Retos Debugger

INTEGROVANÝ LADÍCÍ NÁSTROJ

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 1.88

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: 19.05.2003
Datum posledního uložení dokumentu: 19.05.2003
(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.
1.88	1.XX	Tu	19.05.2003	Úprava dokumentu dle ISO9000.

1.2. Účel dokumentu

Tento dokument slouží jako popis integrovaného vývojového nástroje sloužícího k ladění aplikačních programů vytvořených pro řídicí jednotky stavebnice KIT.

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 orientovat se v používání programového vybavení SofCon.

2. Termíny a definice

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

3. Úvod

Program ReTOS Debugger (RTD) je integrovaný vývojový nástroj sloužící k ladění aplikačních programů vytvořených pro řídicí jednotky stavebnice KIT. Pro uplatnění všech ladících možností RTD by aplikace měla používat realtimový operační systém ReTOS.

Aplikace je možno psát v jazyku Turbo Pascal na počítači IBM/PC. Na tomto počítači (dále označovaném jako **Master počítač**) lze spustit RTD přímo z prostředí Borland překladačů. Program RTD umožňuje přeloženou aplikaci ve tvaru .EXE souboru podle požadavků uživatele rozdělit na několik modulů (soubory BIN) a ty umístit na zadaném absolutní adresy. Vytvořené binární soubory je možné zavést z RTD do řídicího jednotky stavebnice KIT (dále označovaném jako **Slave počítač**), buď pomocí sériové linky nebo pomocí simulátoru EPROM pamětí.

Pomocí programu RTD lze aplikaci nejen zavést, ale i spustit nebo ladit za pomoci příkazů předávaných po sériové lince. Na straně Slave počítače komunikuje RTD buď s BIOS monitorem nebo Kernel monitorem (dodáván s firemním systémem reálného času ReTOS), který je součástí laděné aplikace a je tvořen speciálním procesem běžícím paralelně s aplikací. Při ladění je možno symbolicky zadávat breakpointy, symbolicky vypisovat a modifikovat globální a částečně také lokální proměnné a parametry. Dále lze monitorovat stav aplikace tzn. vypisovat fronty procesů, obsahy schránek atd., mimo jiné je možno číst a modifikovat IO prostor. Při vzniku run-time chyby v aplikaci je možno zpětně určit číslo chyby a programovou jednotku, dále unit, s číslem řádky, na kterém chyba vznikla. Po ukončení ladící relace se všechny důležité informace uloží do souboru typu .DST k dalšímu použití při příští ladící relaci.

4. Ovládání

V následujících odstavcích předpokládáme, že čtenář je obeznámen s realtimovým jádrem ReTOS v rozsahu v jakém je popsáno v uživatelském manuálu. Ovládání programu RTD je navrženo jako kompatibilní s integrovaným prostředím překladače Turbo Pascal.

Program je ovládán pomocí menu a horkých kláves. V hlavním menu programu se nacházejí příkazy pro vyvolání vedlejších menu: Hlavní menu obsahuje následující příkazy:

Place

- menu pro výběr aplikace, její umísťování a generování souborů typu .BIN

BIOS

- menu pro komunikaci s BIOS monitorem Slave počítače. Umožňuje zavedení aplikace po sériové lince, její spuštění, zjišťování informací o BIOSu, modifikace konfigurační tabulky (pouze u řídicích jednotek, které tuto tabulku podporují viz. manuál popisující BIOS), zápis souborů do paměti Slave počítače a čtení bloků paměti ze Slave počítače s jejich zápisem do souborů.

Kernel

- menu pro komunikaci s Kernel monitorem. Umožňuje nastavování, prohlížení a rušení breakpointů, suspendování a opětovné spuštění všech procesů aplikace a návrat do BIOS monitoru.

Dump

- menu umožňující prohlížet resp. nastavovat hodnoty proměnných v aplikaci. Je možno prohlížet stav všech procesů, vypisovat obsahy schránek, vypisovat resp. modifikovat obsahy globálních i lokálních proměnných a portů. Všechny výpisy je možno provádět symbolicky a v požadovaném formátu. Vybrané proměnné lze sledovat v okně Watch.

File

- standardní menu pro zobrazování souborů

Windows

- standardní menu pro ovládání oken

Options

- menu pro zadávání parametrů komunikace. Pokud používáme sériovou komunikaci mezi Master a Slave počítačem, je třeba po spuštění programu RTD nejprve nastavit parametry této komunikace. Parametry se zadávají příkazem Communication v menu Option. Po zadání příkazu je možno zadat používaný COM a určit příslušné přerušení. Implicitně je nastaven COM1 a přerušení IRQ4. Dále je možno zadat baudovou rychlost komunikace a to jak pro přenos normálních zpráv tak pro zavádění programu, které se implicitně provádí větší baudovou rychlostí.

Vzhledem k tomu, že program RTD může být provozován i v síti, nastavuje se také číslo uzlu v síti, které je přiřazeno Master počítači a Slave počítači, s kterým se bude komunikovat.

Nastavení parametrů komunikace musí souhlasit s nastavením parametrů při spuštění KernelMonitoru (viz dále).

Navázání komunikace se Slave počítačem je indikováno v pravé horní části obrazovky v modrém poli. Pokud RTD komunikuje s BIOS monitorem, je v tomto poli text "BIOS monitor". Je-li v tomto poli text "Disconnected", znamená to, že spojení se Slave počítačem je přerušeno a je třeba zkontrolovat spojovací kabel a nastavení parametrů v menu Option/Communication. Nastavené parametry se po ukončení programu automaticky uloží do souboru RTD.CFG, odkud se také při spuštění programu automaticky čtou.

5. Postup při ladění

5.1. Umístění aplikace

5.1.1. Popis příkazů pro umístění aplikace

Před spuštěním programu RTD musí být aplikace přeložena do tvaru .EXE a musí být vygenerována mapa aplikace (.MAP) a to nejlépe v detailním tvaru. Zdrojové texty programu spolu s příslušným souborem .EXE a mapou by měly být umístěny v aktuálním adresáři. Dokud není zadáno jméno aplikace (jméno příslušného souboru .EXE), je většina příkazů programu RTD zamaskována. V tomto stavu, ale při navázání komunikace s Slavem počítačem, lze provádět pouze výpisy a modifikaci paměti Slave počítače pomocí absolutních adres příkazem Dump a Modify/Evaluate v menu Dump (viz dále).

Zadání jména aplikace se provede výběrem příslušného souboru .EXE z nabídky souborů v okně Application Name. Po zadání jména aplikace se zkontroluje aktuálnost příslušné mapy aplikace. Pokud mapa není přítomna nebo není aktuální, je hlášena chyba a jméno aplikace zůstává nedefinované. V tomto případě zbývá jediná věc: opustit program RTD a přeložit aplikaci s parametrem pro generování mapy.

Je-li úspěšně zadáno jméno aplikace, snaží se RTD nalézt příslušný konfigurační soubor: RTD.CFG. Tento konfigurační soubor je generován automaticky po ukončení relace a uschovává všechny důležité informace programu RTD týkající se nastavení a parametrů příkazů, které jsou zadávány uživatelem. Není-li konfigurační soubor nalezen je zobrazena informační hláška a je vygenerován implicitní soubor RTD.CFG.

Zvolenou aplikaci, je potřeba umístit do adresního prostoru Slave počítače. K tomuto účelu slouží příkaz Memory Design, který se nachází v menu Place. Umístění musíme provést vždy tehdy, když vytváříme novou aplikaci, nebo tehdy, když chceme umístění aplikace změnit nebo když jsme do aplikace přidali (resp. z ní ubrali) unity. Jinak se popis umístění aplikace přečte z dalšího konfiguračního souboru: jméno_aplikace.DST, takže jej není třeba opakovaně zadávat. (Pozn.: Tento binární soubor .DST naleznete v textové podobě jako soubor s koncovkou .MEM.) Umístěná aplikace se skládá z modulů umístěných do pamětí RAM a ROM, případně FLASH. Každý modul může obsahovat libovolný počet unit a může být umístěn na požadované místo v paměti. Jediné co není dovoleno, aby se moduly obsahující data a kód aplikace překrývaly. Při vzniku této chyby RTD generuje varovnou hlášku.

Každý modul má přiřazenu zaváděcí adresu, délku, seznam unit, které modul obsahuje, a dále příznak, jedná-li se o modul typu ROM, RAM nebo IC. Jednotlivé moduly se vytvářejí v okně Memory List, které se otevře po zadání příkazu Memory Design.

Okno Memory List zobrazuje jméno modulu, zaváděcí adresu, délku a typ modulu (ROM, RAM nebo IC). Zaváděcí adresa každého modulu je udávána v ***paragrafech (16 bytů)*** a odpovídá tak hodnotě segmentových registrů. Ovšem jeho délka se počítá ***lineárně***. V okně Memory List je možno vyvolat příkazy Insert - vytvoření nového modulu, Delete - zrušení modulu a Edit - úprava již hotového modulu. Po zadání příkazu Insert se otevře okno Edit Memory, které umožňuje zadat parametry nového modulu. V okně Edit Memory je možno zadat jméno modulu (implicitně je stejné jako jméno aplikace), počáteční adresu modulu (údaj se zadává v paragrafech) a jeho délku (údaj se zadává v počtech BYTE). Maximální délka jednoho modulu je 20000h. Pokud zadáme číslo větší, je jeho délka v programu upravena, ale na monitoru se to neprojeví. V dolní části okna se zobrazuje velikost dosud neobsazené části modulu, do které mohou být umístěny další unity. Pokud je paměť modulu vyčerpána, je na příslušném místě zobrazeno "over". Dále se v okně Edit Memory zobrazují dva seznamy unit - unity již umístěné do modulů (Used - tento seznam je na začátku prázdný) a dále seznam dosud neumístěných unit (Unused). V okně Edit Memory je možno používat příkazy Insert a Delete. Příkazem Insert se přesune první unit ze seznamu Unused do seznamu Used. O délku této unity se zkrátí velikost dosud neobsazené části paměti modulu. Opačný postup nastane při zadání příkazu Delete.

