** definuje přerušení F (INT 0Fh), které je využíváno UART 8250 B (COM B).

```
IRQ10h      = $10;
```
Konstanta **IRQ10h** definuje přerušení 10h (INT 10h), které bývá překryto SW přerušením od Video služeb.

```
IRQ11h      = $11;
```
Konstanta **IRQ11h** definuje přerušení 11h (INT 11h), které je rezervované pro interní WatchDog.

```
IRQ12h      = $12;
```
Konstanta **IRQ12h** definuje přerušení 12h (INT 12h), které používá časovač Timer1.

```
IRQ13h      = $13;
```
Konstanta **IRQ13h** definuje přerušení 13h (INT 13h), které používá časovač Timer2 (je použit jako předdělička pro Timer0).

```
IRQ14h      = $14;
```
Konstanta **IRQ14h** definuje přerušení 14h (INT 14h), které je využíváno interním systémovým UART.

```
type
  tIrq      = Irq8  ..Irq14h;
```
Intervalový typ možných IRQ.

4.3.4.2. Adresa řadiče přerušení

```
AICU188     = A188PCB+$22;
```
Proměnná obsahuje IO adresu interního řadiče přerušení procesoru Kit188ER.

Pozn: Pro povolování či zakazování jednotlivých přerušení IRQ je výhodné použít procedur uvedených v kapitole „5 Funkce pro obsluhu řadiče přerušení“.

4.3.5. Obsluha systémového časovače

```
ITCU     : Longint = 65536; pro DOS na PC
                   = 54926; pro KITV40/8,16
                   = 65475; pro KIT386EXR/25
                   = 64821; pro KIT386EXR/33
```

Proměnná obsahuje přednastavenou hodnotu děliče systémového časovače 0 a je nastavována v inicializační sekci jednotky podle řídicí desky a prostředí (DOS/KIT). Této hodnotě odpovídá perioda časovače přibližně 55ms.

Pozn: Pro Kit188ER se příslušná hodnota načte přímo z BIOS, stejně tak i konstanty ITimerClk a DefIRQpDay. Z toho plyne výhoda, že vygenerovaná aplikace není závislá na konkrétním BIOS a tedy ani na konkrétní frekvenci procesoru Kit188ER.

```
ITimerClk: Longint = 1193180; pro DOS na PC
                   = 1000000; pro KITV40/8,16
                   = 1190476; pro KIT386EXR/25
                   = 1178571; pro KIT386EXR/33
```

Proměnná obsahuje frekvenci [Hz] přivedenou do systémového časovače 0 a je nastavována v inicializační sekci jednotky podle řídicí desky a prostředí (DOS/KIT).

```
DefIRQpSec  : real    = 18.206481933; { pulsu/sec } pro DOS na PC
                      = 18.206313950; { pulsu/sec } pro KITV40/8,16
                      = 18.181871075; { pulsu/sec } pro KIT386EXR/25
                      = 18.181938393; { pulsu/sec } pro KIT386EXR/33
```

Proměnná obsahuje implicitní hodnotu frekvence přerušení systémového časovače 0 v počtu pulsů za sekundu a je nastavována v inicializační sekci jednotky podle řídicí desky a prostředí (DOS/KIT).

Pozn: Hodnota této proměnné by měla odpovídat zhruba výpočtu `ITimerClk/ITCU`.

```
DefIRQpDay  : Longint = 1573040; { pulsu / den }
```

Proměnná obsahuje implicitní hodnotu počtu přerušení systémového časovače 0 za den a je nastavována v inicializační sekci jednotky.

Z důvodů kompatibility s BIOS a staršími aplikacemi jsou v případě nastavena nepřesná hodnota (viz poznámka).

Pozn.: Zachování kompatibility je nutné z důvodu, jelikož konstanta stejného významu se používá i v BIOS v přerušení od systémového časovače, ve kterém se inkrementuje DWORD na adrese $0040:$006C (Tuto hodnotu vrací také funkce BIOSu AH=00h Int1Ah). Pokud tento čítač přeteče výše zmíněnou konstantu počtu přerušení za den, vynuluje se a začne čítat od nuly.

```
DefIRQTime  = 55; {[ms]}
```

Konstanta obsahuje implicitní přibližnou periodu přerušení systémového časovače 0 v milisekundách.

