VYDÁNÍ TŘETÍ

2018

Řízení procesu GROWE®

Uživatelský manuál

Řízení systémových procesů

Projekt GROWE 2018

Řízení procesu GROWE®

Uživatelský manuál

Projekt GROWE 2012-18

Vydání **TŘETÍ**, **ver. I.** - **User Type**, sestavení **#062_UTM Beta VII**.

Poslední změna: **28.10.2023 22:22:27**

Nejnovější verzi manuálu Projekt GROWE naleznete na adrese:

http://projekt-growe.cz/wp-content/uploads/2017/01/Manual_GROWE-UserVersion.pdf

Nejnovější verzi manuálu Skript Studio I. naleznete na adrese:

http://projekt-growe.cz/wp-content/uploads/2017/05/Manual-Skript-Studio-I.pdf

Screnshoty (obrázky obsahu obrazovek) v manuálu se mohou lišit od fyzické podoby programů v závislosti na verzích vývoje při tvorbě manuálu. Tyto změny jsou zpravidla minoritní, pokud je změna rozsáhlejšího rázu, je manuál po implementaci změn a **BETA** testech aktualizován.

Veškeré dotazy týkající se distribuce tohoto manuálu směřujte na:

E-mailovou adresu: support@projekt-growe.cz

Webová adresa projektu: www.projekt-growe.cz

OBSAH

1. PROJEKT GROWE

1.1. POPIS PROJEKTU

Projekt, vznikl jako nápad v době, kdy jsem se spolupodílel na projektu pece pro tavení hliníku, plně řízené PC. Hardwarová základna byla pro řízení tak náročného procesu rozsáhlá, počítač sám o sobě obsahoval několik řídících „desek" propojených na výkonové spínače, krokové motory se zpětnou vazbou, senzory teploty, průchodu vzduchu, plynu zplodin a dalších.

Po několika letech jsem se k nápadu řízení procesů pomocí **PC** vrátil, kdy jsem potřeboval 24 hodinový záznam z procesu **GROWE** a automatickou regulací dle příkazového souboru s dynamickou změnou na základě měřených veličin (teploty, času).

Hlavní obrazovka programu Řízení MINI I.

1.2. FILOZOFIE ŘÍZENÍ PROCESU

Filozofie řízení procesu **GROWE** spočívá na principu vytvoření řídícího skriptu-souboru příkazů po jednotlivých dnech pro celý cyklus. Tento je po vytvoření překontrolován a následně načten do hlavního řídícího programu. Ten pak běží v plně automatickém režimu, kdy dochází pouze ke kontrole dosahovaných hodnot při běhu cyklu, a to jak lokálně, což je odečet hodnot na obrazovce, tak vzdáleně, pomocí dálkové správy. Měřené hodnoty a hlášení řídícího programu jsou ukládány do **.LOG** souborů k další analýze.

1.3. NÁVRH HARDWARE A SOFTWARE

Softwarová základna je postavena na požadavku maximální spolehlivosti, kdy nesmí dojít k jeho pádu, ani k nestandardnímu chování, nebo ovlivňování jeho běhu dalšími spuštěnými programy. Proto je program postaven na operačním systému **MS-DOS**, který splňuje všechny požadavky na použitelnost,

rychlost, spolehlivost, kapacitu. Programy jsou vytvořeny v prostředí **Microsoft Visual Basic for DOS ver. 1.0 Pro**, komunikační rutiny jsou vytvořeny v assembleru **MASM for DOS ver. 6.11 PDS**. Verze s podporou dálkové správy používá komunikační protokol **TCP/IP**.

Pro řízení procesu a komunikace s periferiemi (reléová deska, teplotní čidlo) je zvoleno klasické **PC** s porty **COM** a **LPT**. Řídící deska je napojená na **LPT** portu a obsahuje 8 relé spínající napětí **240V/4A** se zpětnou vazbou - kontrolou stavu jednotlivých relé. Teplotní čidlo DS18B20 je připojeno na sériovém portu **COM**.

Komunikace přes **LPT** port je zajištěno přímým adresováním a čtením portu, teplota je čtena pomocí externího programu **DIGITEMP**, se zápisem hodnot do souboru.

Řídící reléová deska je uložena do plastového rozvaděče odolného proti vlhkému prostředí a spíná 7 zásuvek **240 V**, na které jsou připojeny koncové spotřebiče. Nad zásuvkami jsou kontrolní **LED** diody, které jsou přímo napájeny spínaným napětím **240 V**, a tak slouží jako vizuální kontrola stavu napájení.

Rozvaděč má hlavní vypínač, kterým lze bezpečně vypnout hlavní napájení, zemnění připojených zařízení zůstává zachováno, stejně jako funkce řízení, tj. ovládání jednotlivých relé.

1.4. PROGRAM ŘÍZENÍ MINI I.

Program slouží k plně automatickému řízení procesu **GROWE** dle předdefinovaného skriptu. Skript obsahuje příkazy po jednotlivých dnech pro ovládání 7 spínacích relé. Tyto spínají a vypínají napájení jednotlivých spotřebičů procesu **GROWE** jako hlavní a vedlejší osvětlení, zálivka, ventilace a podobně. Dále v tomto procesu se snímají veličiny (teplota na světle a ve stínu), podle kterých se spínají a vypínají spotřebiče k udržení optimálních hodnot v procesu. Jako příklad řízení je možné uvést stav, kdy nastane překročení limitu teploty a sepne se přídavné odsávání, bez ohledu na příkaz uvedený v řídícím skriptu. Takto poběží až do snížení teploty pod kritickou hranici, poté dojde k vypnutí přídavného odsávání. Dále pak běží příkazy dle řídícího skriptu.

Program je navržen na plně automatický běh, bez nutnosti zásahů uživatele. Na začátku procesu **GROWE** se nadefinuje řídící skript, vloží se do řídícího programu a tento se spustí. Nicméně toto je až jeden z posledních kroků, předchází mu kompletní zapojení všech spotřebičů, kontrola funkčnosti, natavení jejich vlastností (průtok čerpadla, výkon zvlhčovače) a další konfigurace prostředí.

Nicméně manuální ovládání je umožněno, a to jak lokálně tj. přepnutí programu do manuálního modu, a ovládání jednotlivých relé klávesnicí, tak vzdáleně, a to pomocí komunikačního programu **Client Remote Control**, pomocí kterého se lze připojit po síti **LAN** s řídícím programem. Pro tuto komunikaci se používají síťové služby **Microsoft** (protokol **TCP/IP**, služba sdílení **RAM** disku). Vzdálená služba je vytvořena systémem **Client-Server**, má několik úrovní zabezpečení použití příkazů, kdy např. uživatel bez uživatelských práv je schopen zjišťovat běh vzdálené služby, testovat rychlost odpovědi služby **SERVER**, ale nemůže zasáhnout do procesu. Po úspěšném přihlášení ke službě **SERVER** (pomocí uživatelského jména a hesla) dojde k navýšení práv a s tímto i rozšíření dostupných příkazů.

Uživatel s právy **USER** může např. zjišťovat a měnit systémový datum a čas na řídícím **PC** (počítač, na kterém běží řídící program), zjišťovat stav a aktuální hodnoty procesu, odeslání logovacích souborů s

historií záznamu všech klíčových událostí, zobrazení aktuálního náhledu na hlavní obrazovku řídícího programu.