Vytvoření modulu se potvrdí příkazem OK. Vytvořený modul je zařazen do seznamu modulů a je zobrazen v okně Memory List, kde ho lze modifikovat pomocí příkazu Edit. Pro editaci se opět využívá okno Edit Memory. Po umístění všech unit do modulů opustíme okno Memory List příkazem OK. Po umístění aplikace jsou odmaskovány ty příkazy, jejichž provedení vyžaduje umístění aplikace.

Po zadání všech požadovaných modulů je nejprve potřeba vygenerovat příslušné binární tvary všech kódových modulů. To provede příkaz Generate v menu Place. Během generování se zobrazují jména jednotlivých modulů a unit, které jsou v modulech uloženy. Pro moduly umístěné v paměti ROM se vygeneruje soubor: jméno_modulu.BIN, který obsahuje binární obraz příslušného modulu.

Pozn.: Počáteční adresa každého ROM modulu, který je zaváděn výše jak $80000 musí být dělitelná **$20** (v lineárních adresách). V případě, že tato počáteční adresa modulu není dělitelná **$20** (v lineárních adresách), je zobrazeno chybové hlášení a požadována oprava této adresy.

5.1.2. Popis typu modulů – RAM, ROM a IC

Unity, programové jednotky aplikace, se dělí na datové a kódové. Datovými unitami nazýváme unity s názvem Data, Stack a Heap. Všechny ostatní unity jsou kódové. Datové a kódové unity se nesmí umístit současně do jednoho modulu, který může být buď typu ROM, IC nebo RAM.

Jak už bylo uvedeno existují tyto typy modulů: moduly typu RAM, do kterých se umísťují pouze datové unity (Data, Stack a Heap), moduly typu ROM, do kterých se umísťují pouze kódové unity a modul typu IC, do kterého se umisťují zvláštní jednotky, které vyžadují, aby běžely v paměti RAM. (V dnešní době v modulu typu IC musí být umístěna jednotka Graph a ScGraph a v aplikaci může být pouze jeden modul tohoto typu.) Kódové unity odpovídají programovým jednotkám Turbo Pascalu. Kromě nich existuje ještě služební jednotky InitData, Loader a InitCode (tato jednotka se vytváří pouze existujeli modul typu IC). Modul obsahující jednotku Loader je dále nazýván spustitelným modulem.

Programová jednotka Loader obsahuje úvodní hlavičku aplikace s její startovací adresou a proto musí být umístěna v paměti od adresy $C000:0000 do adresy začátku BIOSu. (Pozn.: Začátek BIOSu se pro různé řídicí jednotky stavebnice KIT liší. Pro KitV40 je začátek BIOSu na adrese $FE00:0000 a pro Kit386EXR na adrese $FD00:0000). Pouze v této oblasti BIOS hledá spustitelné ROM moduly po RESET. Z nalezených platných ROM modulů je vždy, ale spuštěn pouze poslední. Při umisťování modulu je třeba dávat pozor na konflikt s BIOSem grafických karet, který je umístěn v rozmezí adres $C000:0000 do $C700:0FFF, tj. Váš první spustitelný modul musí být umístěn na adrese $C800:0000. Pro zrušení paměťového prostoru rezervovaného pro grafické karty je nutno provést zásahy, které se liší pro KitV40 a pro Kit386EXR, viz. dále

Velikost jednotky Heap je nastavena implicitně do konce paměti modulu, ve kterém je jednotka Heap umístěna.

Upozornění: Pokud Kit386EXR nabootuje v default módu, tj. identifikační řetězec obsahuje *Def*, pak se po restartu nikdy nespustí aplikace, ale BIOS Monitor. Tento mód slouží pouze k novému nastavení konfigurační tabulky.

Pozn.: V seznamu unit se může vyskytnout unit s nulovou délkou kódu. To není chybou. Z unity se může využívat například pouze nějaká definice konstant, nebo něco podobného.

Pozn.: Modul typu IC musí být vždy umístěn v paměťovém prostoru RAM, zpravidla je segment menší než $8000.

Pozn.: Počáteční adresa každého ROM modulu, který je zaváděn výše jak $80000 musí být dělitelná **$20** (v lineárních adresách). V případě, že tato počáteční adresa modulu není dělitelná **$20** (v lineárních adresách), je zobrazeno chybové hlášení a požadována oprava této adresy.

Pozn. *Zrušení rezervace paměťového prostoru pro grafické karty:*

- *pouze pro KitV40*: Na požádání lze dostat upravenou verzi řadiče sběrnic v programovatelném zaměnitelném obvodu GAL, který neuvolňuje prostor pro grafické karty. Touto úpravou získáte paměť, která je standardně vyhrazena pro grafické karty. Tuto změnu doporučujeme pouze v případech, kdy nechcete používat KitV40 s grafickou kartu.
- *pouze pro Kit386EXR*: V systémech s Kit386EXR se vyřazení rezervace paměti pro grafické karty provede změnou konfigurační tabulky. Tato tabulka je podrobně popsána v manuálu BIOS386EXR a změna tabulky je popsána dále.

5.2. Zavedení aplikace do paměti RAM

Po vygenerování binárního obrazu všech modulů (popis v kapitole Umístění aplikace) je možno zavést soubory typu BIN do Slave počítače. K tomu je možno použít buď sériové linky nebo externího simulátoru paměti EPROM. Před popisem postupu pro zavedení aplikace do paměti RAM budou ještě popsány možné stavy RTD. Tyto stavy jsou zobrazovány v pravé horní části obrazovky v modrém poli. Tyto stavy informují uživatele o tom, s čím RTD právě komunikuje.

Pokud RTD komunikuje s BIOS monitorem, je v tomto poli text "BIOS monitor", pokud komunikuje s Kernel monitorem, je v tomto poli text "Kernel: xxxxx"(xxxxx vždy nahrazuje running, suspend, checksum). Je-li v tomto poli text "Disconnected", znamená to, že spojení se Slave počítačem je přerušeno a je třeba zkontrolovat spojovací kabel a nastavení parametrů v menu Option, kde se zadávají parametry komunikace (viz dále).

Zavedení aplikace je možné pouze v případě, že v pravém horním rohu RTD je zobrazen text "BIOS monitor" nebo "Kernel: xxxxx". Spuštění zavedení aplikace se provede příkazem LOAD v menu BIOS. Během přenosu programu po sériové lince se v dialogovém okně zobrazuje postup přenosu. Přenos lze kdykoliv přerušit stisknutím Ctrl-Break.

Pozn.: Pro přenos programu po sériové lince lze nastavit vyšší baudovou rychlost než rychlost komunikace Rtd a BIOS monitoru (viz příkaz Communication v menu Option).

Pozn.: Z principu funkce paměti FLASH je doba zápisu dvou stejných programů do paměti RAM a FLASH odlišná.

5.3. Zavedení aplikace do paměti FLASH

Pro postup zavedení aplikace do paměti FLASH platí stejná pravidla jako pro zavedení aplikace do paměti RAM. Tento přístup umožňuje, aby program byl laděn v paměti RAM a poté pouze změnou zaváděcích adres jednotlivých modulů aplikace, viz. kapitola Umístění aplikace, tuto aplikaci nahrát do paměti FLASH.

Dále bude popis zaměřen na zvláštnosti pamětí FLASH, které je třeba vzít do úvahy při zavádění aplikace do paměti FLASH.

RTD umožňuje nahrávat aplikaci přímo do pamětí FLASH typu ATMEL 29CO10, 29CO20, 29CO40A. Dle typu paměti jsou fyzicky adresovány od $E0000h - $FFFFFh, $C0000h -$FFFFFh, $80000h - $FFFFFh. Z tohoto prostoru je pro uživatele standardně nedostupná paměť od $FE000h - $FFFFFh, oblast BIOS monitoru a oblast $A0000h - $C7FFFh vyhrazená pro grafické karty.

Pozn.: Spustitelný modul musí být umístěn v paměťové oblasti od $C0000h do začátku BIOSu, viz. Popis typu modulů – RAM, ROM. (Pozn.: Začátek BIOSu se pro různé řídicí jednotky stavebnice KIT liší. Pro KitV40 je začátek BIOSu na adrese $FE000h a pro Kit386EXR na adrese $FD000h).

Pozn. *Zrušení rezervace paměťového prostoru pro grafické karty:*

- *pouze pro KitV40*: Na požádání lze dostat upravenou verzi řadiče sběrnic v programovatelném zaměnitelném obvodu GAL, který neuvolňuje prostor pro grafické karty. Touto úpravou získáte paměť, která je standardně vyhrazena pro grafické karty. Tuto změnu doporučujeme pouze v případech, kdy nechcete používat KitV40 s grafickou kartu.
- *pouze pro Kit386EXR*: V systémech s Kit386EXR se vyřazení rezervace paměti pro grafické karty provede změnou konfigurační tabulky. Tato tabulka je podrobně popsána v manuálu BIOS386EXR a změna tabulky je popsána dále.

5.4. Zavedení aplikace do simulátoru EPROM

Pro zavedení aplikace do simulátoru paměti EPROM slouží okno EPROM simulátor, které obsahuje příkazovou řádku pro EPROM simulátor. V příkazové řádce je možno použít makra $FILEn a $SEGn, za které RTD dosadí skutečná jména souborů typu .BIN, ve kterých jsou uloženy obsahy příslušných modulů a jejich zaváděcí adresy. Součástí okna je také bázová adresa EPROM simulátoru. Všechny adresy jsou zadávány v paragrafech (segmentech). Použijeme-li tedy v příkazové řádce makra $FILE1 a $SEG1, nahradí RTD první z nich jménem souboru typu .BIN, ve kterém je uložen binární tvar prvního modulu typu ROM a makro $SEG1 nahradí hexadecimálním tvarem zaváděcí adresy tohoto modulu, od které se napřed odečte bázovou adresu simulátoru paměti EPROM. EPROM simulátor dostane tedy zaváděcí adresu jednotlivých modulů relativně vůči své bázové adrese. Po zavedení aplikace do Slave počítače (resp. do EPROM simulátoru) je aplikace připravena ke spuštění. Již v tomto okamžiku je možno provádět výpisy resp. modifikaci dat aplikace pomocí příkazů menu Dump. Z důvodu, že kód je umístěn v ROM paměti, nelze KernelMonitorem nastavit breakpointy.

