Začínáme s BP7 a o.s. Retos

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 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		We		První vydání
1.10		Tu	19.05.2003	Úprava dokumentu dle ISO9000

1.2. Účel dokumentu

Tento dokument slouží jako příručka pro základní seznámení s balíkem knihoven pro tvorbu aplikací ve vývojovém prostředí Borland Pascal / o.s.RETOS.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Tento dokument předpokládá znalost programování v prostředí Borland Pascal. Čtenář by se měl seznámit také s popisem operačního systému RETOS.

2. Termíny a definice

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

3. Úvod

Tento článek se snaží objasnit problémy, na které narazí programátor při tvorbě prvního programu pro KitV40 nebo Kit386 v šestnáctibitovém prostředí ***SofCon***. Podává první informace o obsahu balíku dodávaných knihoven, o způsobech ladění, o možnostech jak dostat program do procesoru, o způsobech programování.

Vývojové prostředí Borland Pascal / o.s.RETOS je určené pro tvorbu a odladění složitých i jednodušších aplikačních programů, určených pro řídicí jednotky KITV40, KIT386EX s terminály TERM01, TERM05, TERM06, pro průmyslové terminály TERM10 nebo pro kompaktní řídicí systémy KOMPAKT, pokud úloha vyžaduje poměrně rychlé časové odezvy. Vlastní programy píše uživatel v programovacím jazyce Borland Pascal, V7.0 (nutná znalost dynamických proměnných a alespoň základní znalost objektového programování), přičemž využívá bohaté nabídky procedur a funkcí, obsažených v knihovnách ***SofCon***.

Firma ***SofCon® s.r.o.*** nabízí řadu knihovních modulů, které slouží pro tvorbu aplikace a pro ladění programu. Knihovní jednotky jsou určeny pro překladač jazyka Borland Pascal. Jednotky se standardně dodávají ve formě .TPU. Uživatel dostává k dispozici soubory s "interface" sekcemi, manuály a příklady použití jednotlivých knihoven. Některé vybrané knihovny se dodávají ve zdrojové formě a po konzultaci může uživatel získat i některé další zdrojové formy knihoven.

Knihovny ***SofCon*** se dodávají ve versi pro prostředí DOS a MCP. Prostředí DOS je standardní prostředí počítače PC, kde můžeme ladit program se simulátory hardwarového prostředí, nebo s expanzí sběrnic PCKit a na ni připojenými periferními deskami. Verse MCP je určena pro prostředí procesoru KITV40, nebo KIT386. Pouhým slinkováním programu s jinou versí stejných knihoven a změnou adres periferií přejdeme od PC do prostředí KIT.

Programátor musí zvládnout nejen vlastní program, ale také překladač jazyka Pascal, připojit správnou versi knihoven, vygenerovat Rom moduly pro procesor KIT a zavést je do něj. Všechny tyto akce jsou citlivé na správné nastavení podmínek překladu a cest k adresářům. Na demonstračním CD ***SofCon s.r.o.*** nacházejí zdrojové soubory demonstračních programů, které jsou uloženy i s prostředím překladače a podmínkami překladu. Odtud se dá inspirovat a podmínky překladu, include soubory a konfigurační soubory kopírovat.

Programové vybavení ***SofCon*** pro prostředí KIT se sestává ze základní knihovny Borland Pascalu Turbo.tpl, která se v MCP versi obrací přímo na Bios procesoru KITV40 nebo KIT386 a balíku knihovních modulů. Knihovny obsahují ovladače ke všem složitějším periferním deskám, balík pro ovládání terminálů, balík komunikačních programů, operační systém reálného času RETOS, knihovny pro práci se systémovým časovačem Int08 a knihovnu pro práci s flash pamětmi, regulační knihovny a další užitečné programy. Většina knihoven je napsaná objektově, ale naplno jsou objektové vlastnosti využity jen u knihoven pro terminály a u komunikačních knihoven. Aplikace s terminálovými knihovnami jsou založené v prázdných startovních aplikacích (BeginT10, BeginT01 ...), takže uživatel pouze doplní svůj kód do rozběhnuté aplikace s terminálem. Komunikační knihovny mají v manuálu dobré demonstrační příklady, které se dají použít aniž by bylo nutné hluboce pátrat po významu dědičnosti.

Většinu poznatků uváděných v tomto dokumentu lze přečíst podrobněji v softwarové dokumentaci k "Prostředí Borland Pascal / RETOS". Tento článek slouží pro první seznámení s programováním v šestnáctibitovém prostředí počítačů KitV40 nebo Kit386.