Jelikož stav periferií se může měnit z důvodu inicializací patřičných ovladačů (např. ovladač systémového časovače v knihovně Tick), měla by aplikace přednostně používat (jen pro čtení) následující proměnné uvedené v této kapitole. Dalším důležitým upozorněním je, že **v případě použití ovladače systémového časovače v knihovně Tick se následující tři proměnné vztahují pro původní (tj. „NEzrychlené“) přerušení systémového časovače.** V rámci onoho „zrychlování“ může totiž docházet k nepřesnostem při volání původního „nezrychleného“ systémového časovače oproti standardním 55ms a proto proměnné v této kapitole

slouží jako korekční činitelé (např. pro aplikaci a knihovny pro odměřování časových intervalů, které využívají právě standardního „nezrychleného“ systémového časovače).

`ActIRQTime : word = DefIRQTime; {[ms]}`

Proměnná obsahuje aktuální přibližnou periodu přerušení systémového časovače 0 v milisekundách. Tato proměnná se při zrychlování systémového časovače 0 mění.

`ActIRQpSec  : real    = DefIRQpSec; { pulsu / sec }`

Proměnná obsahuje skutečnou hodnotu frekvence přerušení systémového časovače 0 v počtu pulsů za sekundu a je nastavována v inicializační sekci jednotky. Při případné změně frekvence přerušení systémového časovače se tato proměnná musí nastavit na správnou skutečnou hodnotu. To zajišťuje např. knihovna Tick, která se systémovým časovačem pracuje. Aplikace by nikdy neměla měnit hodnotu této proměnné, ale pouze ji číst.

`ActIRQpDay  : Longint = DefIRQpDay; { pulsu / den }`

Proměnná obsahuje skutečnou hodnotu počtu přerušení systémového časovače 0 za den a je nastavována v inicializační sekci jednotky. Při případné změně frekvence přerušení systémového časovače se tato proměnná musí nastavit na správnou skutečnou hodnotu. To zajišťuje např. knihovna Tick, která se systémovým časovačem pracuje. Aplikace by nikdy neměla měnit hodnotu této proměnné, ale pouze ji číst.

Zde je nutno také mít na paměti, že BIOS používá pro časovač Int1Ah stále hodnotu odpovídající konstantě DefIRQpDay. Proto aplikace musí zvážit, zda při potřebě znát počet přerušení za den má použít proměnnou ActIRQpDay nebo DefIRQpDay. Pokud toto zjištění provádí ve spojitosti s časovačem Int1Ah, musí vždy použít DefIRQpDay. Pokud ale aplikaci zajímá opravdu skutečný počet přerušení časovače za den, použije proměnnou ActIRQpDay.

Pozor! Byla zrušena proměnná IRQTime. Pokud vám při překladu stávajících aplikací s touto verzí knihovny překladač hlásí chybu „Unknown Identifier“, nahraďte proměnnou IRQTime za ActIRQTime. V žádném případě byste však neměli hodnotu proměnné ActIRQTime měnit, tj. pokud jste v aplikaci volali např. „IRQTime := 60“, můžete toto volání zrušit – nenahrazovat za „ActIRQTime := 60“. Pozn: Ruční nastavení IRQTime bylo dříve nutné pro korekci měření časů, v současné době se tato korekce provádí automaticky.

Podobně platí pro proměnné IRQpSec vs. DefIRQpSec, ActIRQpSec a IRQpDay vs. DefIRQpDay a ActIRQpDay.

5. Funkce pro obsluhu řadiče přerušení

5.1. Procedura NonSpecEndOfInt

`procedure NonSpecEndOfInt;`

Tato InLine procedura provede ukončení nespecifického hardwarového přerušení. (Na PC,KitV40,Kit386 se jedná o OUT 20h,20h)

5.2. Procedura EnableIRQ

```
procedure EnableIRQ (IRQNo:tIrq);
```

Procedura povolí v řadiči přerušení IRQ s číslem IRQNo.