Pokud se uživatel připojí s právy **ADMINISTRATOR** může navíc provádět přerušení automatického běhu procesu, změna stavů jednotlivých relé (zapnutí – vypnutí spotřebičů), spuštění auto testu řídící reléové desky (kompletní test se zjišťováním kompletní funkčnosti všech relé systémem – zapnutí relé, zjištění stavu, vypnutí relé, zjištění stavu). Dále může zaměnit řídící skript za jiný, restartovat proces **GROWE** (dojde k restartu programu s načtením řídících příkazů, např. po záměně skriptu), deaktivovat služby v řídícím programu jako je odpojení čtení stavu klávesnice (program nebude možné lokálně ovládat, ani přerušit jeho běh), restart služby dálkové správy (např. při chybě komunikace) a další. Služba **SERVER** navíc umí automaticky odpojit přihlášeného uživatele po určité době nečinnosti, pro zajištění bezpečnosti proti neoprávněnému přístupu do procesu.

Veškerá komunikace probíhá v šifrované podobě, lze ji možno proto použít pro vzdálený přístup přes internet. Služba SERVER zasílá všechny soubory (**.LOG** soubory, řídící skript) v šifrované podobě, pro jeho dešifrování slouží program **Cryptor eXtended**, který je součástí programového balíku.

Jako další program, který je rovněž součástí tohoto balíku je program **Builder Pro**, pro tvorbu a editaci řídícího skriptu.

Pro kontrolu obsahu tohoto skriptu slouží program **DBF Read**, který po jeho úspěšné kontrole přidá do skriptu digitální podpis s CRC kontrolním součtem. Pokud by došlo po vložení podpisu do skriptu k jeho jakékoli změně obsahu, zneplatní se tím tento digitální podpis. A při dálkové změně skriptu by (podle nastavení stupně ochrany řídícího programu) může dojít k odmítnutí tohoto skriptu, výměna v tomto případě neproběhne a program bude dále používat původní skript.

Všechny přístupy a zásahy do řídícího programu jsou ukládány do **.LOG** souborů, pro další analýzu, popř. zjišťování chyb při běhu programu.

Program sám je vytvořen tak, aby byl schopen se vypořádat se vzniklými chybami při běhu, takto vzniklá chyba je zapsána do **.LOG** souboru a program běží dále bez přerušení. Pro udržení funkcionality běhu procesu **GROWE** je naprosto nežádoucí jakékoli přerušení běhu programu, s výjimkou úmyslného přerušení ze strany uživatele, kdy tento "ví, co dělá...". Program umí i vyřešit chyby ve čtení externích hodnot (např. teploty), kdy by mohlo dojít nárůstu hodnot na kritickou hranici. Vyhodnotí cyklicky se opakující se chybu a upraví běh programu na "bezpečný běh", kdy procesy, které by mohly být klíčové v nárůstu hodnot nad hraniční, budou nastaveny na stav vypnuto, jejich funkci převezme "bezpečné zařízení" a program ignoruje příkazy pro jejich opětovné zapnutí. V tomto stavu program setrvá do zásahu uživatele, který je v tomto případě klíčový pro nápravu chyby a změně do standardního běhu programu.

Pro příklad lze uvést stav, kdy dojde k chybě při čtení teploty. Program zjistí cyklickou chybu čtení této hodnoty. V této situaci není možné monitorovat výši této hodnoty, hrozí situace, že dojde k dosažení nebo překročení kritické hodnoty, a protože o této situaci program "neví", nedojde k zapnutí přídavného odsávání. Po několikátém neúspěšném čtení, program vyhodnotí chybu jako trvalou, a protože je předpoklad nárůstu této hodnoty, dojde k vypnutí hlavního osvětlení, které toto teplo produkuje. Ale aby nedošlo k nežádoucímu přerušení osvětlení, sepne se náhradní zářivkové osvětlení, které nezpůsobuje nárůst teploty, ale nepřeruší světelný cyklus. Program potom ohlásí trvalou chybu

a v tomto stavu čeká na zásah uživatele, který vymění vadné teplotní čidlo a restartuje řídící program. Tento přepne zpět hlavní osvětlení dle příkazu z řídícího skriptu a proces běží dále.

Samotný program má i ochranu proti chybě neprovedení příkazu z řídícího skriptu. Dobrým příkladem je v tomto případě stav, že dojde k přerušení cyklické zálivky co 15 minut. Kontrola stavu vlhkosti měřenou čidlem není v současné verzi programu zahrnuta (nicméně po přechodu na platformu **Mini-PC** je možné, že tato funkce bude dopracována) a hrozí riziko, že zálivka se neprovede. Pro kontrolu platnosti stavu všech relé (a tím i spouštěné příslušné spotřebiče) běží paralelně procedura, která nezávisle kontroluje fyzický stav všech relé a jejich okamžitých stavů, vyplývající z řídícího skriptu. Tato kontrola se nazývá **validace**, pokud dojde k neshodě mezi fyzickou a kontrolní hodnotou, nastane stav, kdy běh procesu **GROWE** může být negativně ovlivněn a změněn oproti požadavkům podle řídícího skriptu. Pokud se chyba detekuje jako cyklická, musí dojít k nápravě tohoto stavu a v této chvíli program zaznamená chybu a restartuje se do výchozího stavu. Poté dále normálně pokračuje v běhu procesu **GROWE**.

Program obsahuje 4 nezávislé obrazovky (pracovní plochy), na které se zapisují aktuální informace, o řídícím skriptu, stavu validace, seznam příkazů pro aktuální den, o stavu teplot a jejich max. a min. dosažených hodnot, stavu jednotlivých relé, stavu odsávání a dalších. Přepínání mezi nimi probíhá za použití klávesových zkratek.

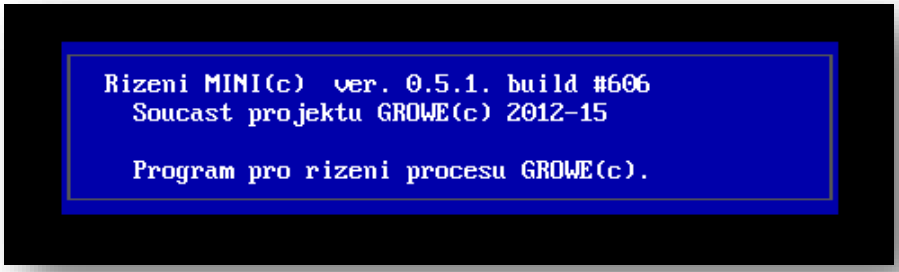
Program je dostupný pro běžné (nikoli ale volné) použití a distribuci, z toho důvodu je běh programu svázán s licenčním souborem. V něm jsou informace o hardware **PC**, které jsou při startu programu čteny a porovnávány s aktuálním hardware, na němž je program spuštěný. Na zařízeních, pro které nebyla vydána licence běží program pouze v **DEMO** módu. O neplatnosti licence je uživatel při startu programu o této skutečnosti informován.

Program má možnosti nastavení konfigurace uložené v **.INI** souboru. Konfigurační informace hlavního programu jsou uloženy v souboru **RIZENIMI.INI**, konfigurace služby vzdálené správy používá soubor **SERVER.INI**. Soubory **.INI** jsou ve formátu zápisu konfiguračních souborů **Windows 3.1**. Zápis do nich je možný, pokud "víte, co děláte". Pokud dojde k neplatnému zápisu určité konfigurační hodnoty, program ji odmítne a načte se výchozí hodnota a v souboru **.INI** se na tuto hodnotu neplatný záznam přepíše, nebo vloží nový, platný. Stejný postup je zvolen při smazání těchto souborů, při spuštění programu dojde k jejich novému vytvoření s výchozími hodnotami, které jsou i načteny.