4. Adresáře

Poněvadž verse programu do prostředí DOS (prostředí počítače PC) a do prostředí MCP (prostředí procesoru Kit) potřebují různé knihovny, různé základní knihovny Borland Pascalu turbo.tpl a různé nastavení podmínek překladu, jeví se jako nejjednodušší překládat každou z versí do jiného adresáře a pracovat se společnými zdrojovými programy. Podmínky překladu se umístí do zvláštního include souboru. Tyto soubory jsou dva, jeden pro DOS a druhý pro versi MCP (verse pro Rom procesoru Kit). V demonstračních programech na CD disku v adresáři SW\SWBP je toto uspořádání striktně dodržované. Pokud celý adresář SW včetně podadresářů z CD disku překopírujeme na pevný disk svého počítače, bude možno všechny příklady rovnou překládat a budou fungovat. V demostračních programech se v adresáři se zdrojovými programy překládá a generuje verse DOS do prostředí PC. V podřízeném adresáři ROM se tvoří verse MCP.

Také knihovny jsou rozděleny do adresářů DOS7 a MCP7 pro versi DOS a MCP.

5. Pascal 7

Překladač Borland Pascal 7 není součástí CD disku ***SofCon***, poněvadž není volně šiřitelný. Uživatel našeho prostředí si jej musí koupit, například i ve firmě ***SofCon spol. s r.o.***

Překladač Borland Pascal 7 potřebuje nastavit do souboru Autoexec příkaz Path s cestou k souboru BP.EXE. Většinou je umístěn v adresáři \BP\BIN. Pascal voláme z aktivního adresáře, kde vytváříme aplikaci. Z tohoto adresáře Pascal natahuje své konfigurační soubory BP.TP, BP.DSK a základní knihovnu TURBO.TPL. Vzhledem k tomu, že knihovna TURBO.TPL je v prostředí ***SofCon*** modifikovaná a je rozdílná pro versi DOS a MCP, je třeba ji z adresáře s TPU soubory knihoven přehrát do správných adresářů aplikace. Pokud toto neučiníme, natáhne se standardní knihovna z adresáře BIN překladače a aplikace nebude fungovat. V adresářích CD disku jsou uložené správně nastavené konfigurační soubory pro BP7 i soubor TURBO.TPL, takže překladač bude rovnou správně fungovat. Konfigurační soubory jsou volně kopírovatelné mezi adresáři do nové aplikace. Po nakopírování je nutné pouze změnit cestu do aktuálního adresáře.

5.1. Nastavení překladače

Ve spuštěném překladači BP7 je třeba nastavit :

a. V menu File nastavit cestu k aktivnímu adresáři.

b. V menu Options nastavit podmínky překladu. Doporučujeme zapnout signalizaci všech RuntimeErrors , celého Debugging a generaci kódu

pro procesor 286. Do podmínek překladu, pokud se překládají i některé knihovny, je třeba napsat Ver7, DOS, nebo Ver7, MCP.

c. V menu Options nastavit cesty ke zdrojovým souborům, knihovnám a do adresáře BIN pascalu. Doporučujeme používat tečkové konvence. Aktuální adresář je .\; nadřízený adresář je ..\;

d. V menu Options nastavit pro MCP versi detailní MapFile od Linkeru. Z tohoto souboru se berou informace, jak z EXE souboru vytvořit ROM moduly pro procesor Kit.

e. V menu Options.Tools nastavit pro MCP versi cestu pro volání programu RetosDebugger a do Command line zadat : $EXENAME /G.

f. Pro rozsáhlejší projekty je vhodné nastavit Primary File.

5.2. Překlad

Borland Pascal umí dva stupně překladu. První stupeň je Compile, kde se překládá pouze Primary File nebo aktivní soubor a soubory s uloženými změnami. Od ostatních zdrojových souborů se použijí již existující TPU soubory. Při rozsáhlejších změnách je vhodné zvolit druhý stupeň překladu Build. Při něm se znovu vytvoří všechny TPU od dosažitelných zdrojových programů. I tento stupeň však "nahlíží" do existujících TPU souborů a například, po výměně knihoven za novější versi, občas selže a hlásí "unit version mismatch". Potom nepomůže nic jiného, než všechny TPU soubory v aplikaci smazat a vše znovu přeložit.