5.3. Procedura DisableIRQ

```
procedure DisableIRQ(IRQNo:tIrq);
```

Procedura zakáže v řadiči přerušení IRQ s číslem IRQNo.

5.4. Procedura SetIRQroutine

```
procedure SetIRQroutine(IRQNo:tIrq; P:pointer);
```

Procedura nastaví vlastní obsluhu přerušení IRQ s číslem IRQNo.

5.5. Procedura GetIRQroutine

```
function  GetIRQroutine(IRQNo:tIrq):pointer;
```

Funkce vrátí ukazatel na obsluhu přerušení IRQ s číslem IRQNo.

5.6. Procedura PushfCli

```
procedure PushfCli;
```

Inline procedura, která uloží příznaky procesoru (FLAGS) na zásobník a zakáže přerušení. Příklad je uveden u procedury Popf.

5.7. Procedura Popf

```
procedure Popf;
```

Inline procedura, která obnoví příznaky procesoru (FLAGS) ze zásobníku. Používá se ve spojitosti s procedurou PushfCli pro zakázání a povolení přerušení. To znamená, pokud programátor vyžaduje provést určitou akci (algoritmus) tak, aby jej nepřerušilo žádné přerušení, zavolá před tímto algoritmem proceduru PushfCli a po provedení algoritmu Popf.

```
Př. PushfCli; {zakáže přerušení}
    ... {akce během níž se vyžaduje zakázané přerušení}
    Popf; {povolí přerušení}
```

Příkazy PushfCli a Popf lze do sebe i vzájemně vnořovat:

```
Př. Procedure MyAlg;
    Begin
      PushfCli;
      ... {akce během níž se vyžaduje zakázané přerušení}
      Popf;
    End;

    Procedure MyAlg2;
    Begin
      PushfCli;
      ... {začátek akce během níž se vyžaduje zakázané
           přerušení}
      MyAlg; {zavolání procedury}
      ... {dokončení akce během níž se stále vyžaduje zakázané
```

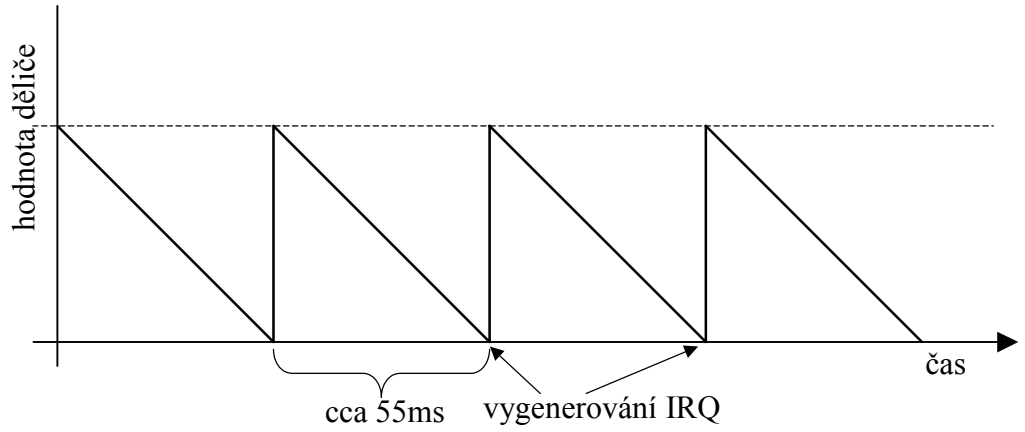
```
        přerušení}
    Popf;
End;
```

Pozn: Pokud by funkce MyAlg místo PushfCli volala jen instrukci CLI (zakázání přerušení) a místo Popf volala instrukci STI (povolení přerušení), mělo by to za následek to, že v proceduře MyAlg2 po zavolání MyAlg by bylo přerušení povolené, což ale nechceme. To znamená, že použití procedur PushfCli a Popf je výhodnější než jen CLI a STI, protože vrací příznaky procesoru do původního stavu.

Pozor! Programátor by si měl dát dobrý pozor, aby nezakazoval přerušení na příliš dlouhou dobu. Tím by mohlo dojít v lepším případě například k nezpracování přijatých znaků po sériové komunikaci nebo v horším případě k resetu procesoru díky neobčerstvené funkci WatchDog.