1.5. SPUŠTĚNÍ PROGRAMU

Program se spustí příkazem **MINI.EXE**, s následujícími nepovinnými přepínači:

/manual	program umožní manuální ovládaní, automatické řízení je vypnuto
/nosaver	vypne spořič obrazovky
/version	vypíše informaci o verzi programu a ukončí se
/demo	spustí **DEMO** režim – zápis na **LPT** port je simulovaný
/help	zobrazí nápovědný text a ukončí se
/file:xxxxxxxx.DBF	načte řídicí skript **xxxxxxxx.DBF**. Pokud není přepínač použit, je hledán v konfiguračním souboru .INI, pokud není nalezen ani tam, je zobrazen dialog na výběr souboru
/testboard	provede se test řídící reléové desky na portu **LPT** a program se ukončí



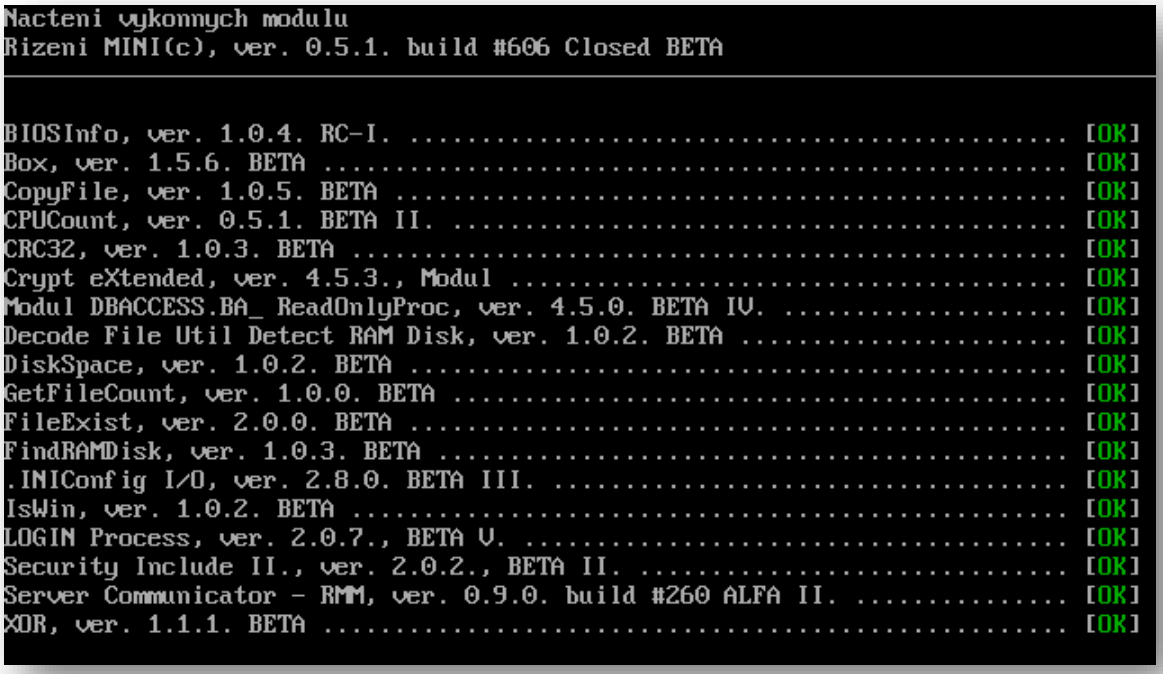
Úvodní hlášení při spuštění

Příklad spuštění programu:

MINI.EXE /nosaver /file:skript.DBF

Program se spustí bez aktivního spořiče obrazovky a použije jako řídící skript soubor **SKRIPT.DBF**

POZNÁMKA:
*příkazy pro spuštění programu mohou být zadávány jak malými, tak velkými písmeny, operační systém **MS-DOS** není CaSe SeNsItIvE, tz. nerozlišuje mezi těmito znaky. Proto příkazy **MINI.EXE /DEMO** a **mini.exe /demo** jsou naprosto totožné.*



Načítání výkonných modulů programu

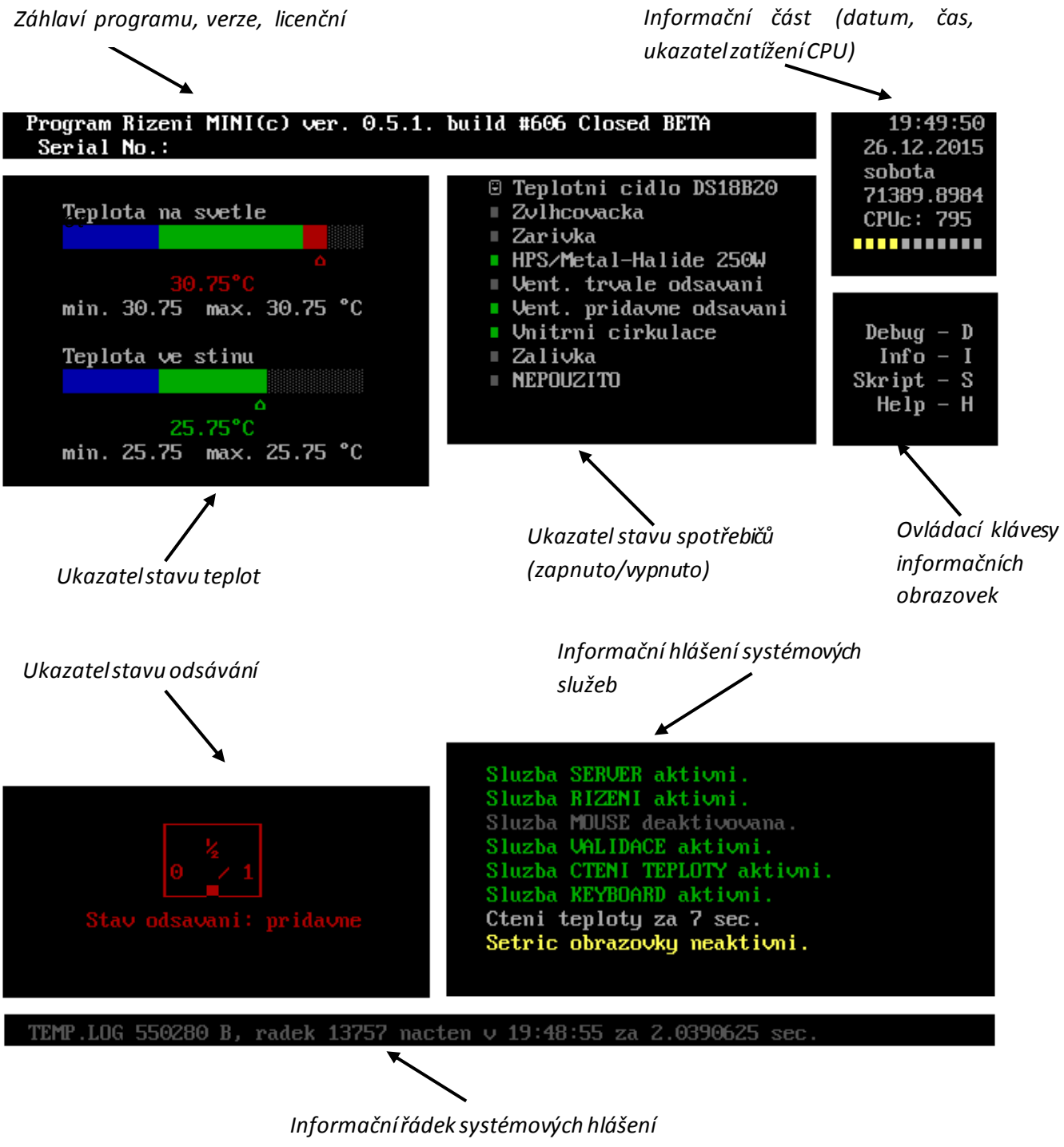
Taktéž všechny výkonné moduly jsou opatřeny informací o verzi vývoje, viz. obrázek Načítání výkonných modulů programu. Každý modul má určitou úlohu při běhu programu, pouze aktuální verze zajistí úplnou funkcionalitu systému jako celku. Pokud např. bude Váš program obsahovat zastaralý modul **Server Communicator**, nebude dálková správa obsahovat nejnovější příkazy. Samotná funkčnost a stabilita programu tím není nijak narušena, všechny moduly prošly náročným **BETA** testováním. Pokud by se přesto v programu vyskytla chyba, je informace o verzi důležitou informací pro její úspěšnou nápravu.