5.5. Odstartování aplikace

Aplikaci je možno odstartovat příkazem GO z menu BIOS, který požádá o zadání absolutní adresy a provede skok na tuto adresu. Vhodnější způsob je však použití příkazu Run (Ctrl-F9), který se také nachází v menu BIOS a provádí implicitně skok na startovací adresu aplikace.

Po provedení příkazu Run se nejprve rozběhne speciální proces tzv. Kernel monitor, pokud je v aplikaci přítomen. Tento proces umožňuje komunikaci s Master počítačem a provádí příkazy programu RTD. Kernel monitor je standardní proces jádra ReTOS, který má přiřazenu maximální statickou prioritu (tuto hodnotu nelze nastavit standardními prostředky), takže žádný proces aplikace nemůže mít stejnou nebo větší statickou prioritou. Pokud je v aplikaci v parametrech KernelMonitoru, zadáno spuštění na dotaz. Je nejdříve spuštěn KernelMonitor a po chvíli se zobrazí dotazovací okno, má-li být spuštěna vlastní aplikace. Pokud spuštění odmítneme (příkazem No resp. Esc), objeví se ve stavovém poli vpravo nahoře nápis "Kernel suspend". Ve stavu suspend jsou všechny procesy s výjimkou Kernel monitoru pozastaveny a tento stav můžeme využít k nastavení breakpointů. Definitivní

odstavování aplikace se provede příkazem Continue v menu Kernel (Ctrl-F9). Ten odstartuje proces Main a tím se rozběhne celá aplikace. Po spuštění všech procesů se v pravém horním rohu zobrazí text "Kernel running". Tento text svítí i v případě, že v aplikaci běží alespoň jeden proces.

5.6. Přerušení a obnovení chodu aplikace

Procesy aplikace je možno kdykoliv (Pokud je přítomen KernelMonitor) převést do stavu suspend příkazem SuspendAll a opět kdykoliv rozběhnout příkazem Continue. Pozn.: Horká klávesa Ctrl-F9 se chápe podle stavu aplikace, buď jako příkaz Run pro počáteční odstartování Kernel monitoru nebo jako příkaz Continue pro ukončení stavu suspend a nebo jako příkaz pro pokračování procesu, který je pozastaven na breakpointu.

Nastane-li v aplikaci runtimová chyba, předá se hlášení o této chybě do Master počítače. Program RTD se pokusí otevřít příslušný zdrojový soubor a nastavit v něm kurzor na místo vzniku chyby. Pokud se to nepovede, ohlásí alespoň jméno souboru a příslušnou řádku, na které k chybě došlo. Vlastní aplikace je zresetována a řízení je předáno BIOS monitoru. Běh aplikace se násilně ukončí příkazem Reset. Na tento příkaz provede KernelMonitor proceduru Halt. Ta způsobí vyvolání řetězu Exit procedur aplikace a následný přechod do BIOS monitoru. Při provedení příkazu Reset v menu Kernel se všechny breakpointy z aplikace odstraní, avšak RTD si je dále pamatuje. Pokud provedeme opětovný start, jsou breakpointy znovu automaticky nahrány ještě před spuštěním aplikace. Při opouštění programu RTD se breakpointy automaticky ukládají do souboru desktop typu .DST.

Kdykoliv za běhu aplikace je možno zkontrolovat checksum aplikace příkazem Checksum v menu Kernel. Při používání funkce CheckSum a práci s breakpointy je na tomto místě třeba vysvětlit jak breakpointy pracují, aby byly zřejmá některá hlášení.

Pokud se kód aplikace nachází v RAM paměti lze nastavovat breakpointy. Po nastavení breakpointu do procesu, blíže kapitola Breakpointy, se do kódu aplikace na místo breakpointu vloží rutina, která volá službu jádra KernelMonitoru, ve které se rozhoduje zda volající proces má být zastaven. (Pozn. Kód může být sdílen více procesy.) V případě, že volající proces je stejný s procesem, u kterého je nastaven breakpoint, je tento proces zastaven. V opačném případě proces pokračuje v provádění kódu. Z tohoto výkladu už lze tušit, že provádění checksum přes tento pozměněný kód nebude vždy bez problémů. Tzn. pokud bude program legálně ukončen buď pomocí příkazu Reset v programu RTD nebo pomocí exit procedury jádra je program vrácen do původního stavu, obslužné breakpoint rutiny jádra jsou nahrazeny původními instrukcemi. Pokud ale aplikace bude ukončena tvrdým Reset nebo vypnutím napájení, je tento kód při dalším spuštění poškozen a nelze ho spustit. Tzn. aplikace v místech breakpointů místo původních instrukcí obsahuje volání obslužných rutin jádra, které po novém spuštění nemají inicializovány žádné struktury a jejich aktivace by vedla ke zhroucení aplikace. Proto je před každým spuštěním proveden checksum, který provede kontrolu a je-li vše v pořádku aplikaci je spuštěna. V opačném případě je zobrazena chyba checksum.

V případě, že je spuštěna funkce Checksum a v aplikaci jsou použity breakpointy, se kód vrátí po dobu výpočtu checksum do původního stavu a po ukončení výpočtu checksum jsou do kódu navráceny breakpointy.

Z tohoto výkladu je také patrné, že breakpointy z RTD nelze umisťovat do aplikace umístěné v paměti typu ROM, ale pouze do paměti typu RAM. Jediný možný

způsob jak pracovat s breakpointy v aplikaci, která je umístěna v paměti typu ROM, je breakpointy připravit přímo v kódu a tento kód nahrát do EPROM případně FLASH, viz. kapitola Ladění v EPROM a FLASH paměti.

Pro urychlení práce je možno použít příkaz GenLoadRun, který provede postupně příkazy Generate, Load, a Run.

Z důvodu omezené zapisovatelnosti do paměti FLASH lze s breakpointy pracovat pouze jako s pamětí typu EPROM.

5.7. BreakPointy

Breakpointy je možno zadávat pouze pokud aplikace obsahuje ReTOS. Při zadávání breakpointů může být základní proces ReTOS tzv. Kernel monitor jak ve stavu running, tak ve stavu suspend.

Zadávat breakpointy lze přímo do příslušného zdrojového souboru v okně otevřené pomocí (klávesy F3) nebo pomocí adres. Breakpoint se zadává a ruší standardním způsobem horkou klávesou Ctrl-F8. Protože však v jádře ReTOS může více procesů sdílet stejný kód, je třeba ještě jednoznačně určit proces, který na zadaném breakpointu zastaví. Toto určení se provádí zadáním jména procesu. Tzn. proces, který bude mít shodné jméno jako breakpoint bude zastaven a ostatní procesy budou nastavený breakpoint ignorovat. Pokud je proces na breakpointu zastaven, je zobrazena informační zpráva a pokud je potřeba, otevře se příslušný zdrojový soubor a kurzor se nastaví na dosažený breakpoint.

Pokračování procesu, který je pozastaven na breakpointu se dosáhne tak, že v okně zobrazujícím zdrojový soubor na zvolený breakpoint najedeme kurzorem a stiskneme klávesu Ctrl-F9. Zrušení breakpointu se provede opět klávesou Ctrl-F8. Pokud zrušíme breakpoint, na kterém je pozastaven některý proces, pokračuje proces v činnosti. O aktuálním stavu všech breakpointů nás informuje okno Breakpoints, které obsahuje seznam všech nastavených breakpointů. Pro každý breakpoint je uveden zdrojový soubor, řádka, kde se breakpoint nachází, jméno procesu, pro který je breakpoint nastaven a dále absolutní adresa breakpointu.

Aktivní breakpointy, na kterých je pozastaven některý proces, jsou označeny hvězdičkou. V okně Breakpoints je možno zadávat příkazy Insert, Delete, Continue, View, Registers a DeleteAll. Příkaz Insert slouží pro zadávání breakpointů pomocí absolutní adresy. Příkaz Delete slouží ke zrušení vybraného breakpointu a příkaz DeleteAll ke zrušení všech breakpointů. Příkaz Continue způsobí, že proces pozastavený na některém breakpointu pokračuje, avšak breakpoint se nezruší. Příkaz View zobrazí příslušný soubor a řádku, na které je breakpoint nastaven a příkaz Registers zobrazí hodnoty, které byly v registrech v okamžiku, kdy breakpoint nastal. Další informace o breakpointech jsou uvedeny v kapitole Přerušení a obnovení chodu aplikace.

5.8. Ladění v EPROM a FLASH paměti

KernelMonitor umožňuje také ladění v EPROM a FLASH paměti. V takovém případě je, ale třeba si do zdrojového programu předem připravit breakpointy (např. pomocí příkazu asm nebo pomocí inline procedury).

Při vlastním ladění, ale Kernel monitor ignoruje všechny breakpointy, které explicitně nenastavil. To znamená, že procesy se na předem vytvořených

breakpointech nezastavují, dokud je nenastavíme znovu v Master počítači, aby v RTD mohly být znovu vytvořeny struktury potřebné při práci s breakpointy.

5.9. Ukončení aplikace

Běh aplikace se ukončí příkazem Reset. Na tento příkaz provede Kernel Monitor proceduru Halt. Ta způsobí vyvolání řetězu Exit procedur aplikace a následný přechod do BIOS monitoru. Při provedení příkazu Reset v menu Kernel se všechny breakpointy z aplikace odstraní, avšak RTD si je dále pamatuje. Pokud provedeme opětovný start, jsou breakpointy znovu automaticky nahrány ještě před spuštěním aplikace. Při opouštění programu RTD se breakpointy automaticky ukládají do souboru desktop typu .DST.

