Příručka uživatele

u@8255sFTP

*Utilita pro **MDOS File Commander***

Základní informace

Utilita je určena pouze pro MFC verze 2 a vyšší. Jedničkové verze MFC byly testovací a věci okolo utilit ještě nebyly pořádně vyřešeny.

U utility **je přiložen její zdrojový text** ve formátu do Promethea (maska *p *.b*). Máte tak možnost podívat se, jak to vlastně funguje a větší šanci k tvorbě své vlastní a funkční utility. Pokud hodláte v tomto zdrojáku něco měnit (opravit, vylepšit atd.), pak pamatujte, že je slušnost **informovat o tom původního autora** (zvláště, když to pak hodláte šířit dál).

Kdybyste náhodou zapomněli, tak utilita se do MFC nahrává stiskem **SS+U** a spouští se klávesou **3** nebo **U**.

Utilita je uložena v paměti počítače pouze dočasně. MFC utilitu za určitých okolností (kopírování souborů) maže a je tedy třeba ji pak nahrát znovu.

Je možné pracovat pouze s JEDNOU konkrétní disketou a mechanikou. Nejde číst soubory např. z mechaniky A a zapisovat je na B. Nelze ani použít 2 různé diskety na stejné mechanice (tj., že z jedné diskety by se četlo a na druhou se zapisovalo). **Pokud disketu během práce v utility módu vyměníte za jinou, následky mohou být katastrofické (v nejhorším případě až ztráta veškerých dat na disketě)!**

Zápis dat (podprogram U_WBYTE) **funguje pouze na zx disketě.** Toto omezení lze však obejít tak, že se soubor zkopíruje z pc na zx disketu a budete pracovat tam (např. u konverze TAP2files).

u@8255sFTP

Utilita slouží pro přenos souborů mezi dvěma počítači přes obvod 8255 (po kabelu). Využití najde především v situacích, kdy se sejdou dva spectristé s naprosto nekompatibilními systémy a chtějí si vyměnit data. Např. D40/80<-->MB02, nebo paradoxně také D40(5,25")<-->D80(3,5"). V oblasti MB02 přislíbil podporu této utility Shrek (autor MB Commanderu)…

Snažte se zabránit situaci, aby se po propojení dvou 8255 (a zapnutí počítače) začaly obvody "přetlačovat", tj. aby byly stejné porty na obou 8255 nastaveny jako výstupní. Mohlo by dojít až ke zničení obvodů. Majitelé D40/80 mohou být celkem v klidu, protože MDOS 1 nechává interface zablokovaný a MDOS 2 vše nastavuje jak má (všechny porty jako vstupní).

Funkční klávesy:

CS+1	návrat do MFC
I	příjem souborů z druhého počítače
O	poslání souborů na druhý počítač
SS+Z	odblokování interface (nutné u MDOSu 1)
SS+X	povolení vypínání mechaniky při LOADu

Po spuštění utility musíte zvolit, jestli budete data vysílat ven (file OUT) nebo je odněkud přijímat (file IN). Jakmile zvolíte příslušnou volbu (nebo klasika CS+1 pro Escape…), dojde k nastavení obvodu 8255 a zobrazí se hláška "*Ready to work. Press any key.*". Pouze tehdy, když je toto hlášení zobrazeno **na obou** počítačích, můžete stisknout libovolnou klávesu (kromě CS+1 samozřejmě) pro zahájení přenosu. Přenos se dá přerušit stiskem SPACE (příp. BREAKu).

Jinak celé to pracuje obdobně jako např. v M602 na PC…

U MDOSu 1.0 je třeba ještě před přenosem odblokovat interface v disketové jednotce (pokud jste tak už neučili v Basicu před spuštěním MFC…), proto je zde volba SS+Z.

Pomocí SS+X můžete povolit, aby při načítaní souborů z diskety mechanika zhasínala. Moc Vám to však nedoporučuji.