UPOZORNĚNÍ:

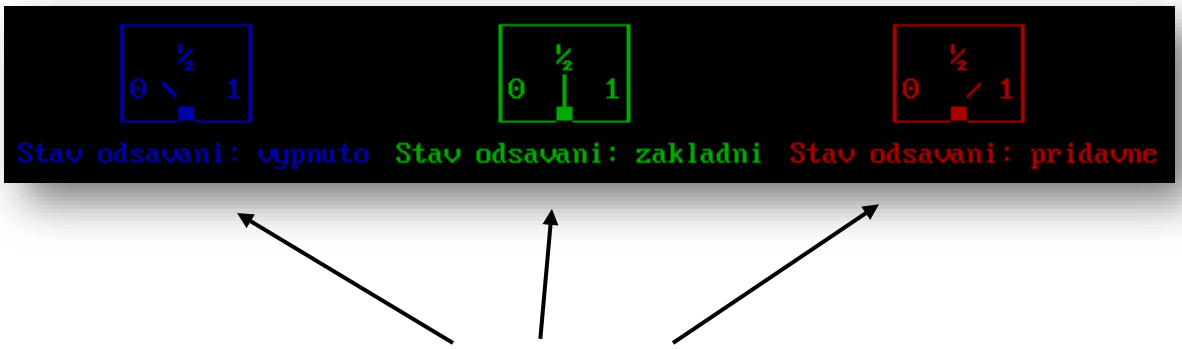
*Prosím věnujte pozornost úvodním informacím o programu, zobrazující se při startu. Program se neustále vyvíjí (vylepšuje se funkčnost a opravují možné chyby), proto je označen informací o verzi a buildu (číslo sestavení). Pokud je verze programu nižší než **1.x.x.**, jedná se s největší pravděpodobností o vývojovou verzi ve stádiu vývoje **ALFA** nebo **BETA**. Prosím aktualizujte program na nejnovější verzi, postup je popsán v kapitole* **7.2. Aktualizace programů**

1.6. HLAVNÍ A POMOCNÉ INFORMAČNÍ OBRAZOVKY

1.6.1. HLAVNÍ OBRAZOVKA



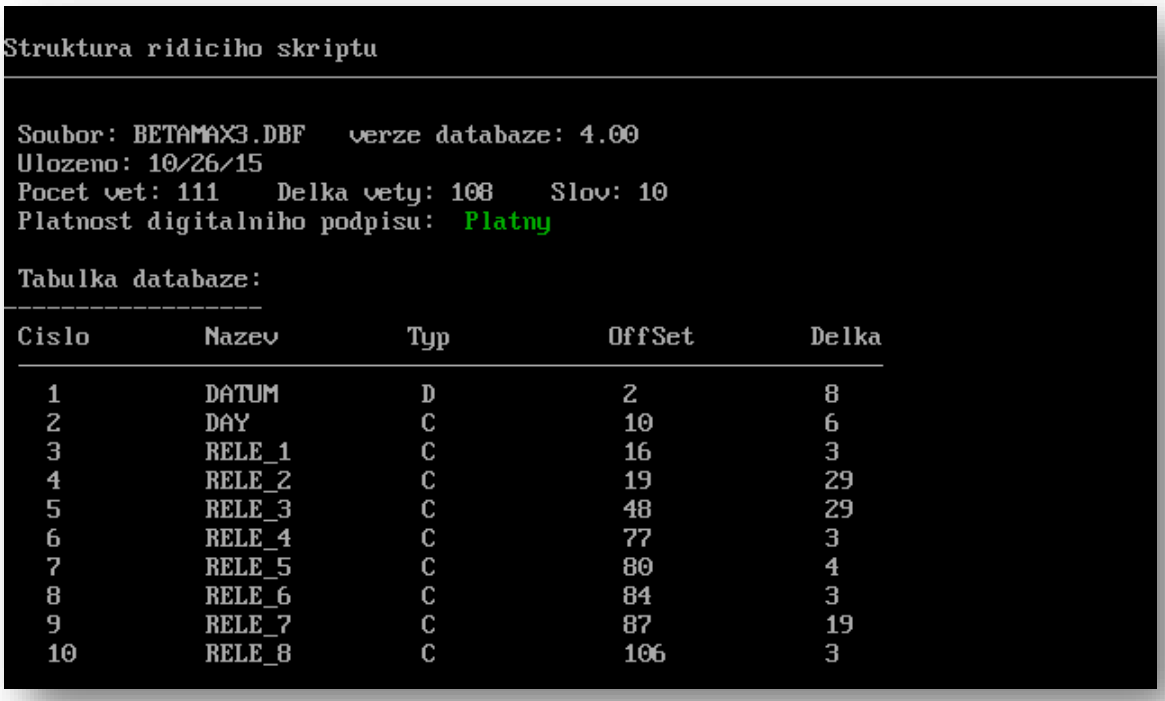
Jak je popsáno v kapitole [6.3. Program Skript Builder Pro], existují tři stavy funkce odsávání, **základní přídavné** a **vypnuto**. Každý z těchto stavů je odlišen ukazatelem a jeho barvou. Pro názorný příklad jsou zde uvedeny.