5.10. Zobrazení dat aplikace

V menu Dump se nacházejí všechny příkazy pro zobrazování dat aplikace. Příkazy pro výpis jsou povolené pouze tehdy, když je navázána komunikace s BIOS monitorem nebo Kernel monitorem. Příkazy pro výpis procesů a schránek jsou dostupné pouze tehdy, když RTD komunikuje s Kernel monitorem. Příkaz Process slouží k výpisu všech procesů, které jsou spuštěny v jádře ReTOS. Po zadání příkazu se požádá o jméno procesu, který má být monitorován. Jméno může být nejednoznačné, takže pomocí jména "*" lze vypisovat všechny procesy. Nejprve se vypisují procesy z fronty Ready a potom z fronty Delay. Kromě jména a statické a dynamické priority se ve frontě Ready vypisují tzv. časové pokuty, které se zvyšují pokaždé, když je procesu přiřazen procesor. Ve frontě Delay se zobrazuje absolutní čas, na který proces čeká buď v proceduře Wait (WaitUntil) nebo v operaci s timeoutem. Pokud proces provádí operaci receive bez timeoutu nebo operaci send na synchronní schránku, je v příslušném sloupci poznamenáno Receive/Send. U procesů, které stojí na breakpointu, je poznamenáno Breakpoint. Příkaz MailBox slouží k výpisu schránek. Schránka se zadává jednoznačným jménem. U schránky se vypisují jména všech procesů, které buď čekají na příjem nebo na odevzdání zprávy. Dále se vypisují všechny zprávy resp. jejich počet u prázdných zpráv. Zprávy je možno vypisovat ve zvoleném formátu.

Příkaz Dump slouží k hexadecimálnímu výpisu bloku paměti. Blok se zadává adresou a délkou. Adresa se zadává jako dvojice hexadecimálních čísel segment:offset (neuvádí se znak $). Segment je možno kromě absolutního vyjádření rovněž zadávat symbolicky. Pro data z Data segmentu je možno použít segment ve tvaru DS, tedy například DS:1000. Aby bylo možno zadávat adresu v tomto tvaru, musí být již program umístěn příkazem Set Memory. Kromě symbolu DS je možno použít také symbol IOB nebo IOW pro výpis IO prostoru. Pokud je offset IOB pak se čte nebo zapisuje do IO prostoru byte a při offsetu IOW word.

Ve speciálním případě je také možno provádět výpis lokálních proměnných a parametrů procedur. V tomto případě musí být příslušná procedura pozastavena na breakpointu. Adresa se udává ve tvaru: #jméno procesu:offset, kde jméno procesu označuje ten proces, v němž byla procedura vyvolána, a který musí být pozastaven na breakpointu. Offset je v tomto případě interpretován relativně vůči registru BP. Výpočet skutečného offsetu se provádí modulo 64KB, takže např. adresa pro výpis první lokální proměnné procedury z procesu P se zadá ve tvaru #P:FFFE (Na adrese #P:0 je uložena stará hodnota BP).

Příkaz Evaluate slouží k symbolickému výpisu a modifikaci proměnných. Po zadání tohoto příkazu se otevře okno Evaluate & Modify. V dolní části okna se zobrazuje seznam všech globálních proměnných aplikace. Pro nastavení kurzoru na kteroukoliv proměnnou se adresa a jméno proměnné objeví v příslušných položkách okna.

Před výpisem proměnné je nejprve třeba zadat požadovaný formát výpisu. Formáty se dělí na jednoduché a složené. Jednoduché formáty se označují znaky dle následující tabulky:

Rozsah čísel \ Formát výpisu (číselný typ) \ (číselná soustava)	Hexadecimální tvar (HEXA)	Desítkový tvar
0 až 255	BX	B
-128 až 127	SX	S
0 až 65535	WX	W
-32768 až 32768	IX	I
-2147483648 až 2147483647	LX	L
reálná čísla	vždy pouze desítkově	R[:h1:h2]

Použité formátovací parametry u reálných čísel mají následující význam:

h1 .. počet desetinných míst před desetinnou tečkou.

h2 .. počet desetinných míst za desetinnou tečkou.

Např.:

Výpis proměnné ve formátu Shortint v desítkové soustavě je při zadání formátu S, při potřebě provést výpis v hexadecimální soustavě se zadá SX. Podobně se postupuje při Longint, kdy se zadá L při výpisu v desítkové soustavě a LX v hexadecimální soustavě.

Struktury a nehomogenní pole se popisují jako seznam formátů v kulatých závorkách oddělených čárkami . Výpis začíná na zadané adrese a položky jsou vypisovány podle typu a pořadí v závorkách s inkrementovanou adresou. Pro výpis ukazatele ve tvaru (offset, segment) použijeme formát: (wx, wx).

Např.:

Výpis nehomogenní pole obsahující word, byte a short int je při zadaném formátu (w,b,s) v desítkové soustavě, při výpisu v hexadecimální soustavě se ke každému formátu přidá X.

Homogenní pole se popisují formátem ve tvaru: [dolní mez .. horní mez] formát proměnné podle tabulky popisující jednotlivé formáty číselných typů. Ukazatel lze tedy vypsat také jako pole dvou slov pomocí formátu: [1..2]wx.

Např.:

Výpis pole byte o třech položkách se provede zadáním formátu [0..3]B v desítkové soustavě, v hexadecimální soustavě se přidá X.

Po zadání adresy, formátu a přístupu se výpis proměnné provede příkazem Evaluate. Hodnota proměnné se zobrazí v řádce Value. Pokud chceme hodnotu proměnné změnit, napíšeme požadovanou hodnotu do položky Value a provedeme příkaz Modify. Hodnoty se zadávají ve stejném formátu, v jakém se vypisují. Příkaz AddWatch v menu Dump slouží k přidání položky do okna Watch. Po zadání příkazu se otevře okno AddWatch, které má obdobný tvar jako okno Evaluate & Modify. Na rozdíl od něho se však po zadání příkazu OK okno AddWatch uzavře, otevře se okno Watch a proměnná se přidá k již zobrazovaným proměnným. Rušení proměnné v okně Watch se provede pomocí kurzoru a klávesy Del. Je-li okno Watch uzavřeno, je

možno jej kdykoliv znovu otevřít příkazem Watch. Zobrazení všech proměnných a stavů aplikace se provádí pouze jednorázově při otevření příslušného okna. Okna Watch a Dump jsou automaticky obnovena provedeme-li v okně Evaluate & Modify příkaz Modify. Vybrané okno je možno obnovit příkazem Redraw (F7). Příkazem RedrawAll (F8) se obnoví všechna otevřená okna, která zobrazují stav aplikace. Příkazem AutoRefresh je možno provádět automatické obnovování všech oken, které zobrazují stav aplikace; tedy oken Dump, Watch, Processes a MailBox. Interval obnovy je nastavitelný od 2 do 98 sekund, implicitně je 30 sekund.

Pokud je otevřeno více oken a interval obnovování je krátký, může dojít k zahlcení komunikace a okna se přestanou obnovovat. V takovém případě je třeba interval přiměřeně prodloužit a vyčkat, až se komunikace zotaví.

Potřebujeme-li zjistit zdrojový soubor a číslo řádky příslušející absolutní adrese z ROM paměti aplikace, můžeme použít příkaz Find error z menu Place.

5.11. TerminalIn a TerminalOut

Příkazy TerminalIn a TerminalOut umožňují simulovat pro aplikaci terminál. V aplikaci můžeme definovat výstupní textový soubor voláním procedury AssignRemote. Zápis do takového souboru se převede na zaslání zprávy do Master počítače a její zobrazení v okně Terminal. Pokud toto okno není otevřeno, zprávy se ztrácejí. Při čtení z tohoto souboru otevře RTD vstupní okno a očekává zadání řetězu. Zadaný text je potom zaslán aplikaci jako vstup ze souboru. Pokud vstupní okno opustíme klávesou ESC, žádný text se nepřenese a proces, který provedl příkaz Read na vstupní soubor, je pozastaven. Běh tohoto procesu je možno kdykoliv obnovit příkazem TerminalIn. Po zadání tohoto příkazu se znovu zobrazí vstupní okno a je možno text dodatečně zadat.

Procedura AssignRemote definující soubor, který reprezentuje vzdálený terminál, je součástí jednotky KernMon.

```
procedure AssignRemote(var F : Text);
```

6. Parametry komunikace

Parametry komunikace se zadávají příkazem Communication v menu Option. Po zadání příkazu je možno zadat parametry komunikace, tj. COM a příslušné přerušení.

Např.:

Definice standardních portů ‘COM=1 IRQ=4 Bd=4800 BdLd=19200’ nebo
‘COM=2 IRQ=3 Bd=4800 BdLd=19200’

COM=1	zadání portu
IRQ=4	zadání přerušení
Bd=4800	komunikační rychlost s BIOS a Kernel monitorem
BdLd=19200	rychlost zavedení programu do pamětí

Definice nestandardního portu ‘ADD=$110 IRQ=5 Bd=4800 BdLd=38400’

ADD=$110 zadání adresy portu

IRQ=5 zadání přerušení

Bd=4800 komunikační rychlost s BIOS a Kernel monitorem

BdLd=38400 rychlost zavádění programu do pamětí

(pouze KIT V40/16MHz, KIT386EXR)

Pro Kit386EXR lze použít i baudovou rychlost 57600 nebo 115200.

Implicitně je nastaven COM1 a přerušení IRQ4. Dále je možno zadat baudovou rychlost komunikace a to jak pro přenos normálních zpráv tak pro zavádění programu, které se implicitně provádí větší baudovou rychlostí. Nastavitelné přenosové rychlosti pro řídicí desky stavebnice KIT jsou následující:

Baudová rychlost / Řídicí systém	Nižší baudové rychlosti	19200 Bd	38400 Bd	57600 Bd	115200 Bd
KITV40/8	ano	ano	ne	ne	ne
KITV40/16	ano	ano	ano	ne	ne
KIT386EXR	ano	ano	ano	ano	ano

Baudová rychlost pro komunikaci s biosem KitV40 je v BIOSu pevně nastavena na 4800Bd, jiné nastavení tedy nemá smysl.