Pokud nastane při ukládání na disketu nějaká chyba, utilita se vrátí do MFC. Může se Vám tak stát, že naposled ukládaný soubor nebude zcela v pořádku (nebude celý). Takovýto soubor poznáte podle toho, že **nebude mít nastaven žádný atribut** (HSAPRWED) a také by měl být vždy na poslední pozici ve výpisu.

Chybová (a jiná) hlášení:

Zpráva (hláška)	Význam
`8255 unblocked (OUT 153,16).`	Informace, že interface v D40/80 byl odblokován.
`Disk I/O error!`	Chyba při práci s disketou.
`OK. Stop drive when reading.`	Informace, že mechanika se bude při LOADu vypínat.
`Ready to work. Press any key`	Očekává se stisk libovolné klávesy (kromě CS+1). Poté začne samotný přenos.
`Receiving file:` *name*	Právě přijímaný soubor.
`Sending file:` *name*	Právě vysílaný soubor.
`Transfer error!`	Chyba při přenosu (break nebo error).
`Waiting (InfoPacket)...`	Přijímací počítač čeká na nějaké data z vysílacího počítače.

Pozn.: Při jakémkoliv návratu do MFC, jsou porty 8255 nastaveny jako vstupní (při spuštění této utility nebo po SS+Z rovněž).

Podrobnosti k utilitě (hlavně pro programátory)

Začnu raději úplně od začátku. **8255** je obvod hojně využívaný u ZXS a jeho hlavním úkolem je paralelní komunikace. K dispozici jsou 3 brány (A, B, C), přičemž kterákoliv z nich se dá nastavit buď jako vstupní (počítač data přijímá) nebo jako výstupní (počítač data posílá). **FTP** je zkratka z anglického **F**ile **T**ransfer **P**rotocol a jak už název napovídá, jde o protokol, který se stará o přenos souborů po síti.

Myšlenka napsat nějaký dokonalý přenos souborů i pro ZXS se mi honila v hlavě už dlouho, ale do kupy jsem to dal (díky Aragornovi) teprve nedávno. Mým cílem bylo stvořit něco, co by dovolilo přenos souborů přes obvod 8255 a to bez závislosti na konkrétním systému. Jinými slovy, je jedno jaký HW a systém mám (D40/80, MB02, DivIDE, dtpIDE…), pokud mám 8255 a vhodný program, můžu přenášet odkud chci a na co chci. A tak vznikla utilita U@8255sFTP a definice 4 vrstev (architektura celé "sítě"):

Level 3
Příprava souborů pro přenos a "vyšší" řízení samotného přenosu. Obstarává si sám aplikační program přičemž ale musí dodržovat definici vrstev Level 2 (packety) a Level 1 (návratové hodnoty, tj. flagy).

Level 2
sFTP (*speccy File Transfer Protocol*). Definice packetů.

Level 1
M-8255-TP (*MTs-8255-Transfer Protocol*). Nejnižší low level rutiny zajišťující spolehlivý přenos packetů mezi dvěmi 8255.

Level 0
Hardwarové propojení dvou 8255 (specifikace kabelu).

Specifikace propojovacího kabelu

Spojit dvě 8255 jde mnoha způsoby. Já jsem se snažil vymyslet takové, které by fungovalo i na Didaktiku GAMA (*jak známo, tam je C0 použito pro stránkovací účely, takže brána C3…0 nepřipadá v úvahu*) a také, aby to pak bylo i přehledné v samotném programovém kódu.

A0	*o----------o*	*A0*	
A1	*o----------o*	*A1*	
A2	*o----------o*	*A2*	
A3	*o----------o*	*A3*	
A4	*o----------o*	*A4*	
A5	*o----------o*	*A5*	
A6	*o----------o*	*A6*	
A7	*o----------o*	*A7*	*A0…7 – data (1 byte)*
B4	*o----------o*	*B4*	*busy/ready bit*
B5	*o----------o*	*B5*	*XOR error bit*
B7	*o----------o*	*B7*	*BREAK bit*
C4	*o----------o*	*C4*	*busy/ready bit*
C5	*o----------o*	*C5*	*XOR error bit*
C7	*o----------o*	*C7*	*BREAK bit*
GND	*o----------o*	*GND*	