Jednotlivé stavy odsávání

1.6.2. OBRAZOVKA STRUKTURA ŘÍDÍCÍHO SKRIPTU

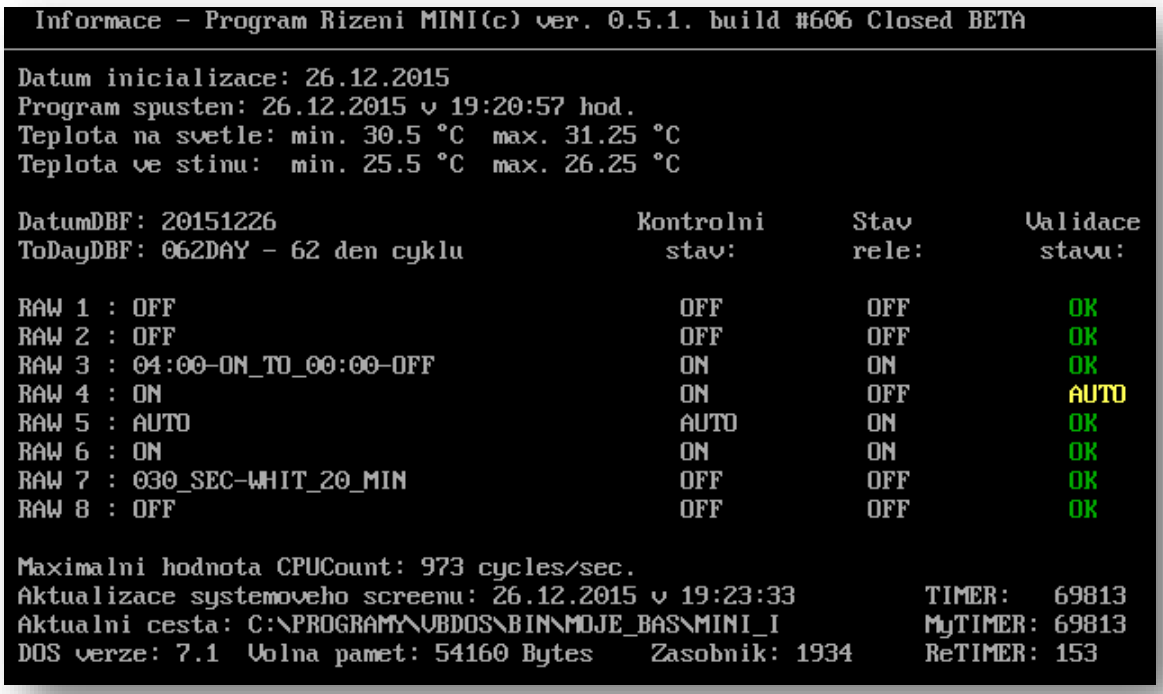
Obrazovka obsahuje podrobné informace o řídícím skriptu, název souboru **.DBF**, verze databáze, datum uložení, počet vět, délka věty, platnost digitálního podpisu a tabulka databáze.



Náhled na obrazovku informací o struktuře řídícího skriptu

1.6.3. OBRAZOVKA SYSTÉMOVÉ INFORMACE

Obrazovka systémových informací obsahuje data získaná při běhu programu. Obsahuje datum inicializace skriptu, datum a čas spuštění/restartování programu, minimální a maximální dosažená teplota, pořadový den cyklu, načtené příkazy pro aktuální den a spotřebiče, kontrolní a fyzický stav sepnutí spotřebičů, validace stavu, min. hodnota **CPUCount**, aktuální adresář, ze stavu paměti a zásobníku, verze **OS** a pomocné proměnné pro řízení časových událostí. Sběr dat probíhá nepřetržitě od spustění a čím je program v běhu déle, tím data získávají na přesnosti a vypovídací hodnotě. Posbíraná data se ukládají do **.LOG** souborů, po ukončení procesu **GROWE** je lze použít pro podrobnější kterého je program spuštěn, informace o analýzu. Kopii těchto souborů lze také kdykoli získat pomocí programu dálkové správy.

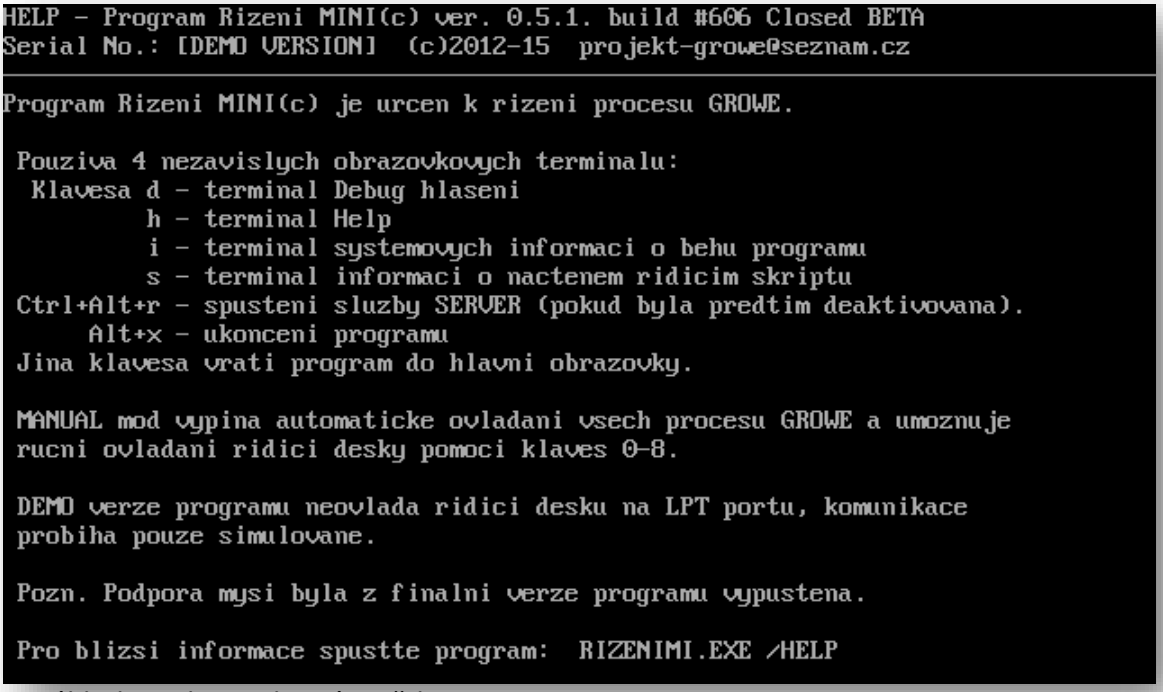


Náhled na obrazovku systémových informací

Důležitými informacemi jsou sloupce dat je Kontrolní stav, Stav relé a Validace stavu. Kontrolní stav vyjadřuje předpokládaný stav relé, vypočtený kontrolní procedurou na základě příkazů. Stav rele je fyzický stav jednotlivých spotřebičů čtených přímo z řídící desky a Validace stavu je výsledek porovnávající oba stavy. Pokud jsou shodné, je stav řízení tzv. validní – bezchybný, pokud ale dojde k rozporu mezi oběma výsledky, musí program na tento rozpor adekvátně zareagovat a prvést nápravu, pokud je to nutné. Takto je hlídán spolehlivý chod procesu, pokud nevalidní stav překročí stanovenou mez, vyhlásí program alarm a převede se do vegetativního (udržovacího stavu), kdy již je nutný zásah uživatele.

1.6.4. OBRAZOVKA NÁPOVĚDY – HELP

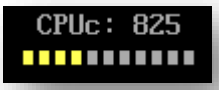
Obrazovka nápovědy obsahují základní informace o použití a ovládání programu. Podrobné informace jsou v uvedeny v tomto manuálu, nápovědná obrazovka nemůže obsáhnou svou velikostí ani část problematiky řízení procesu **GROWE**, proto i informace na ní jsou pouze obecné.



Náhled na obrazovku nápovědy

1.7. DŮLEŽITÉ FUNKCE PROGRAMU

1.7.1. CPUCOUNT – UKAZATEL STAVU VYTÍŽENÍ SYSTÉMU



Ukazatel vytížení

Ukazatel vytížení systému CPUCount měří hodnotu **CPUc,** což je počet cyklů za 1 sec. Tato hodnota je měřena nepřetržitě a slouží k informování o vytížení systému. Program při svém běhu neustále oslovuje všechny důležité části, které se podílejí na procesu GROWE, měření teploty, stav spotřebičů ve vztahu k příkazům v řídícím skriptu a mnoho dalších. Každá tato funkce programu zabírá určitý procesorový čas, prostě trvá určitou chvíli. Pokud by došlo k vytvoření fronty, kde čekají funkce na své provedení a v součtu časů, které tyto procesy potřebují pro své vykonání, by zaměstnaly centrální procesor (CPU) na několik sekund, mohlo by dojít k promeškání některého příkazu, což je neakceptovatelné z důvodu spolehlivosti.