6. Funkce pro obsluhu systémového časovače

Pojmem systémový časovač se nazývá časovač 0 procesoru, který s frekvencí **ITimerClk** provádí dekrementování z hodnoty děliče (**ITCU** nebo jiné nastavené) na 0 (viz obrázek), přičemž při dosažení hodnoty 0 vygeneruje hardwarové přerušení IRQ 0 (Int 08h).



Výše zmíněný obrázek a popis platí pro PC, KitV40 a Kit386EXR. Pro **Kit188ER** funguje časovač nikoliv jako dekrementální, ale inkrementální, tj. z hodnoty 0 čítá do hodnoty ITCU a při jejím přetečení vygeneruje hardwarové přerušení IRQ 8 (Int 08h). Na to je třeba brát zřetel, zejména při používání funkce ReadTimer0 (viz níže).

6.1. Funkce SetTimer

```
procedure SetTimer(V:word);
```

Funkce nastaví dělič systémového časovače na hodnotu V, čímž se změní frekvence přerušování systémového časovače. **Pozor!** Hodnota tohoto děliče je závislá podle použitého řídicího systému. Proto se zpravidla používá formulace volání `SetTimer((ITCU div X) and $ffff)`, kde ITCU je implicitní hodnota děliče časovače pro daný řídicí systém odpovídající 55ms a X je požadovaný počet zrychlení časovače (např. pokud X=3 bude se přerušení od systémového časovače volat s periodou 55/3 ms).

6.2. Funkce GetTimer

```
function GetTimer:word;
```

Funkce vrátí aktuální hodnotu děliče systémového časovače. Vrátí-li funkce hodnotu 0, jedná se vlastně o hodnotu maximální, tj. 65536.

6.3. Funkce GetRealIRQTimer

```
function GetRealIRQTimer:real;
```

Funkce vrátí přesnou hodnotu aktuální periody (v ms) volání přerušení systémového časovače. Tuto periodu vypočítává pomocí konstant ActIRQpSec, ITCU a z GetTimer. Pokud je systémový časovač jednotkou Tick „zrychlen“, vrátí periodu takto zrychleného časovače.

6.4. Funkce ReadTimer0

```
function ReadTimer0:word;
```

Pro PC, KitV40 a Kit386EXR funkce vrátí aktuální hodnotu systémového časovače v počtu načítaných dekrementačních taktů, tj. kolik taktů zbývá do vyvolání přerušení od časovače.

V případě Kit188ER funkce vrátí aktuální hodnotu časovače v počtu inkrementačních taktů, tj. kolik taktů uplynulo od minulého přerušení od časovače.

7. Funkce pro zjištění parametrů řídicí desky

Upozornění: Protože BIOSy pro procesor V40 nepodporují nastavování dále uváděných hodnot, vrací tato jednotka typické hodnoty. Proto se může stát, že u zákaznických BIOS nebo konfigurací nebudou vracené hodnoty odpovídat skutečnosti.

7.1. Funkce GetCPUID

```
function GetCPUID:Byte;
```

Funkce vrací podle nastavení proměnné SystIdent kód procesoru osazeného v řídicí desce.

Kód procesoru	Typ procesoru
0	V40
1	AMD Am188ER
2	Intel 386EXR

tabulka 1: Kódování typu procesoru

7.2. Funkce GetFreqID

```
function GetFreqID:Byte;
```

Funkce vrací podle nastavení proměnné SystIdent kód frekvence procesoru.

Typ procesoru	Kód frekvence procesoru	Frekvence procesoru
V40	1	8 MHz
	2	16 MHz
Intel 386EX	1	25 MHz
	2	33 MHz
AMD Am188ER	1	40 MHz
	2	44,236 MHz
	3	48 MHz
	4	49,152 MHz

tabulka 2: Kódování frekvence procesoru

7.3. Funkce GetBoardID

```
function GetBoardID:Byte;
```

Funkce vrací podle nastavení proměnné SystIdent kód řídicí desky.