Pro řízení překladu je výhodné používat podmíněný překlad. Tím můžeme měnit adresy pro DOS a MCP versi, použít nebo naopak vyřadit simulátory hardware, nastavit jinou konfiguraci hardware atd. Všechny podmínky překladu soustředíme v include souboru, který se vkládá do každého zdrojového souboru. Include soubor musí být různý pro versi DOS a MCP !

6. Jak dostat program do procesoru Kit

Překladač BP7 vytváří EXE soubory s programem a MAP soubory s popisem přiřazení adres. V MCP versi po úspěšném překladu zavoláme z prostředí překladače program RetosDebugger. Nastavení Tools je popsáno výše. RetosDebugger z EXE souboru vytvoří moduly pro ROM a RWM do paměti procesoru Kit. Jedná se o standardní ROM moduly Biosu PC, které Bios v paměti vyhledává a spouští. Program je tvořen jedním RWM modulem a minimálně jedním modulem ROM.

RWM modul obsahuje code segment, stack a heap. V dolních adresách modulu je data segment. Nad data segmentem je volná paměť pro heap a od horních adres segmentu směrem dolů roste stack. Stack a heap rostou ve volné paměti proti sobě. Délka RWM modulu není omezená.

Jeden ROM modul je maximálně dlouhý 128kB. Moduly ROM obsahují code segment programu. Unity mohou být do ROM modulů poskládány celkem libovolně s výjimkou Loaderu, který musí být umístěn v adresové oblasti C0000h až FE000h pro

KitV40, nebo FD000h pro Kit386. Loader totiž obsahuje hlavičku ROM modulu. Bios ROM moduly vyhledává pouze v této adresové oblasti počínaje adresou C0000h.

První nalezený neporušený ROM modul Bios spustí. Loader umístěný mimo tuto oblast, například pro ladění v oblasti rwm, Bios nenajde a musíme ho spustit příkazem Run RetosDebuggeru.

V oblasti od A0000h do C7FFFh je umístěna Videoram a BIOS karty VGA, kterou často používáme pro ladění, proto do této oblasti standardně nemůžeme umístit naši aplikaci.

V případě potřeby využití výše uvedené oblasti pro aplikaci je třeba nepoužívat VGA kartu pro ladění, navíc je však třeba použít jiný obsah GAL paměti na procesorové desce - příslušnou podporu poskytne firma SofCon.

V oblasti FE000h až FFFFFh je umístěn BIOS na desce KITV0, v oblasti FD000h až FFFFFh je umístěn BIOS na desce KIT386EX , do těchto oblastí též nelze umístit aplikaci (a ani to Retos Debugger nepovolí).

Do procesoru lze moduly zavést třemi způsoby:

a. RetosDebugger umí komunikovat s Biosem procesoru Kit a ve spolupráci s ním umí automaticky zavádět moduly do paměti Flash i RWM. Pomocí propojek na procesoru lze zrušit běžící aplikaci a nastartovat pouze Bios. Po navázání spojení s Biosem vyměníme kód aplikačního programu.

b. Všechny Rom moduly i Bios vypálit v programátoru pamětí na správných adresách do paměti Eprom nebo Flash. Tuto paměť zasunout do desky procesoru.

c. Všechny Rom moduly a Bios nahrát do simulátoru Eprom, který nahrazuje v procesoru paměť Eprom. Přímo z RetosDebuggeru je možné volat obslužný program simulátoru. Pro simulátor SimEprom 02 firmy Elnec existuje v ***SofConu*** další programová podpora.

Program RetosDebugger má samostatný manuál, ve kterém je způsob tvorby ROM modulů a jejich zavádění popsán.

6.1. Počátky a délky modulů procesoru KitV40

Následující tabulky popisují nejběžnější rozložení RWM a ROM modulů pro procesor KitV40 s různými velikostmi paměti ROM a RWM. Řádky tabulky s RWM (128k), RWM (512k) odpovídají procesoru s osazenou pamětí RWM 128kByte, nebo 512kByte. ROM1, ROM2 a ROM3 jsou řádky s počátky a délkami jednotlivých ROM modulů, které mohou být pro velikost paměti ROM vygenerovány. Například pro 64kByte Eprom stačí vygenerovat jediný ROM modul s počátkem F000h (Údaj odpovídá code segmentu.) a délkou E000h (Délka je lineární.). Větší paměti se už nedají celé pokrýt jedním modulem a proto se musí generovat více modulů. Položky Start a Délka se přímo zadávají do RetosDebuggeru při tvorbě Memory Design. Base paměti ROM je adresa, kde začíná paměť ROM. Tato adresa se musí zadávat při nahrávání programu do simulátoru Eprom.