1.7.2. KRYPTOVÁNÍ

Kryptování (šifrování) je proces, kdy důležité části programu, které mohou být dostupné třetím stranám, jsou zakódovány způsobem, který umožňuje naprosto znečitelnit těchto dat a otočením procesu data zase obnovit do původního stavu. Takto se přenášejí příkazy při dálkové správě,

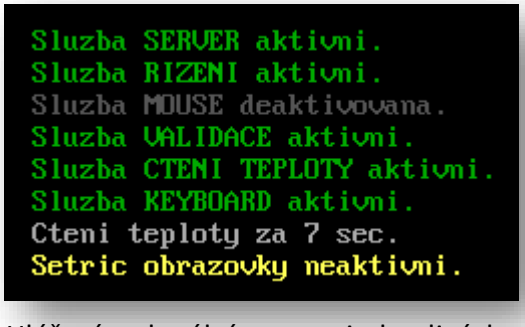
zakryptovány jsou také soubory obsahující přihlašovací údaje a veškerá přenášená data. Pro kryptování a dekryptování slouží utilita **Cryptor eXtender Pro.**

1.7.3. SLUŽBY

Služby jsou části programu, vykonávající určitou tematicky zaměřenou činnost, např. dálkovou správu. Služby mohou být deaktivovány a opět aktivovány za běhu hlavního programu. Obsahují i sebe-kontrolní mechanizmus, kdy při vzniku chyby zachycené službou, se tato automaticky odpojí z činnosti, aby nenarušila běh hlavního programu a provede se zápis o chybě do **.LOG** souboru.

Seznam služeb:

- **Server** - služba dálkové správy
- **Řízení** - služba automatického běhu procesu GROWE podle řídícího skriptu.
- **Validace** - služba pro kontrolu běhu hlavního programu nezávislou procedurou.
- **Čtení teploty** - služba čtení teploty z čidla Dallas DS18B20.
- **Keyboard** - služba čtení vstupu z klávesnice.
- **Spořič obrazovky** - služba vypínající výpis na obrazovku, zápis těchto informací běží na pozadí.



Hlášení o aktuálním stavu jednotlivých služeb

Podle barvy jednotlivých služeb lze rychle „zjistit“ jejich stav a funkcionalitu. **Zelená** standardně označuje bezchybný chod, naopak **červená** označuje závažnou chybu v běhu služby. Pokud je služba označena **žlutě**, je dočasně vypnuta, pokud je označena **šedě**, je deaktivována (např. nastavením v konfiguračním souboru).

1.8. KONFIGURACE PROGRAMU

Konfigurace programu se provádí nastavením dostupných parametrů v konfiguračních souborech **.INI**. Hlavní řídící program používá tyto soubory:

- **MINI.INI**
- **RELDESK.INI**

- **SERVER.INI**

Každý z těchto souborů obsahuje konfigurační parametry pro určitou funkcionalitu programu. Zápis je ve v textové formě, **MS-INI** formátu. Pokud není zápis syntakticky správný, je tento údaj nebo údaje ignorovány a načteny výchozí hodnoty.

Formát obsahu souboru dle specifikace:

První záznam-hlavička .INI souboru, je vždy uzavřen v hranatých závorkách [INI]. Pokud není nalezen, je načítání přerušeno chybovým hlášením. Název sekce je rovněž uzavřen v hranatých závorkách. Každá sekce pak obsahuje několik klíčů, minimálně vsak jeden. Obsah jedné sekce končí označením další sekce. Každý klíč je vždy na samostatném řádku. Klíč je ve tvaru NÁZEV_KLÍČE=HODNOTA. Názvy sekcí a klíčů nesmí obsahovat mezery na počátku a konci. Mezery mezi slovy jsou povoleny. Veškeré zápisy jsou CaSe SeNsItIvE a takto jsou i vyhledávány. Řádky začínající znakem # jsou komentáře a mohou se vyskytovat na kterémkoli řádku v souboru. Řádky s komentáři a prázdné řádky jsou ignorovány.

```
[INI]
[configuration]
TemperatureStatusBar1=Teplota na svetle
TemperatureStatusBar2=Teplota ve stinu
FanStatusBar1=Stav odsavani
ScreenSaverDelay=0
ScriptFileName=betamax3.dbf
IntervalReadTemperature=60
ResetLPTOnStart=0
ServerOnStart=1
Sound=0
```

Příklad obsahu konfiguračního souboru

UPOZORNĚNÍ:

INI soubor musí být uložen ve formátu text ANSII. Čtení a zápis formátu Unicode a UTF-8 nefunguje správně.

Popis struktury .INI souboru:

[INI]	'Povinná hlavička souboru
[Files]	'Název sekce
one=Hallo	'Název klíče, za znakem "=" je hodnota klíče
pi=3.1415927	

POZNÁMKA:

Pokud je v textu uveden znak apostrof, ' označuje se tím, že text za tímto znakem je poznámka uživatele a je programem ignorována.

1.8.1. SOUBOR MINI.INI

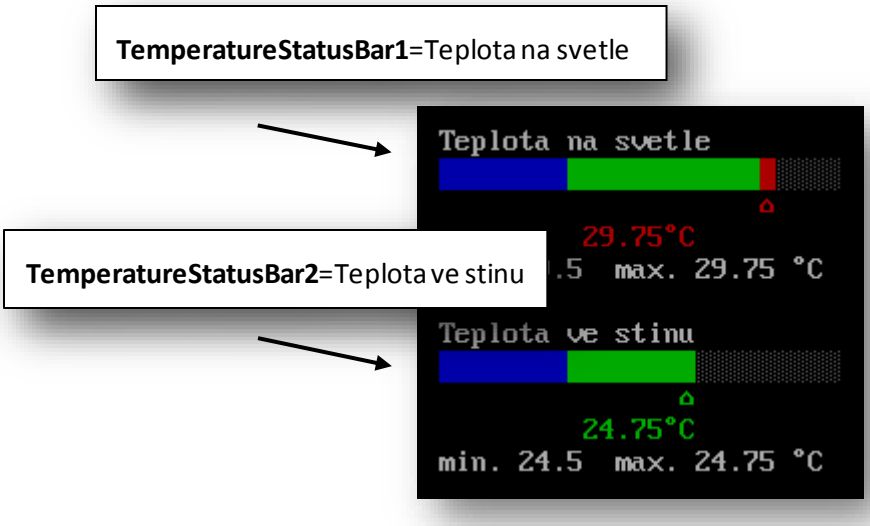
Textový soubor, obsahující výchozí konfigurační proměnné, důležité pro běh programu MINI.EXE a jeho umístění je v adresáři C:\GROWE\RIZENI\BIN

```
[INI]
[configuration]
TemperatureStatusBar1=Teplota na svetle
TemperatureStatusBar2=Teplota ve stinu
FanStatusBar1=Stav odsavani
ScreenSaverDelay=0
ScriptFileName=betamax3.dbf
IntervalReadTemperature=60
ResetLPTOnStart=0
ServerOnStart=1
Sound=0
```

Příklad souboru MINI.INI

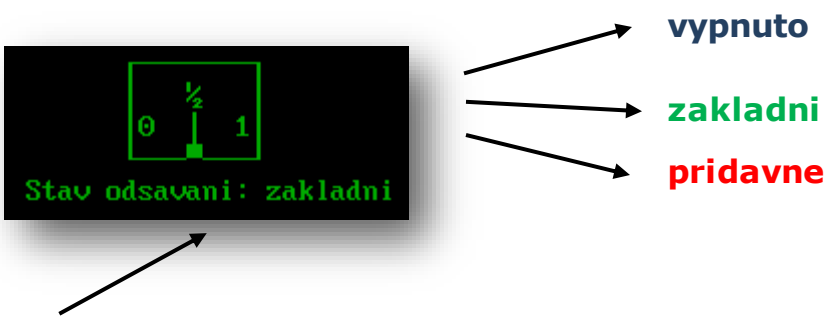
Popis jednotlivých konfiguračních proměnných hlavního programu:

TemperatureStatusBar1-2 - Názvy 2. grafických ukazatelů teploty. Max. délka názvu není stanovena, neměla by výrazně přesáhnout délku ukazatele, aby nenarušil integritu hlavní obrazovky.