Vzhledem k tomu, že program RTD může být provozován i v síti, nastavuje se také číslo uzlu v síti, které je přiřazeno Master počítači a Slave počítači. Nastavení parametrů komunikace musí souhlasit s nastavením parametrů při spuštění KernelMonitoru. Implicitní nastavení je NOD=255, DNO=254. Provozováním v síti se míní, že v uživatelské síti shodím jeden (pouze jeden) procesor do biosu a potom s ním mohu komunikovat pomocí RetosDebuggeru pod číslem uzlu 255. Toto číslo a také číslo uzlu RetosDebuggeru je nastaveno napevno v biosu, takže jiné než toto nastavení nemá smysl.

Aby aplikace mohla být laděna pod programem RTD, musí používat jednotku KernMon. Jednotka KernMon obsahuje v interface hlavičku procesu KernelMonitor.

```
Procedure KernelMonitor(
  ‘NAM=PRT<space>
   LSB=5000<space>          { konst. – velikost vysílacího bufferu}
   NOD=254<space>           { číslo stanice v komunikační síti }
   DNO=255|’+               { číslo master v komunikační síti }
  ‘NAM=V40<space>           { definice komunikačního kanálu pro řídicí jednotku KitV40 }
                            { pro Kit386EXR se používá kanál NAM=COM COM=1 IRQ=4 }
   BD=4800<space>           { přenosová rychlost }
   BIT=8<space>             { počet bitů }
   PAR=ODD<space>           { druh parity }
   STOP=2<space>            { počet stop bitů }
   LRB=5000’,               { konst. – velikost přijímacího bufferu }
[True/False])               { spuštění aplikace je požadováno od mastera }
```

Pro správný běh aplikace musí být proces KernelMonitor spuštěn hned po inicializaci hlavního procesu a správy času. Parametry Kernel Monitoru musí odpovídat parametrům, zadaným příkazem Communication v menu Option.
Příklad pro řídicí jednotku KitV40:

```
begin
  StartMain(250,254);
  InitInterruptStack(2, 512); StartTimeSlicing(8);
  KernelMonitor('NAM=PRT LSB=5000 NOD=254 DNO=255|'+'NAM=V40 BD=4800 BIT=8
PAR=ODD STOP=2 LRB=5000',false);
{ Spuštění ostatních procesů aplikace }
  ...
```

Příklad pro řídicí jednotku Kit386EXR:

```
begin
  StartMain(250,254);
  InitInterruptStack(2, 512); StartTimeSlicing(8);
  KernelMonitor('NAM=PRT LSB=5000 NOD=254 DNO=255|'+'NAM=COM COM=1 IRQ=4
BD=4800 BIT=8 PAR=ODD STOP=2 LRB=5000',false);
{ Spuštění ostatních procesů aplikace }
  ...
```

7. Parametry programu

Program RTD může být spuštěn s parametry. Příkazová řádka programu má tvar: RTD jméno [parametry]. Jméno určující aplikaci a jednotlivé příkazy musí být odděleny mezerou.

/G - parametr způsobí, že se provede automatické generování aplikace. Aby mohl být parametr akceptován, je potřeba, aby jméno v příkazové řádce určovalo platnou aplikaci s aktuální mapou a aby k této aplikaci existoval desktop soubor .DST.

/L - parametr způsobí, že se provede automatické zavedení vygenerované aplikace. Podmínky pro akceptování parametru jsou stejné jako pro parametr /G. Navíc je však třeba, aby bylo navázáno spojení s BIOS monitorem Slave počítače. Pokud na Slave počítači běží aplikace, provede se automaticky příkaz Reset a komunikace se naváže automaticky.

/E způsobí automatické vyvolání EPROM simulátoru

/R způsobí automatické odstartování aplikace. Podmínky jsou shodné s příkazem /L.

/T způsobí zobrazení okna TerminalOut, do kterého se zobrazují výpisy vzdáleného terminálu realtimového operačního systému ReTOS, viz. manuál ReTOS.

/X po provedení všech příkazů zadaných jako parametr programu se ukončí RTD. Tento příkaz je především používán při práci se simulátorem paměti EPROM.

/M řídí používání myši v prostředí RTD. Parametr **„/M–„** způsobí zakázání používání myši v RTD. Při použití **„/M+„** můžete myš v RTD používat, ale v některých operačních systémech může docházet k padání RTD případně k problémům při komunikaci s řídicím systémem.

/IC řídí používání modulu INITCODE. Parametr **„/IC-„** způsobí zakázání používání modulu INITCODE. RTD se chová jako předchozí verze, které neuměly pracovat s module typu INITCODE. Parametr **„/IC+„** způsobí zapnutí používání modulu INITCODE. Modul INITCODE se v dnešní době automaticky používá pouze pro knihovny Graph a ScGraph. Pokud jsou v programu použity výše uvedené jednotky a není použita ani jedna varianta tohoto parametru, zobrazí se hlášení. Toto hlášení Vás upozorní na použití tohoto parametru.

/W:XXXX nebo /W=XXXX
řídí časy prodlev (TIMEOUT) při přepínání komunikační rychlosti a vysílání dat.

Parametry musí být zadány v odpovídajícím pořadí. Typická příkazová řádka vypadá např. takto:

RTD APPL /G /L /R

pro ladění přes sériovou komunikace,nebo

RTD APPL /G /E /X

ladíme-li pomocí EPROM simulátoru.

Program RTD je možno s výhodou používat jako tool v prostředí Borland Pascalu nebo Borland C++. V příkazové řádce použijeme místo jména aplikace makra $SAVE ALL $EXENAME a výše uvedené parametry. Ladění aplikace pak spočívá v přepínání mezi překladačem a programem RTD s minimálními nároky na obsluhu.

Protože některé parametry jsou obecně použitelné pro více projektů, umí RTD pracovat se systémovou proměnnou RTDCFG. Pomocí této proměnné můžete provést implicitní nastavení RTD pro všechny projekty. Pokud použijete v příkazové řádce různé parametry než v systémové proměnné, jsou tyto parametry přidány k dříve zadaným parametrům. Pokud jsou ale parametry stejné, má vyšší prioritu parametr v příkazové řádce.
Např.

- RTDCFG = "/IC+ /G"
  RTD byl spuštěn s parametry rtd /L /R.
  Po spuštění RTD se provedou příkazy v tomto pořadí: /IC+, /G, /L a /R.

- RTDCFG = "/IC+ /G"
  RTD byl spuštěn s parametry rtd /IC- /L /R.
  Po spuštění RTD se provedou příkazy v tomto pořadí: /IC-, /G, /L a /R. U parametru /IC se použije příkaz zadaný na příkazové řádce a přepíše předchozí nastavení pomocí systémové proměnné.

8. Čtení a zápis souboru

Pro zápis souborů do paměti slouží příkaz ***Write to Kit*** , který naleznete při

volbě ***BIOS*** v hlavním menu. Po výběru ***Write to Kit*** se zobrazí dialogové okno s příkazovou řádkou, do které se píší názvy souborů a adresa v paměťovém prostoru, na kterou budou tyto soubory zapsány, viz. dále příklad zápisu souboru. Při zadávání jednotlivých souborů nelze zadat absolutní adresy zápisu těchto souborů tak, aby se tyto soubory navzájem přepsaly. Při vzniku situace, kdy by hrozilo, že se soubory v paměťovém prostoru začnou přepisovat, není tento příkaz proveden a je zobrazeno hlášení upozorňující na soubor, který by přepsal už zapsaný soubor. Je-li zadána cesta a soubor, který nelze najít, tak příkaz také není proveden a je vyžadována oprava špatně zadaného souboru. Adresa, na kterou je soubor zapsán, je zadávána *lineárně* a *hexa*. (Pozn.: délka zapisovaného bloku dat je odvozena z délky zapisovaného souboru a ta je posléze upravena dle velikosti sektoru paměti FLASH. Tato velikost je rozdílná pro řídicí systém KITV40(256B) a KIT386(512B). U posledně zmíněného se nerozlišuje 8b a 16b režim a velikost vlastního sektoru. Pokud budete zapisovat blok dat o délce nedělitelné velikostí sektoru je zbytek sektoru smazán, tj. přepsán hodnotami od předchozího zápisu.)

Pro čtení paměťového bloku se zápisem do souboru slouží příkaz ***Read from Kit***, který se vyvolá stejně jako příkaz ***Write to Kit***. Po výběru ***Read from Kit*** se zobrazí dialogové okno s příkazovou řádkou, do které se zadávají názvy souborů, lineární adresa a délka paměťového bloku, který bude zapsán do souboru, viz. příklad čtení paměťového bloku. Při zadávání adresy a délky paměťového bloku je třeba mít na paměti, že adresa je zadávána *lineárně* a *hexa a* délka bloku *hexa*.

Příklad zápisu souboru: /***Write to Kit***/

Při prvním spuštění příkazu Write to Kit je na příkazové řádce tato nápověda

```
ABSADDn=FILENAMEn.EXTn ABSADDx=FILENAMEx.EXTx
```

Za FILENAMEn.EXTn (x) se zadá soubor, který bude zapisován do paměti, místo ABSADDn (x). Adresa se zadává jako lineární adresa v hexa tvaru. Těchto definic souborů lze zadat v příkazové řádce více a jejich definice je oddělena mezerou. Při překrývání jednotlivých souborů v paměti nebo neexistenci souboru je ohlášena chyba.