Procesor s pamětí Eprom nebo FLASH 512Kx1=64kB
standardní GAL=adresa F0000h až FFFFFh, z toho BIOS FE000h až FFFFFh

Paměti	START(adresa segmentu)	DÉLKA(v bytech)
RWM (128kB)	00B0h	20000h
RWM (512kB)	00B0h	80000h
ROM1	F000h	E000h
ROM2	-	-
BÁZE paměti ROM	F000h	

Procesor s pamětí Eprom nebo FLASH 1Mx1=128kB
standardní GAL=adresa E0000h až FFFFFh, z toho BIOS FE000h až FFFFFh

Paměti	START(adresa segmentu)	DÉLKA(v bytech)
RWM (128kB)	00B0h	20000h
RWM (512kB)	00B0h	80000h
ROM1	E000h	1E000h
ROM2		
BÁZE paměti ROM	E000h	

Procesor s pamětí Eprom nebo FLASH 2Mx1=256kB
standardní GAL=adresa 80000h až 87FFFh a C8000h až FFFFFh,
z toho BIOS FE000h až FFFFFh

Paměti	START(adresa segmentu)	DÉLKA(v bytech)
RWM (128kB)	00B0h	20000h
RWM (512kB)	00B0h	80000h
ROM1	C800h	18000h
ROM2	E000h	1E000h
ROM3	8000h	800h
BÁZE paměti ROM	C000h	

Procesor s pamětí Eprom nebo FLASH 4Mx1=512kB
standardní GAL=adresa 80000h až 9FFFFh a C8000h až FFFFFh,
z toho BIOS FE000h až FFFFFh

Paměti	START(adresa segmentu)	DÉLKA(v bytech)
RWM (128kB)	00B0h	20000h
RWM (512kB)	00B0h	80000h
ROM1	C800h	18000h
ROM2	E000h	1E000h
ROM3	8000h	20000h
BÁZE paměti ROM	8000h	

7. Možnosti ladění programu

Jednou ze silných stránek šestnáctibitového softwarového prostředí ***SofCon*** je možnost pohodlného ladění aplikace ve vývojovém prostředí Borland Pascal.

a. Aplikaci lze ladit v první fázi ve versi DOS v počítači PC a místo hardware napsat simulátory. Ty se spustí jako další procesy operačního systému RETOS, simulují funkci hardware a mohou také periodicky vypisovat obsah zajímavých proměnných na displej, nebo děj interpretovat graficky na monitoru PC. V tomto stupni se dá aplikace velmi dobře vytvořit a zhruba odladit v integrovaném prostředí Borland Pascal 7, nebo pod TurboDebugerem. Nepotřebujeme k tomu nic jiného než počítač PC.

Když už potřebujeme přistupovat na reálný hardware, máme několik možností ladění:

b. Můžeme zůstat v režimu překladu DOS na počítači PC a počítač doplnit kartou PCKit. To je vlastně interface z ISA sběrnice na obě sběrnice použité v procesoru Kit. Na kartě je Pbus i IOBus, pouze se nacházejí na jiných adresách. K desce PCKit připojíme ostatní hardware a z počítače PC vlastně uděláme Kit. Můžeme ladit v integrovaném prostředí i pod TurboDebugerem jako v předchozím případě. Rozdíl je v rychlosti počítačů. Dále musíme nastavit jiné adresy I/O prostoru, jiné vektory přerušení a nemůžeme přistupovat na absolutní adresy paměti jako v KITovi. Změny se snadno realizují podmíněným překladem. Tento režim poskytuje nejmocnější ladicí možnosti reálné aplikace. Takto se dají odladit aplikace řízení celých velkých strojů.

c. Přejdeme s programem do reálného počítače KitV40 nebo Kit386, ke kterému připojíme kartu Vega. Na Vega monitor budeme vypisovat obsahy proměnných, které nás zajímají. Aplikaci nelze krokovat, běží skoro správnou rychlostí, pouze ji zdržují výpisy na Vegu. Výpisy mohou být prováděny periodicky ve zvláštním procesu a tím si vlastně vytvoříme velice pružný zobrazovač stavu aplikace, toho co nás zrovna zajímá. Proměnné vidíme v tom tvaru, ve kterém je chceme interpretovat. Výpisy na monitor z procesoru Kit se doporučuje tvořit v alfanumerickém režimu. Grafika většinou zabere moc času procesoru. K procesoru KitV40 se dá připojit pouze osmibitová Vega s ISO sběrnicí přes kartu EXPPC00. K procesoru Kit386 se dá připojit Vega se sběrnicí PC104.