Ukazatele aktuální teploty na hlavní obrazovce

FanStatusBar1 - Název ukazatele odsávání, ke kterému se automaticky přidává text podle stavu (vypnuto, základní, přídavné), proto max. délka názvu by neměla výrazně přesáhnout přiměřenou délku (viz. obrázek), aby nenarušil integritu hlavní obrazovky.



FanStatusBar1=Stav odsavani:

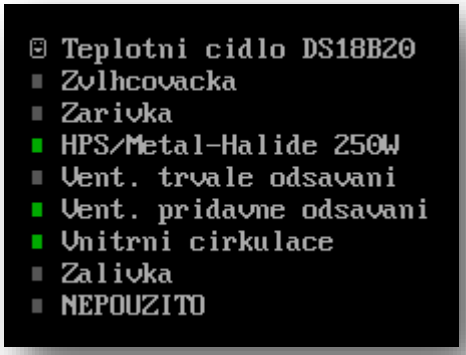
- **ScreenSaverDelay** - Délka prodlevy před spuštěním šetřiče obrazovky v sec. Maximální délka prodlevy je 32768 sec. pokud je hodnota 0, je šetřič vypnutý.
- **ScriptFileName** - Název souboru řídícího skriptu. Název musí být ve formátu 8+3, s koncovkou .DBF
- **IntervalReadTemperature** - Interval čtení teploty v sec. Interval musí být zde nastaven na stejnou hodnotu jako externí utilita pro čtení teploty (viz. 4.5. Nastavení teplotního čidla). Minimální hodnota by neměla být menší než 2-3, z důvodu rizika zahlcení programu příliš často se opakujícími se operacemi. Nicméně hodnota by měla být zvolena optimálně, podle odhadu prostředí na rychlost změn hodnot.
- **ResetLPTOnStart** - RESET řídící desky při startu programu do výchozího stavu (tj. vše vypnuto), hodnota 0 - reset se neprovede, hodnota 1 - reset se provede při každém spuštění.
- **ServerOnStart** - Spuštění služby dálkové správy při běhu programu řízení. Hodnota 0 - služba se nespustí, hodnota 1 - služba se spustí.
- **Sound** - Aktivace výstražných zvukových signálů, hodnota 0 - signalizace vypnuta, hodnota 1 - signalizace zapnuta.

1.8.2. SOUBOR RELDESK.INI

umístění v adresáři C:\GROWE\RIZENI\BIN

```
[INI]
[configuration]
TempSensorName_1=Teplotni cidlo DS18B20
TempSensorName_2=not used
DeviceName_1=Zvlhcovacka
DeviceName_2=Zarivka
DeviceName_3=HPS/Metal-Halide 250W
DeviceName_4=Vent. trvale odsavani
DeviceName_5=Vent. pridavne odsavani
DeviceName_6=Vnitrni cirkulace
DeviceName_7=Zalivka
DeviceName_8=NEPOUZITO
```

Příklad .INI souboru



Zobrazení údajů v řídícím programu

Na příkladu vpravo je názorně vidět, jak se obsah .INI souboru, obsahující popisky spotřebičů, promítne do zobrazení těchto popisků v hlavním řídícím programu.

1.8.3. SOUBOR SERVER.INI

```
[INI]
[configuration]
Vertex=m:\
LogOFFTimer=0
```

Příklad souboru SERVER.INI

umístění v adresáři C:\GROWE\RIZENI\BIN

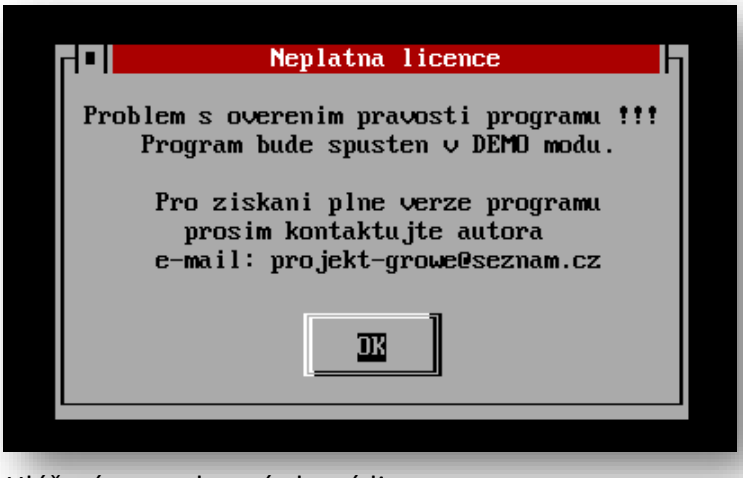
Popis jednotlivých konfiguračních proměnných služby dálkové správy:

Vertex - písmeno komunikačního uzlu, viz. kapitola **4.3. Konfigurace VERTEXu – komunikačního uzlu.**

LogOFFTimer - prodleva v sec., po které je přihlášený uživatel odhlášen při nečinnosti. Maximální hodnota je 65535 sec., pokud je hodnota 0, je funkce automatického odhlášení vypnuta.

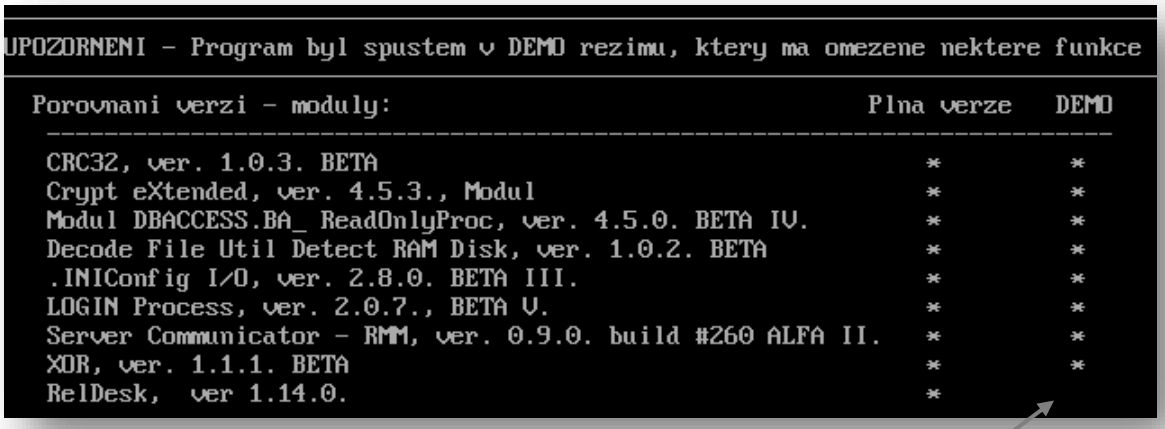
1.9. KONTROLA PLATNOSTI LICENCE PROGRAMU

Ochrana proti nedovolenému kopírování programu spočívá v tom, že každá instalace je pevné svázaná s výrobním číslem a názvem základní desky PC. Při startu programu je kontrolována platnost licence uložená v souboru **LICENCE.KEY** a případě neplatnosti licence se program spustí v demonstračním režimu. V tomto případě se na obrazovku vypíše hlášení o problému s ověřením pravosti kopie.