Např. budeme chtít zapsat soubor data1.txt na adresu $8002:0000 a data2.txt na adresu $8100:1020

Na příkazové řádce bude zadáno:

```
Data1.txt=80020 Data2.txt=82020 (pozor na převod adres)
```

Příklad čtení paměťového bloku: /***Read from Kit***/

Při prvním spuštění příkazu Read from Kit je na příkazové řádce tato nápověda:

```
FILENAMEn.EXTn=ABSADDn,LENn FILENAMEx.EXTx=ABSADDx,LENx
```

Za FILENAMEn.EXTn se zadá soubor, do kterého se bude čtený paměťový blok zapisovat, místo ABSADDn (x) a LENn (x) bude v hexa tvaru zadána lineární adresa délka čteného paměťového bloku. Těchto bloků lze najednou číst více (nápověda „n“ a „x“), bloky se mohou překrývat a jejich definice musí být oddělena mezerou.

Např. budeme chtít přečíst paměťový blok z adresy $8002:0000 délce 16 bytů do souboru data1.txt a z adresy $8102:1000 o délce 256 bytů do souboru data2.txt.

Na příkazové řádce bude napsáno:

```
Data1.txt=80020,10 Data2.txt=82020,100 (pozor na převod adres)
```

Pokud soubory Data1.txt a Data2.txt už existují, pak musíte povolit jejich přepsání.

9. Modifikace konfigurační tabulky

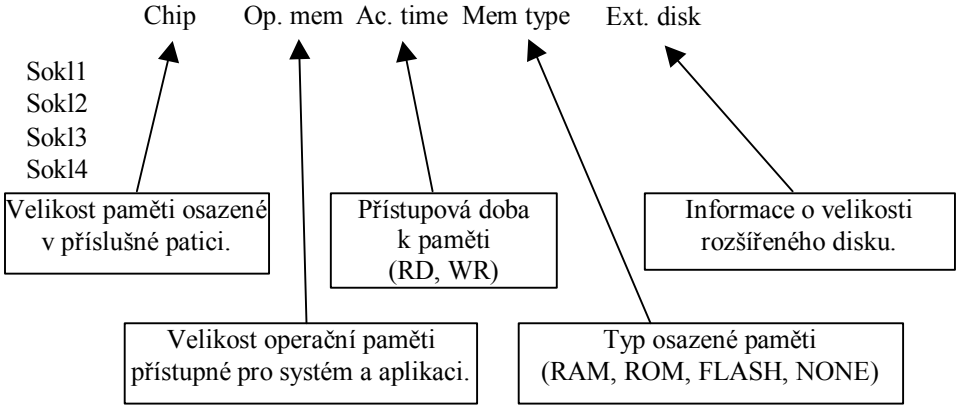
Konfigurační tabulka BIOSu 386EXR slouží ke změně rozložení paměťového prostoru určeného pro aplikační program a pro archiv dat výrobního procesu. V této tabulce lze kromě změny rozložení paměťového prostoru změnit přístupovou dobu k jednotlivým pamětem a nastavení konfiguračního slova. (Bližší popis konfigurační tabulky naleznete v manuálu BIOS386EXR)

Menu určené pro změnu konfigurační tabulky lze vyvolat pouze je-li navázána komunikace s procesorem 386EX (viz. Získání informací o BIOSu). Tohoto menu vyvoláte výběrem *BIOS* z hlavního menu a poté *Modify cfg table*.

Výsledkem tohoto příkazu je zobrazení tabulky, která je rozdělena do několika částí. První část, umístěná v horní části tabulky, slouží pro nastavení parametrů jednotlivých pamětí osazených na řídicí jednotce Kit386EXR. Druhá část, umístěná uprostřed tabulky, slouží k nastavení rezervace paměti pro grafickou kartu. Třetí část, umístěná v spodní části tabulky, slouží pro nastavení konfigurační slova. Změnou tohoto slova se ovlivňuje chování BIOSu a nastavení některých periferních obvodů procesoru 386EX. Poslední část tabulky jsou tlačítka, který provedou změnu konfigurační tabulky v BIOS, tlačítko OK nebo všechny změny budou stornovány, tlačítko Cancel nebo provedení přepočtu a kontroly hodnot, tlačítko Update&check values. Dále bude popsána editace a stručný popis významu parametrů konfigurační tabulky. Vlastní editace a přechod na další parametr v tabulce se provádí použitím myši, tabulátoru nebo ALT + zvýrazněné písmeno, číslice, tzn. stejně jako se obsluhuje Borland Pascal. (Pozn.: Všechny parametry udávající velikosti paměti jsou zadávány a zobrazovány v kB.)

Upozornění: U zadávání velikostí pamětí nebo oblastí je možné zadávat obecně pouze tyto hodnoty: 16, 32, 64, 128, 256, 512 a 1024. U některé parametrů, ale lze zadat pouze vybrané hodnoty. V těchto případech se lze řídit chybovými hlášeními RTD nebo pro jednotlivé parametry vyhledat výčet správných hodnot v manuálu BIOS386EXR.

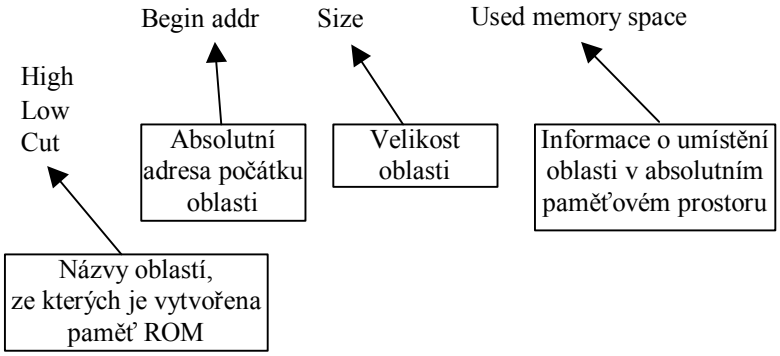
První část sloužící pro nastavení parametrů paměti je rozdělena do čtyř řádků nazvaných Sokl1, Sokl2, Sokl3 a Sokl4. Patice jsou číslovány od konektoru RESET. Zkrácené názvy patic v konfigurační tabulce odpovídají pojmenování pamětí v manuálu Kit386EXR následovně: Sokl1 – ROM Low, Sokl2 – RAM Low, Sokl3 – ROM High a Sokl4 – RAM High. Vysvětlení významu parametrů pamětí je na následujícím obrázku, který zobrazuje pouze první část konfigurační tabulky.



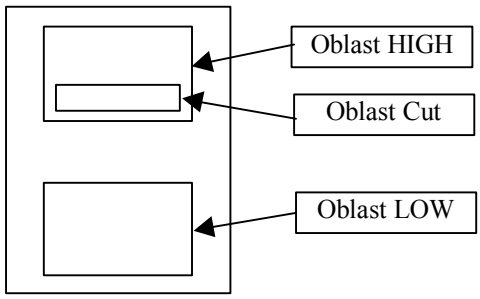
Jediným parametrem, který nelze editovat jsou hodnoty ve sloupci Ext. disk, který pouze slouží jako informace o velikosti paměti vyhrazené pro rozšířené disky. (Pozn.: Bližší popis naleznete v manuálu DiskIO) Velikost disku je dána rozdílem velikosti osazené paměti a velikostí paměti vyhrazené pro aplikaci a systém, tj. operační paměť.

Druhá část slouží k vyhrazení paměťového prostoru pro paměť a BIOSy přídavných karet na sběrnici PC104. Standardní nastavení odpovídá rezervaci paměti pro VideoRAM a pro Video BIOS grafických karet. Při zadávání paměťového prostoru oblastí paměti ROM se vždy zadává začátek oblasti a její velikost. Tyto

oblasti se pak skládají podle obrázku následujícího za zobrazením druhé části konfigurační tabulky.



Informace uložená ve sloupci Used memory space je needitovatelná a podává pouze informaci o tom, kde jsou umístěné oblasti popisující paměť ROM v absolutním prostoru. Vytvoření paměťového prostoru pro paměť ROM lze symbolicky popsat takto, viz. následující obrázek.



Paměť ROM

Paměť ROM je složena ze dvou částí, které sčítají (*Low a High*). Prostor mezi těmito dvěmi částmi je rezervován pro paměť karet na sběrnici PC104. Pokud tato paměť nemá být rezervována pak oblast High bude začínat od absolutní adresy $80000 s velikostí paměti 512kB, oblast Low také od adresy $80000, ale její velikost bude 0. *Oblast Cut* pak slouží pro vytvoření okénka v oblasti High a je určena pro BIOSy karet na sběrnici PC104.

Poslední částí konfigurační tabulky je část nazvaná Configuration. V této části se nastavuje konfigurační slovo, které je přímo ovlivňuje nastavení určitých periferií procesoru 386EX a chování BIOSu. Při nastavování periférií procesoru 386EX, ale nesmíte nikdy zapomenout na správné nastavení propojek na řídicí desce. Pokud na to zapomenete může dojít k poškození řídicí desky. V těchto případech je u nastavení položky konfiguračního slova zvlášť upozorněno na změnu nastavení propojek. Toto upozornění by se proto v žádném případě nemělo brát na lehkou váhu.

V konfigurační slově lze nastavovat tyto položky pomocí zatrhávacích polí (Check box). Zaškrnuté políčko vždy znamená první volbu z popisu nebo znamená výběr volby. (např. Zaškrtnuté políčko u SSIO/COM2 znamená výběr SSIO signálů na COM2 nebo zaškrtnuté políčko znamená u VideoWR aktivaci zápisů do VideoRAM).

V RTD můžete nastavovat políčka s těmito vlastnostmi:

- SSIO/COM2 Toto políčko určuje zda na konektoru COM2 bude asynchronní komunikace

nebo synchronní komunikace. Proto je při změně potřeba provést změnu nastavení propojek u komunikací.