Kód řídicí desky	Řídicí deska
1	KitV40
2	JT283
3	Term03
4	Term10
5	Kit386EXR (s Term10 nebo bez)
6	Kit188ER (s Term10 nebo bez)

tabulka 3: Kódování typu řídicí desky

7.4. Funkce GetBaseAddr

```
function GetBaseAddr:LongInt;
```

Funkce vrací podle procesoru a nastavení parametrů paměti bázovou adresu paměti ROM.

7.5. Funkce GetBiosIdStr

```
function GetBiosIdStr(var aStr; aLen:Byte):Byte;
```

Funkce vrací identifikační řetězec BIOS do předaného bufferu aStr o velikosti aLen. Pokud délka předaného bufferu je menší než 31 znaků, vrací se 0. V opačném případě se vrací velikost nastaveného řetězce.

Popis identifikačního řetězce je v samostatných manuálech popisující BIOS vlastní řídicí desky, případně v souboru popis.txt přiloženému k BIOS.

7.6. Funkce GetConfiguration

```
function GetConfiguration:LongInt;
```

Funkce vrací konfiguraci nastavitelných portů a chování BIOS. Tato konfigurace je poplatná řídicí desce a procesoru.

Řídící deska	Řídící deska
KitV40, JT283, Term03, Term10	Nepodporují nastavování portů a BIOS
Kit188ER	0..2b -- hranice mezi Flash a RAM 0 - $40000 1 - $50000 2 - $60000 3 - $70000 4 - $80000 3b -- příznak překopírování Flash do RAM 0 - Aplikace se spustí z Flash 1 - Aplikace se zkopíruje do RAM a tam se spustí 8..11b -- velikost osazené Flash děleno 64KB př. 4 – 256KB
Kit386EXR	0b -- SSIO/Modem sig. at COM2 (DTR/,RTS/) 0 - Modem sig. at COM2 1 - SSIO 1b -- DMA1/Modem sig. at COM2 (TxD,RxD) 0 - Modem sig. at COM2 1 - DMA1 2b -- DMA2/Modem sig. at COM2 (DCD) 0 - Modem sig. at COM2 1 - DMA2 3b -- SMM/INT7 0 - SMM 1 - INT7 (pozor na INT7 -- standardně COM2 - nutno nastavit MCR1.3) 8b -- aktivace video služeb (VGA/Hercules) 0 - video služby neprovádí zápisy do VideoRAM 1 - video služby provádí zápisy do VideoRAM 14b -- rychlost procesoru (25/33MHz) 0 - 25 MHz 1 - 33 MHz

tabulka 4: Kódování konfiguračního slova

8. Funkce pro zjištění paměťového prostoru

V této kapitole se termínem **CS** bude označovat Chip Select pro přístup k jednotlivým částem pamětí.

Typ procesoru	Hodnota abCS	Typ paměti ovládané CS
V40	1	RAM
	2	ROM
188	1	RAM Low
	2	ROM High / RAM High *
	3	ROM Mid0 / RAM Mid0
	4	ROM Mid1 / RAM Mid1
	5	ROM Mid2 / RAM Mid2
	6	ROM Mid3 / RAM Mid3
386	1	RAM
	2	ROMLow
	3	ROMHigh
	4	CutROM

tabulka 5: Kódování CS podle typu procesoru

*** Jedná-li se o ROM či RAM závisí na konfiguračním bitu určujícím jestli se má aplikace kopírovat z Flash do RAM.**

Upozornění: Protože BIOSy pro procesor V40 nepodporují nastavování dále uváděných hodnot, vrací na KitV40 tyto funkce typické hodnoty. Proto se může stát, že u zákaznických BIOS nebo konfigurací nebudou vracené hodnoty odpovídat skutečnosti.

8.1. Funkce GetCSAddr

```
function GetCSAddr(abCS:Byte):LongInt;
```

Funkce vrací bázovou adresu paměti ovládané zadaným CS, který má různé významy podle typu procesoru.

Pozn. Popis CS a jeho typu je uveden v tabulce „tabulka 5: Kódování CS podle typu procesoru“.

8.2. Funkce GetCSSize

```
function GetCSSize(abCS:Byte):Word;
```

Funkce vrací velikosti paměti v KB ovládané zadaným CS, který má různé významy podle typu procesoru.