d. K aplikaci připojíme další proces KernelMonitor s nejvyšší prioritou a zavedeme ji do procesoru Kit. Aplikace se zvětší a zpomalí oproti reálnému běhu. KernelMonitor umí komunikovat s RetosDebuggerem a umí pozastavovat procesy, zjišťovat stav procesů, zjišťovat obsah proměnných. Pokud je aplikace nahraná v paměti RWM procesoru, umí KernelMonitor i nastavit BreakPointy do programu. Vše je podrobně popsáno v příručce o RetosDebuggeru.

e. Můžeme pro výpis proměnných použít obrazovku Term10 a pracovat jako v bodě c.

8. Několik poznámek k tvorbě aplikačního programu

V následujících podkapitolách bych rád objasnil několik důležitých informací o programování procesorů KitV40 a Kit386 v šestnáctibitovém prostředí ***SofCon***.

8.1. RETOS

Operační systém reálného času RETOS poskytuje prostředky pro pseudoparalelní běh více úloh, které obsluhují danou aplikaci. (Například jednu úlohu tvoří automat řídící stroj, druhá úloha komunikuje s uživatelem, třetí s nadřízeným počítačem a navzájem komunikují přes proměnné PASCALu, nebo přes schránky RETOSu.) Uživatel ve své aplikaci o.s.RETOS může, ale také nemusí použít. Z knihovních modulů používají RETOS pouze terminálové a regulační knihovny. I při použití těchto knihoven se však dá RETOS vynechat. Pro aplikaci není nutné používat naráz všechny prostředky operačního systému. Například se běžně používají prostředky pro paralelismus, ale mnohem méně často schránky operačního systému. Je totiž mnohem jednodušší komunikovat mezi úlohami přes obyčejné proměnné PASCALu, než přes schránky operačního systému.

Při použití RETOSu se trochu zamlží současnost dějů. Dva děje obsluhované různými úlohami jsou totiž současné až tehdy, když operačním systémem proběhnou obsluhy obou úloh, které děje obsluhují. Mezi během těchto dvou úloh se vkládá alespoň jeden časový Tick operačního systému. To je implicitně 55ms, ale dá se to rozumně zkrátit až na 10 ms. Pokud potřebujeme rychlejší odezvu na nějakou událost, je lepší ji obsloužit mimo operační systém, např. v obsluze přerušení Int08.

RETOS umožňuje nastavit úlohám prioritu, jak často je bude jádro systému vyvolávat. Lze použít dvě základní strategie přidělování priorit.

1. Úlohám lze nastavit různou statickou prioritu a potom úloha s vyšší prioritou se musí vzdávat svého času ve prospěch úlohy s nižší prioritou. Pokud tak neučiní, úlohy s nižší prioritou nebudou obsluhovány. Ovšem úloze s vyšší prioritou to dává možnost pouštět čas pouze tehdy, když si to může dovolit a má jistotu, že každý tick RETOSu bude vyvolaná.
2. Úlohy mají stejnou statickou prioritu a liší se prioritou dynamickou. To zabezpečuje spravedlivé přidělování času, ovšem pokud nechce být úloha rušena, musí si na čas zvednout prioritu, nebo dočasně zakázat přepínání úloh. Tento mód sice zaručuje spravedlivé přidělování času, ale ne úplně pravidelné.

8.2. Přerušení

Do periferních desek jsou zavedeny IRQ3 a IRQ4. Tato přerušení používají komunikační desky a obsluhují je komunikační knihovny. Naše knihovny umí pracovat s přerušením systémového časovače Int08. Umí ho modifikovat, zrychlovat či zpomalovat a zavěšovat na něj další úlohy. Nemaskovatelné přerušení je obslouženo Biosem, ale prázdnou procedurou, nic se nedělá.

8.3. Automaty

Programátoři zvyklí programovat PLC automaty mohou způsobem ala PLC naprogramovat jednotku obsluhující hardware úlohy. Tuto hlavní jednotku, je dokonce s výhodné programovat jako automat. Tato jednotka bude mít svůj vlastní proces RETOSu a bude sdílet čas s ostatními knihovními jednotkami, které například obsluhují terminál, nebo s někým komunikují. Pokud chce někdo úplně vynechat RETOS, tak i to je možné. Napíše vše jako jeden automat a všechny komunikace i postrkování terminálu může provádět sám z hlavního automatu.