- SMM/INT3 Toto políčko určuje zda jeden pin procesoru bude výstupem časovače pro přechod do SMM nebo zda bude vstupem přerušení INT3. Při změně tohoto políčka je nutné provést změnu nastavení propojek u SMM/INT3.
- 25/33 Políčko určuje inicializaci časovačů podle frekvence procesoru. Následkem špatného nastavení tohoto políčka jsou špatně nastavené dělící poměry v procesoru a tím způsobené nepřesnost při obsluze INT8.
- DMA1/COM2 Toto políčko určuje zda na konektoru COM2 bude asynchronní komunikace nebo některé signály DMA1. Proto je při změně potřeba provést změnu nastavení propojek u komunikací.
- DMA2/COM2 Toto políčko určuje zda na konektoru COM2 bude asynchronní komunikace nebo některé signály DMA2. Proto je při změně potřeba provést změnu nastavení propojek u komunikací.
- VideoWR Tato položka aktivuje zápisy video služeb BIOSu. Tato položka, ale v žádném případě neovlivní chování video služeb pokud je v systému grafická karta s vlastním BIOSem. Tato volba je výhodná v systému, který byl oživován s pomocí VGA karty a 512kB pamětí FLASH v prostoru ROM. Po oživení byla zrušena rezervace paměti grafické karty a vlastní karta byla ze systému odstraněna.
  Odstraněním BIOSu grafické karty přechází veškeré služby s video pamětí na služby BIOS řídicí jednotky.

  Nezřídka se stane, že zapomenete odstranit některé výpisy na obrazovku a ty by při zápisu do FLASH paměti způsobovaly restart systému. Proto se pomocí této volby tyto zápisy zruší a systém bezchybně funguje. Po zasunutí grafické karty a aktivování rezervace paměti, bez změny tohoto políčka, tj. políčko je nezaškrtnuté, lze na monitoru sledovat kontrolní výpisy při běhu aplikace.

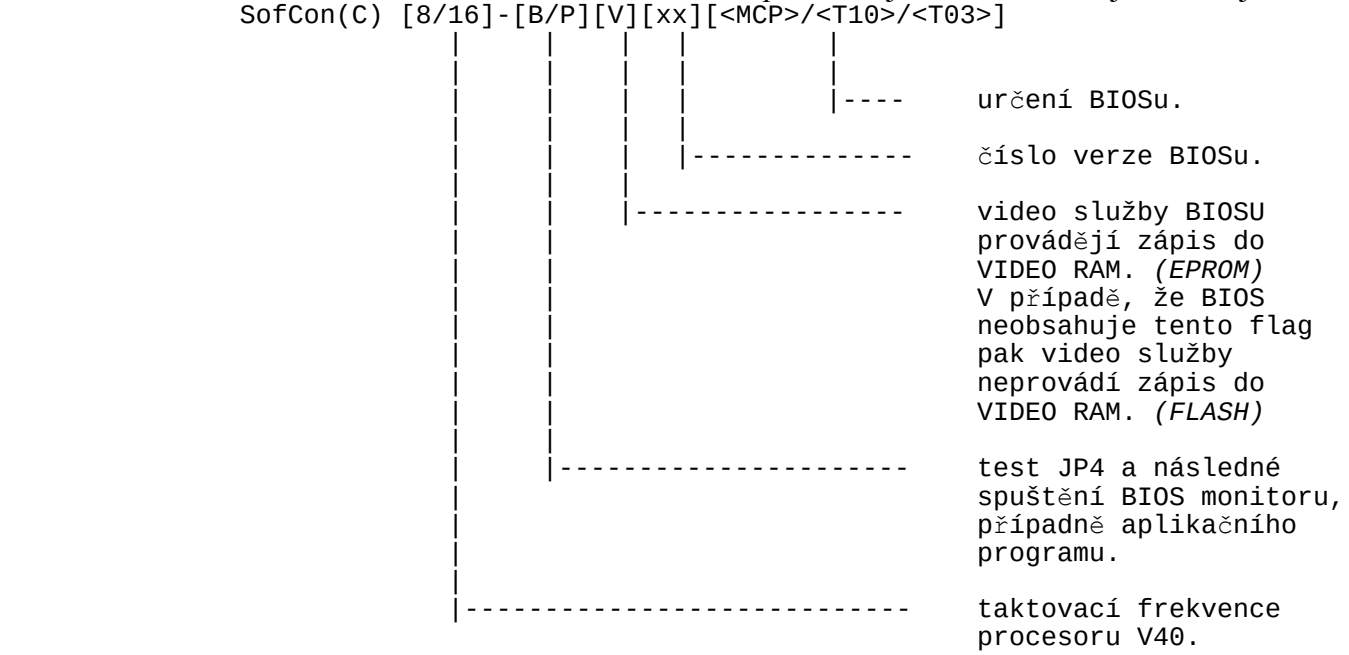
  Pozn.: Pro některé starší grafické toto políčko musí být, ale zaškrtnuto, např. Hercules, některé EGA karty.

10. Získání informací o BIOSu

V případě, že zapomenete verzi BIOSu, který je nahrán v paměti EPROM/FLASH a nemáte k dispozici pracoviště s grafickou kartou, pak informace o BIOS získáte pomocí příkazu ***Get BIOS info***. Tento povel spustíte výběrem ***BIOS*** z hlavního menu a poté ***Get BIOS info***.

Výsledkem této akce je řetězec popisující vlastnosti BIOSu a informace o tom, který typ paměti FLASH identifikoval BIOS. V případě, že BIOS nedetekuje žádnou paměť typu FLASH zobrazí se pouze EPROM xxxx.

Struktura identifikačního řetězce pro řídicí jednotku KitV40 je následující:



Např.:

Pro KitV40 s frekvencí 16MHz pro aplikaci TERM10 bez aktivovaných video služeb a verzí BIOS 1.7 pro paměti FLASH ....

SofCon (c) 16-B17<T10>

Struktura identifikačního řetězce pro řídicí jednotku Kit386EXR je následující:

```
386EXR/[a]-[b]-[c]:[d]-[e][f]<KIT>-[g]
        |   |   |    |   |   |        |
        |   |   |    |   |   |        |-----   nastavení konfiguračního
        |   |   |    |   |   |                registru BIOS386EXR
        |   |   |    |   |   |                [O/M]
        |   |   |    |   |   |
        |   |   |    |   |   |-------------   číslo verze BIOSu
        |   |   |    |   |
        |   |   |    |   |-----------------   test APL/BM a následné
        |   |   |    |                        spuštění BIOS monitoru,
        |   |   |    |                        případně aplikačního
        |   |   |    |                        programu (JP33)
        |   |   |    |                        [B/P]
        |   |   |    |
        |   |   |    |--------------------    přístup k paměti RAM
        |   |   |                             [8/16]
        |   |   |
        |   |   |--------------------------    přístup k paměti ROM
        |   |                                 [8/16]
        |   |
        |   |----------------------------    test DEF/CFG a následné
        |                                     nastavení registrů
        |                                     procesoru 386EX (JP30)
        |                                     [Def/Cfg]
        |
        |--------------------------------    taktovací frekvence
                                              procesoru 386EX
                                              [25/33]
```

Upozornění: Pokud Kit386EXR nabootuje v default módu, tj. identifikační řetězec obsahuje *Def*, pak se po restartu nikdy nespustí aplikace, ale BIOS Monitor. Tento mód slouží pouze k novému nastavení konfigurační tabulky.

Příklad identifikačního řetězce:
Pro Kit386EXR s frekvencí 25MHz s bootováním z konfigurační tabulky s nastaveným 8 bitovým přístupem do paměti ROM, 16 bitovým přístupem do paměti RAM, bez modifikovaného konfiguračního registru a verzí BIOS 1.3 pro paměti FLASH ....

386EXR/25-Cfg-8:16-B13<KIT>-O

Bližší popis možností BIOSů najdete v souboru Popis.txt v adresáři obsahující BIOSy.

11. Zkrácená verze KernelMonitoru

KernelMonitor je možno používat ve dvou verzích: úplné a redukované. Redukovaná verze je výhodná tehdy, jestliže v aplikaci nejsou použity schránky. Procedury pro obsluhu schránek se tak nepřipojí a celá aplikace se zkrátí. Nevýhodou je, že při ladění nelze používat breakpointy. Sestavení redukované verze dosáhneme tak, že místo procedury KernelMonitor zavoláme v aplikaci proceduru SmallKernelMonitor. Parametry obou procedur jsou totožné.

12. Seznam varovných a chybových hlášení

Communication error
Byla přerušena komunikace mezi Master a Slave počítačem, nebo nelze navázat spojení.

Dump length is null or greater than 1KB
Chybně zadaná délka výpisu . Hodnota musí být zadána mezi 0 a 1kB.

Invalid address
Chybný formát adresy. Adresa musí být ve tvaru segment:offset.

Invalid variable format
Chybný tvar formátu proměnné nebo přístupu k proměnné.

Invalid value format
Zadaná hodnota neodpovídá formátu proměnné.

Not enough memory
V Master počítači nezbývá žádná volná paměť.

Ambiguity process name
V daném kontextu je povoleno pouze jednoznačné jméno procesu (bez znaku '*').

Could not open file
Nepodařilo se otevřít uvedený soubor (nebo do něj zapsat). Zkontrolujte nastavení atributu, např. ReadOnly.

Unknown breakpoint occured at address
KernelMonitor hlásí, že nastal breakpoint, o kterém Master neví. Nejlépe je zrušit všechny breakpointy příkazem DeleteAll v menu Breakpoints.

Could not open desktop file
Nepodařilo se otevřít soubor jméno_aplikace.DST (nebo do něj zapsat). Zkontrolujte nastavení atributu, např. ReadOnly.

Breakpoint encountered in process
Uvedený proces dospěl na breakpoint.

Unknown mailbox
KernelMonitor nezná zadanou schránku. Jméno schránky je řetěz uvedený při inicializaci schránky.

Breakpoint setting failed
Nepodařilo se nastavit breakpoint.

Unknown command for KernelMonitor
KernelMonitor nezná příkaz. Pravděpodobně je v aplikaci použit redukovaný Kernel monitor.