Rozmístění signálů na konektoru interface

spoje	č. vývodu	součástky
+5V	1	PC0
IN1	2	OUT1
IN0	3	OUT0
výrez	*4*	*výrez*
PC6	5	**GND**
PC5	6	**PC7**
PC1	7	**PC4**
PC3	8	PC2
PB6	9	**PB7**
PB4	10	**PB5**
PB2	11	PB3
PB0	12	PB1
PA6	13	**PA7**
PA4	14	**PA5**
PA2	15	**PA3**
PA0	16	**PA1**

Kompakt, D40/80, M/P

spoje	č. vývodu	součástky
IN2	1	OUT2
IN1	2	OUT1
IN0	3	OUT0
výrez	*4*	*výrez*
PC6	5	**GND**
PC5	6	**PC7**
PC1	7	**PC4**
PC3	8	PC2
PB6	9	**PB7**
PB4	10	**PB5**
PB2	11	PB3
PB0	12	PB1
PA6	13	**PA7**
PA4	14	**PA5**
PA2	15	**PA3**
PA0	16	**PA1**

Gama '89

M-8255-TP (*MTs-8255-Transfer Protocol*)

Toto je jádro celého přenosu. Jde o 4 podprogramy v assembleru, které by neměly být nijak modifikovány (vylepšovány) – to aby nedošlo k situaci, že si nová verze nebude (časově) rozumět s tou starou.

```
SET_OUT  ld   a,%10000011     ;inicializace 8255 na vysílaní
         out  (127),a         ;A(out), B(in), C7…4(out), C3…0(in)
         nop
         xor  a               ;zpráva pro přijímač (bit4)
         out  (95),a
         ret

SET_IN   ld   a,%10011001     ;inicializace 8255 na přijímání
         out  (127),a         ;A(in), B(out), C7…4(in), C3…0(in)
         nop
         xor  a
         out  (63),a          ;zpráva pro vysílač (bit4)
         ret
```

SET_OUT (pro vysílající počítač) a SET_IN (pro přijímající počítač) **musí být zavolány před začátkem každého přenosu dat** (nemyslím tím před každým packetem, ale před započetím přenosu jako celku). Nastaví se jimi brány 8255, což je životně důležité pro správné fungovaní přenosu. Jakmile je provedena tato inicializace **na obou** počítačích, je možno použít následující 2 hlavní podprogramy:

BYTE_OUT – vyslání jednoho packetu

```
IN:     E – počet pokusů (pak se nastaví "storno")
        HL – ukazatel na začátek packetu (adresa v paměti)
        BC – délka packetu (0-65535 bytes)
OUT:    carry=1 – BREAK/storno/nelze doručit
        carry=0 – odesláno (a přijato) bez chyby
        HL, BC – stejné jako na vstupu
```

```
BYTE_OUT di                   ;přerušení musí být zakázané
         inc  bc              ;posílá se o byte navíc (konečný XOR)
OUT_AG   ld   d,0             ;počáteční hodnota pro XOR
         push bc
         push hl
GO_OUT   xor  a               ;příkaz přijímači ("čekej na byte")
         out  (95),a
WAIT_O1  ld   a,127
```

```
         in   a,(254)          ;je-li stištěn BREAK, tak vše stornuj
         rra
         jr   nc,BREA_O
         in   a,(63)           ;příkaz od přijímače
         bit  7,a              ;vše stornovat?
         jr   nz,BREA_O
         bit  4,a              ;je přijímač připraven?
         jr   z,WAIT_O1        ;není, takže se bude čekat
         dec  bc
         ld   a,b
         or   c
         ld   a,(hl)           ;vyzvedni byte z paměti
         jr   nz,OUT1          ;je-li v BC nula, musí se poslat XOR
         ld   a,d              ;XOR hodnota
OUT1     out  (31),a           ;pošli byte přijímači
         xor  d                ;nová XOR hodnota
         ld   d,a              ;ukládá se do D
         inc  hl
         ld   a,b
         or   c
         ld   a,16             ;příkaz přijímači ("vem si byte")
         jr   nz,OUT2          ;je-li v BC nula, vše bylo odesláno
         ld   a,16+32          ;změn příkaz ("ten byte je XOR")
OUT2     out  (95),a           ;pošli příkaz
WAIT_O2  ld   a,127
         in   a,(254)          ;BREAK znamená stornování všeho
         rra
         jr   nc,BREA_O
         in   a,(63)           ;příkaz od přijímače
         bit  7,a              ;vše stornovat?
         jr   nz,BREA_O
         bit  4,a              ;už ten odeslaný byte zpracoval?
         jr   nz,WAIT_O2       ;čeká se dokud přijímač byte nezpracuje
         bit  5,a              ;packet došel chybný?
         jr   nz,OUT_ERR       ;jestli jo, tak se pošle celý znova
         ld   a,b
         or   c
         jr   nz,GO_OUT        ;další byte
         xor  a                ;packet odeslán celý  a OK
OUT_END  out  (95),a
         pop  hl
         pop  bc
         ret

OUT_ERR  dec  e                ;v E je počet pokusů
         jr   z,BREA_O         ;pokusy vyčerpány takže storno
         pop  hl               ;obnov ukazatele
         pop  bc
         jr   OUT_AG           ;a zkus to znova

BREA_O   ld   a,128            ;příkaz pro přijímač ("storno")
         scf                   ;nastav carry
         jr   OUT_END          ;odešli příkaz a vše skonči
```

BYTE_IN – příjem jednoho packetu

```
IN:     HL – kam ukládat packet
OUT:    carry=1 – BREAK/storno/nelze přijmout bez chyb
        carry=0 – přijato bez chyby
        BC – délka packetu
        HL – stejné jako na vstupu
```

```
BYTE_IN  di                  ;přerušení musí být zakázané
         ld   bc,0           ;vynulujeme počítadlo
         ld   d,c            ;a taky počáteční XOR hodnotu
         push hl
GO_IN    ld   e,5            ;pauza aby se vysílač dostal
         ex   (sp),hl        ;do WAIT_O1 smyčky (5 je optimální
         ex   (sp),hl        ;hodnota)
         dec  e
         jr   nz,GO_IN+2
         ld   a,16           ;příkaz vysílači ("jsem ready")
         out  (63),a         ;pošli příkaz
WAIT_I1  ld   a,127
         in   a,(254)        ;je-li stištěn BREAK, tak vše stornuj
         rra
         jr   nc,BREA_I
         in   a,(95)         ;příkaz od vysílače
         bit  7,a            ;vše stornovat?
         jr   nz,BREA_I
         bit  4,a            ;už něco poslal?
         jr   z,WAIT_I1      ;když ne tak počkáme
         bit  5,a            ;byl to XOR byte?
         in   a,(31)         ;načti co poslal
         jr   nz,IN_ALL      ;s XOR bytem skočíme dál
         ld   (hl),a         ;ulož byte do paměti
         inc  hl
         inc  bc
         xor  d              ;vyxoruj byte
         ld   d,a            ;a schovej ho do D
         xor  a              ;příkaz vysílači ("zpracováno")
         out  (63),a         ;pošli příkaz
         jr   GO_IN          ;další byte

IN_ALL   cp   d              ;je XOR hodnota stejná s tou naši?
         ld   a,0            ;příkaz pro vysílač ("zpracováno")
         jr   z,IN_END       ;je stejná, takže pošli příkaz a konec
         ld   a,32           ;změn příkaz ("pošli to znova")
         out  (63),a         ;pošli příkaz
         pop  hl             ;obnov ukazatel do paměti
         jr   BYTE_IN        ;a zkusíme to znova

BREA_I   ld   a,128          ;příkaz pro vysílač ("storno")
         scf                 ;nastav carry
IN_END   out  (63),a         ;pošli příkaz
         pop  hl             ;obnov ukazatel do paměti
         ret
```

sFTP (*speccy File Transfer Protocol*)

Protokol definuje, jak mají vypadat jednotlivé packety a v jakém pořadí se mají posílat *(pozn.: slovo packet zde nevyjadřuje přesně totéž, co na PC; my tady budeme packet chápat jako část dat bez jakýchkoliv informací navíc)*. O chybovost se sFTP nestará, ta je ošetřena už na nejnižší vrstvě (Level 1).