Pozn. Popis CS a jeho typu je uveden v tabulce „tabulka 5: Kódování CS podle typu procesoru“.

8.3. Funkce GetCSType

```
function GetCSType(abCS:Byte):Byte;
```

Funkce vrací kód typu paměti, viz kapitola „4.2 Typ paměti“, ovládané zadaným CS, který má různé významy podle typu procesoru.

Pro převod vráceného kódu lze použít konstanty hwsysXXX viz kapitola „4.2 Typ paměti“.

Pozn. Popis CS a jeho typu je uveden v tabulce „tabulka 5: Kódování CS podle typu procesoru“.

9. Funkce pro zjištění parametrů pamětí

Typ procesoru	Hodnota aSokl	Označení patic na řídicí desce
V40	1	RAM
	2	ROM
188, 386	1	ROM Low
	2	RAM Low
	3	ROM High
	4	RAM High

tabulka 6: Kódování paměťových patic podle typu procesoru

Upozornění: Protože BIOSy pro procesor V40 nepodporují nastavování dále uváděných hodnot, vrací na KitV40 tyto funkce typické hodnoty. Proto se může stát, že u zákaznických BIOS nebo konfigurací nebudou vracené hodnoty odpovídat skutečnosti.

9.1. Funkce GetOpMemory

```
function GetOpMemory(aSokl:Byte):Word;
```

Funkce vrací velikost v KB použitelné operační paměti v dané patici.

Pozn. Popis paměťových patic a jejich označení na řídicí desce je uveden v tabulce „tabulka 6: Kódování paměťových patic podle typu procesoru“.

9.2. Funkce GetChipSize

```
function GetChipSize(aSokl:Byte):Byte;
```

Funkce vrací velikost v KB paměťového čipu v zadané patici.

Pozn. Popis paměťových patic a jejich označení na řídicí desce je uveden v tabulce „tabulka 6: Kódování paměťových patic podle typu procesoru“.

9.3. Funkce GetMemoryType

```
function GetMemoryType(aSokl:Byte):Byte;
```

Funkce vrací kód typu paměťového čipu v zadané patici.

Pro převod vráceného kódu lze použít konstanty hwsysXXX, viz kapitola „4.2 Typ paměti“.

Pozn. Popis paměťových patic a jejich označení na řídicí desce je uveden v tabulce „tabulka 6: Kódování paměťových patic podle typu procesoru“.

9.4. Funkce GetAccessTime

```
function GetAccessTime(aSokl:Byte):Byte;
```

Funkce vrací přístupovou dobu paměti v ns pro zadanou patici.

Pozn. Popis paměťových patic a jejich označení na řídicí desce je uveden v tabulce „tabulka 6: Kódování paměťových patic podle typu procesoru“.

10. Funkce pro nastavení obsluhy chyb HEAP

10.1. Procedura SetHeapErrorHandler

```
procedure SetHeapErrorHandler;
```

Tato procedura nastaví obsluhu chyb při přetečení HEAP. Po jejím nastavení se při nedostatku paměti **nebude generovat RunError(203)**, ale funkce GetMem a New budou vracet NIL.

11. Ostatní funkce

11.1. Funkce MemCmp

```
function MemCmp(aBuff1, aBuff2: Pointer; aSize: Word): word;
```

Funkce porovná dva úseky paměti velikosti *aSize* na které ukazují ukazatele *aBuff1* a *aBuff2*. Pokud jsou obsahy těchto pamětí stejné, vrátí funkce hodnotu 0. Pokud jsou ale obsahy pamětí jiné, vrátí funkce číslo prvního rozdílného byte.

11.2. Funkce AbsAddrToPtr

```
function AbsAddrToPtr(A:longint):pointer;
```

Funkce převede absolutní (lineární) adresu zadanou v parametru *A* na ukazatel. Přípustný rozsah hodnot parametru *A* je od $0 do $FFFFF (1MB).

11.3. PtrToAbsAddr

```
function PtrToAbsAddr(P:pointer):longint;
```

Funkce převede ukazatel *P* na absolutní (lineární) adresu. Výsledek funkce bude v rozsahu od $0 do $FFFFF (1MB).