8.4. Zálohovaná paměť RWM

Paměť RWM procesoru KITV40 je zálohovaná baterií. To způsobuje, že pascalské proměnné nejsou při prvním spuštění programu vynulovány, jak to bývá v PC a naopak po prvním průchodu programu už mají správnou hodnotu. Například jednou vytvořený dynamický objekt je už vlastně vytvořený trvale. Jeho pointer i proměnné, pokud je destructor nezruší, zůstávají zachovány i po vypnutí napájení. Je zapotřebí mnohem více dbát na inicializaci všech proměnných než v systémech s PC. Jako dobrá metoda se jeví používat místo obyčejných proměnných inicializované konstanty.

Naopak proměnné, které zaručeně chceme zachovat i po výpadku napájení a různých haváriích programu je výhodné umístit je na absolutní adresu úplně mimo oblast RWM modulu programu. Když RWM modul zmenšíme od konce RWM paměti (20000h, 40000h, 80000h) o velikost těchto proměnných, tak budou umístěné úplně mimo data segment a ani přeteklý stack je nepoškodí.

8.5. Paměť Flash

Jedna z dodávaných knihoven umí zapisovat do paměti flash. V této paměti nemusí být pouze code segment, ale i data zapisovaná procesorem. Jen je třeba dbát na maximální počet přepsání jednoho sektoru paměti a na pokus o zápis do paměti Flash jako do RWM. Ochrana paměti totiž způsobí načtení nesmyslných dat code segmentu programu a tím pád programu.

Přepsáním prvního bloku programu s hlavičkou ROM modulu se program stane pro Bios nespustitelným. Pokud potom softwarově zresetujeme procesor, dostaneme se do Biosu a můžeme po komunikaci nahrávat novou versi programu. Tato akce se používá, když chceme nahrávat po komunikaci nové verse a přitom nemáme přístup k propojkám KitV40 nebo Kit386.

8.6. WatchDog

Obvod WatchDog umístěný na procesorové desce je občerstvovaný z Biosu v proceduře obsluhy systémového časovače. To zajistí jeho spolehlivé obsloužení i v aplikačním programu. Ovšem při zhroucení aplikačního programu se málokdy poškodí také obsluhy přerušení a WatchDog zůstane dále občerstvován. Lepší je, pomocí knihovního modulu si vytvořit vlastní obsluhu obvodu WatchDog, ovšem až v odladěné aplikaci.

8.7. RunTime Errors

Borland Pascal může generovat runtime errors od spousty chyb programu. Je výhodné v aplikaci nechat tyto kontroly zapnuté, poněvadž to nijak významně program nezpomaluje ani nezvětšuje a přitom je program hlídán. V MCP versi je třeba v Exit proceduře zapisovat adresu a číslo chyby do zálohované oblasti RWM a mít nástroj, kterým se dá tato oblast číst. Pokud program spadne, tak tuto oblast přečteme a z dresy pomocí RetosDebuggeru zjistíme na které řádce programu k chybě došlo. Ve startovacích aplikacích je jak tato zálohovaná struktura, tak exit procedura založená.

9. Startovací aplikace

Na CD disku se nacházejí adresáře BEGINT10 a BEGN_T01. V těchto adresářích se nacházejí založené minimální aplikace s terminálem Term 10 a s Term 01. Aplikace obsahují úvodní bitmapu, první menu obrazovky, přechod do servisní obrazovky chráněné heslem, obrazovku nastavení hesla a obrazovku výpisu RuntimeErrors. Aplikace mají založené hlavní unity a procesy. Mají založenou strukturu Glb a její inicializaci, která je uložena v takové oblasti paměti RWM, ve které se data spolehlivě udrží díky lithiové baterii i po vypnutí napájení. V Exit proceduře je zápis adresy a čísla RunError do Glb struktury. V servisní obrazovce je možné poškodit hlavičku Rom modulu v paměti Flash, takže lze běžící program softwarově zrušit, přejít do Biosu a nahrát novou versi. V DOS versi se používá simulátor terminálu. Aplikace mají založený soubor Sets.Inc s podmínkami překladu.

Doporučujeme při zakládání nové aplikace vycházet z těchto minimálních aplikací a rozvíjet je.