1. InfoPacket (128 bytes)

Hlavička souboru, který se bude posílat.

Offsest	Délka	Význam
0	1	info byte (identifikace programu, který data posílá); nutné pokud se používá "reserved" místo 0 - "reserved" nepoužito (k dispozici pouze základní info o souboru) 1 - MFC (offset 32 až 63 = 32 bytes z DIR - MDOS) 2 - MFC (offset 32 až 63 = 32 bytes z DIR - MSDOS) 3-255 - zatím nevyužito
1	1	přípona souboru: "P" Basic Program "B" Bytes "N" Number array "C" Character array může být ale i jakýkoliv jiný kód ASCII (0-255); pokud bodyflag<>0, jde o bezhlavičkový blok dat bez ohledu na příponu!
2-11	10	jméno souboru může obsahovat kódy 0-255 "file1" a "file1 " jsou 2 naprosto rozdílné soubory (změna oproti TAP)!
12-15	4	délka souboru (32bit)
16,17	2	délka Basic ("P") programu bez proměnných (jinde nemá význam = 32768)
18,19	2	startovací adresa/řádek
20	1	BodyFlag (nenulové pouze u bezhlavičkového souboru!)
21-127	107	reserved (nepoužito) může libovolně používat aplikační program

Ať už existuje jakýkoliv FS a aplikační software, podprogramy pro přenos souborů **musí vždy** zajistit transformaci vlastností souboru do formátu uvedeného výše (offset 1-20). Je to nutné pro přenos souborů mezi různými systémy (tam offset 21-127 ztrácí význam).

Lze předpokládat, že v budoucnu bude existovat více druhů InfoPacketu (tzn. s jinou délkou než 128 bytes), které budou řešit narůstající požadavky uživatelů (např. podpora adresářů, diagnostika přenosové cesty, přenos celého image diskety). Proto InfoPacket o velikosti 128 bytes berte jako takovou první vlaštovku. Zbytek se vyspeccyfikuje časem. Počet druhů InfoPacketů je samozřejmě omezený – celkem jich může být maximálně 513.

2. DataPacket (1…512 bytes)

Vlastní data patřící souboru. Musí následovat po InfoPacketu o velikosti 128 bytes. **Délka DataPacketu** (ani žádného jiného packetu) **nesmí být větší než 512 bytes**. Přijímací počítač totiž příjme přesně tolik bytes, kolik odešle počítač odesílající a mohlo by tedy dojit ke zhroucení (vždy musíte počítat s tím, že přijímací počítač nemá pro data buffer větší než 512 bytes). Počet DataPacketů závisí na délce souboru.

3. CloseEndPacket (0 bytes)

Speciální varianta InfoPacketu. Říká, že soubor se má uzavřít, tj. že se už celý přenesl (ClosePacket). V určitých případech také znamená korektní ukončení celého přenosu, tj. všechny soubory už byly přeneseny (EndPacket).

Rozeznat pravý význam tohoto packetu není složité:

- čeká-li se na InfoPacket a přijde CloseEndPacket, pak se jedná o EndPacket

- čeká-li se na DataPacket a přijde CloseEndPacket, pak se jedná o ClosePacket

Po DataPacketu musí vždy následovat CloseEndPacket. Kdyby totiž ne a místo něj přišel InfoPacket, tak to program bude chápat jako další DataPacket a data přidá k souboru.

mts.zxs@tiscali.cz

ICQ: 144264603

(připomínky jsou vítány)