EPROM simulator failed. DOS error
Program EPROM simulátor buď nebyl nalezen nebo skončil s chybou.

Invalid breakpoint
KernelMonitor nezná zadaný breakpoint. Nejlépe je zrušit všechny breakpointy příkazem DeleteAll v menu Breakpoints.

Invalid value
Nedovolená hodnota.

Process not stopped at breakpoint
Zadaný proces nestojí na breakpointu.

Checksum error
Chyba checksum. Je potřeba znovu zavést aplikaci.

DS is not valid
Aplikace není ještě umístěna, takže nelze při zadávání adresy používat symbol DS.

Address not found
Zadaná adresa nebyla v mapě nalezena.

Too many breakpoints
Může být zadáno maximálně osm breakpointů.

Could not open config file
Nepodařilo se otevřít soubor RTD.CFG (nebo do něj zapsat). Zkontrolujte nastavení atributu, např. ReadOnly.

Checksum OK
Správný checksum.

Memory module overflow
Do modulu je umístěno více unit, než se jich tam vejde. Přebytečné unity je potřeba umístit do jiného modulu.

Some unit(s) not placed
Některé unity nebyly umístěny. Je potřeba umístit všechny unity.

Invalid line number
Uvedená řádka nebyla nalezena v mapě aplikace. Je potřeba vytvořit mapu typu detailed.

Address not found, Invalid segment
Segment uvedené adresy nebyl v mapě nalezen. Pravděpodobně chyba při zadávání adresy.

Address found, Outside of the code segment
Adresa byla v mapě nalezena, ale je mimo kódový segment. Pravděpodobně chyba při zadávání adresy.

Address not found, Invalid offset
Adresa nebyla v mapě nalezena. Nesouhlasí offset adresy. Pravděpodobně chyba při zadávání adresy.

Address found
Adresa nalezena.

Invalid segment, Probably: MAP file out of date
Neplatný segment. Pravděpodobně je neplatná mapa a je třeba ji znovu vytvořit.

Unit not found
Uvedený unit nebyl nalezen. Pravděpodobně je třeba aplikaci znovu sestavit.

Invalid map file
Nesouhlasí formát mapy. Je potřeba znovu vytvořit mapu.

Memory modules overlap
Adresy paměťových modulů se překrývají. Je potřeba změnit začáteční adresu některého paměťového modulu.

Remove empty memory module
Jeden z vytvořených modulů nemá přiřazenu žádnou programovou jednotku, buď do tohoto modulu přidejte alespoň jednu programovou jednotku nebo ho smažte.

Map file out of date. Please, recompile the application
Nesouhlasí datum vytvoření mapy. Aplikaci je třeba znovu přeložit a vytvořit novou mapu.

Invalid EXE format
Nesouhlasí formát souboru .EXE. Aplikaci je třeba znovu přeložit.

Memory module size exceed 128kB!
Délka paměťových modulů může být maximálně 128KB.

Communication error - TIMEOUT
Během zavádění aplikace do procesoru, tzn. během komunikace RTD s BIOS monitorem nebo Kernel monitorem, došlo k přerušení komunikace. Toto přerušení mohlo být způsobeno buď přerušením sériové linky, rušením nebo přerušením komunikace v MASTER případně SLAVE počítači.

Attempt to change value in ROM
Pokus nastavit hodnotu proměnné uložené v ROM.

Attempt to set breakpoint in ROM
Pokus nastavit breakpoint v paměti ROM.

Bin file out of date! Please, generate the application.
V aplikaci byla provedena změna, je potřeba aplikaci znovu vygenerovat, viz. Umístění aplikace.

Unknown communication paket.
RTD přijal neznámý paket. Chyba může vzniknout při nastavení špatného portu nebo při spuštění aplikace, která komunikuje s RTD.

Invalid CFG format or check your date. Please set options.
V souboru RTD.CFG je chyba. Tato chyba může nastat při poškozeném souboru nebo při přechodu na novější verzi RTD. Při vzniku této chyby RTD zapomene všechna nastavení a parametry příkazů, které se musí znovu zadat – např. komunikační rychlost, parametry příkazu Eprom simulátor atd. V případě, že si jste jisti, že používáte verzi RTD vyšší jak 1.63 pak zkontrolujte systémový čas, podle kterého se rozlišují verze 1.62 a verze nižší.

Invalid DST format or check your date. Please place units in memory modules.
V souboru .DST je chyba. Tato chyba může nastat při poškozeném souboru nebo při přechodu na novější verzi RTD. Při vzniku této chyby se musí provést nové umístění aplikace, při tomto umístění aplikace Vám může být nápomocen soubor .MEM. V případě, že si jste jisti, že používáte verzi RTD vyšší jak 1.63 pak zkontrolujte systémový čas, podle kterého se rozlišují verze 1.62 a verze nižší.

BIOS not support RTD identification.
BIOS, s kterým RTD komunikuje nepodporuje identifikaci. V tomto případě Vám doporučujeme se s námi kontaktovat a požádat o novou verzi BIOS, případně o radu jak tuto chybu odstranit.

Attempt to overwrite BIOS.
Zapisovaná data jsou příliš dlouhá. Při jejich zápisu by se poškodil BIOS, proto zápis nebyl proveden. Data je třeba nahrát znovu s jinou počáteční adresou.

Attempt to overwrite BIOS.
Data (aplikace) byla zapisována do systémové oblasti BIOSu, viz. manuál. Změňte počáteční adresu prvního modulu.

Module overleps BIOS.
Poslední modul je příliš dlouhý, Při jeho zápisu by se poškodil BIOS, proto zápis nebyl proveden. Modul je třeba nahrát znovu s jinou počáteční adresou.

Communication port not initialized.
Zvolený komunikační port se nepodařilo inicializovat. Port buď fyzicky neexistuje a nebo je obsazen operačním systémem Win95, atd.

Invalid segment address. Address must be aligned by $20.
Počáteční adresa ROM modulu není dělitelná $20.

Memory modules overlap.
Upozornění, že při generování modulů se může stát, že některé moduly se budou překrývat, což může způsobit špatně zjistitelnou chybu při běhu aplikace.

Module name already used.
Existují dva moduly se stejným jménem. Pro odstranění této chyby se jméno jednoho z modulů musí změnit.

Communication error – unable switch speed.
Při komunikaci s Slave počítačem došlo k chybě, která může být způsobena operačním systémem, případně rychlostí počítače. V případě, že program nebude fungovat ani v DOS6 a vyšším, Vám doporučujeme se s námi kontaktovat., aby programátoři mohli tuto chybu odstranit.

Communication error: Maybe write module into EPROM extended BIOS or old FLASH.
Při zápisu dat (aplikace) do paměti došlo k chybě, která je způsobena buď , že data (aplikaci) zapisujete do paměti EPROM, nebo do BIOSu připojených desek, případně Vaše paměť FLASH je poškozená. Alespoň do jednoho sektoru nelze provést zápis. Zkontrolujte Vaši sestavu, případně vyměňte paměť FLASH za novou a proveďte nový zápis.

Attempt to overwrite system area.
Data (aplikace) byla zapisována do systémové oblasti BIOSu, viz. manuál. Změňte počáteční adresu prvního modulu.

Module length with checksum exceed 128KB.
Délka žádného modulu nesmí přesáhnout 128kB. Opravte délku modulu, který přesahuje 128kB.

Invalid chip size.
S touto velikostí osazené paměti neumí BIOS řídicí jednotky Kit386EXR pracovat.

Invalid operation memory size.
S touto velikostí operační paměti neumí BIOS řídicí jednotky Kit386EXR pracovat.

Invalid memory type.
Na této pozici nemůže být osazena tato paměť.

Invalid delay time.
Toto prodloužení přístupové doby k paměti nelze v BIOSu řídicí jednotky Kit386EXR nastavit.

Invalid chip select address.
Tato počáteční adresa oblasti nelze v BIOSu řídicí jednotky Kit386EXR nastavit.

Invalid chip select size.
Tato velikost oblasti nelze v BIOSu řídicí jednotky Kit386EXR nastavit.

Unable to store configuration table.
Konfigurační tabulka nebyla v BIOSu změněna. Možnými důvody jsou špatné nastavení propojek (EPROM/FLASH) nebo BIOS je uložen v paměti EPROM.

Invalid chip size or memory type.
U paměti je zadán typ ale zadaná velikost paměti je nula nebo není zadán typ a je zadána velikost. Pokud paměť není osazena pak se musí zadat typ None a velikost 0.

Forbidden coalescence of units with DATA and CODE segment.
V paměťových modulech se nedovoleně kombinují datové (DATA, STACK, HEAP) a kódové unity (vše zbylé).
Pozn. ScGraph a Graph jsou unity, které musí být spouštěny v paměti RAM, proto při jejich umístění dochází ke změně typu paměťového modulu na IC, viz. Umístění aplikace

Only one memory module with initialized code is permitted.
V RTD lze vytvořit pouze jeden paměťový modul typu IC.

Memory module size with initialized code exceed 64kB.
V RTD lze vytvořit pouze jeden paměťový modul typu IC.

Forbidden coalescence of memory modules.
V paměťových modulech se nedovoleně kombinují inicializované kódové unity s unitou –INIT CODE--.
Pozn. ScGraph a Graph jsou unity, které musí být spouštěny v paměti RAM, proto při jejich umístění dochází ke změně typu paměťového modulu na IC, viz. Umístění aplikace

Adding checksum to module occurs exceeding its length.
Při pokusu přidat CHECKSUM k jednotkám v modulu bylo zjištěno, že v modulu nezůstal žádný volný prostor. Chyba se odstraní buď zvětšením velikosti modulu nebo přesunem vhodné jednotky do jiného modulu.

Unknown error.
RTD nezná přijatou zprávu. Pravděpodobně SLAVE počítač posílá příkazy, které RTD nezná.

Internal fail.
Vnitřní chyba programu RTD. Vznikne soubor RTD.LOG, který je třeba přiložit při reklamaci u distributora.