Hlášení o nenalezení platné licence

Co je to **DEMO** mód? Program běží naprosto stejně jako plná verze, funkční je i dálková správa s jednou výjimkou. Neprobíhá komunikace s řídící deskou spínající jednotlivá zařízení. Pro možnost otestování běhu programu je tato komunikace simulovaná, tj. program běží naprosto stejně, kontrolky jednotlivých relé ukazují stav sepnutí, běží kontrola Validace stavu, pouze řídící deska je **TRVALE** ve vypnutém stavu. Je proto nezbytné, aby licenční soubor **LICENCE.KEY** byl trvale přítomen ve stejném adresáři jako hlavní řídící program. Soubor je zakrytován a opatřen kontrolním součtem **CRC**, jakákoliv



Porovnání modulů v DEMO a plné verze programu

změna jeho obsahu způsobí zneplatnění licence. Proto důrazně **NEDOPORUČUJEME** experimentovat s jeho obsahem.

Chybějící modul pro řízení

Při ukončení programu běžícím v **DEMO** režimu se zobrazí tabulka porovnání výkonných modulů, které obsahují plná a demo verze. Modul **RelDesk**, sloužící pro komunikaci s řídící reléovou deskou a **není součástí demo verze**.

Pro získání plné verze programu, nebo platné licence prosím kontaktujte autora programu. Kontaktní údaje jsou uvedeny na konci manuálu.

Platnost licenčních údajů je klíčová, pouhá přítomnost licenčního souboru nemusí být důvodem k uznání platnosti licence a běhu programu v plném režimu.

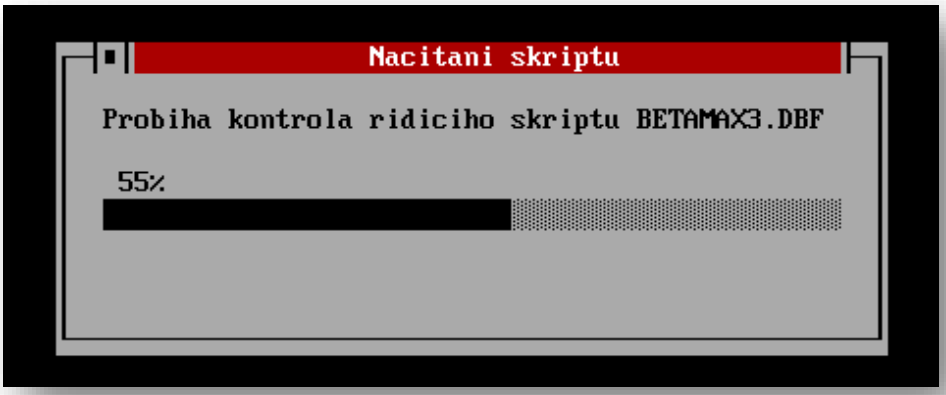
2. ŘÍDÍCÍ SKRIPT

2.1. POPIS SKRIPTU

Řídící skript je soubor příkazů, kterými se program řídí po celou dobu automatického běhu procesu **GROWE**. Při spuštění programu se načtou příkazy pro aktuální den a program je provádí až do půlnoci, kdy jsou načteny nové, aktuální pro další den.

2.2. PŘÍKAZY SKRIPTU

Řídící příkazy jsou uloženy v databázovém souboru **.DBF** ve formátu **dBase ver. 3.xx** s rozšířenou hlavičkou **eXtHead**. Tato obsahuje dodatečné informace, rozšiřující možnosti tohoto formátu.



Načítání a kontrola řídícího skriptu při startu programu Řízení MINI

Příkazy řídicího skriptu pro jednotlivé spotřebiče:

OFF - spotřebič vypnut
ON - spotřebič zapnut
AUTO - automatický režim řízení (ovládá se za pomoci měření aktuálního stavu teplot)
TO hh:mm-ON_TO_hh:mm-OFF - v hh:mm (hodina:minuta) zapnout, v hh:mm (hodina:minuta) vypnout
WHIT sss_SEC-WHIT_mm_MIN - zapnout na sss(sekund) co mm(minut), příkaz se cyklicky opakuje, začíná vždy zapnutím na sss(sekund), **přesná definice logického významu** je - po uplynutí "mm" minut zapnout na "sss" sec., cyklicky

Upozornění:
Skript (soubor databáze) není z důvodu obsahu eXtHead, kompatibilní s formáty dBase ver. 3.xx, 4.xx.

2.3. HISTORIE FORMÁTU .DBF

dBase byl první masověji rozšířený systém řízení báze dat (**SŘBD**, **DBMS**) pro mikropočítače, který byl publikován americkou firmou **Ashton-Tate** pro **CP/M**. A později pro **Apple II**, **Apple Macintosh**, **UNIX[1]**, **VMS[2]** a **IBM PC** pod **DOS**, kde se na mnoho let stal jedním z nejprodávanějších softwarových titulů. **dBase** pomalu úspěšně přecházel pod **Microsoft Windows** a pozvolna uvolňoval podíl na trhu svým konkurentům **Paradox**, **Clipper**, **FlagShip**, **FoxPro** a **Microsoft Access**. **dBase** byl prodán **Borlandu** v roce 1991, který prodal práva k této sortimentní skupině v roce 1999 tehdy nově vzniklé společnosti **dBase Inc**. V roce 2004 **dBase Inc**. změnila svůj název na **dataBased Intelligence, Inc**.

2.3.1. VZNIK DBASE 1

Původním vývojářem **dBase** byl **Wayne Ratliff**. V roce 1978 **Ratliff** napsal databázový program v assembleru pro mikropočítače založené na **CP/M**, aby mu pomohl vyhrát kancelářskou fotbalovou sázku. Založil ho na **JPLDIS** (**J**et **P**ropulsion **L**aboratory **D**isplay **I**nformation **S**ystem) **Jeba Longa** a nazval ho "**Vulcan**" po panu **Spockovi** ze seriálu **Star Trek**. Podle **Ratliffa** byl jazyk použitý v **JPLDIS** jednoduchý. Jsou zde známky toho, že **JPLDIS** byl ovlivněn mainframovským databázovým produktem zvaným **RETRIEVE**.

Na počátku 80. let 20. století, **George Tate** z **Ashton-Tate** uzavřel s **Ratliffem** obchodní dohodu. **Vulcan** se přejmenoval na **dBase** a software rychle vzrostl na popularitě.

2.3.2. DBASE – OBJEKTOVĚ ORIENTOVANÝ JAZYK

dBase byl zařazen mezi moderní objektově orientované jazyky, které běží na 32-bitových **Windows**. Může být použit k výstavbě aplikací se širokým využitím, např. webové aplikace provozované na Windows serveru. **dBase** má přístup k většině moderních databází pomocí rozhraní **ODBC**.

2.3.3. XBASE

Počátkem poloviny osmdesátých let 20. století mnoho jiných společností přišlo s vlastními variantami daného produktu a jazyka. Tyto zahrnovaly **FoxPro** (nyní **Visual FoxPro**), **Arago**, **Force**, **dbFast**, **dbXL**, **Quicksilver**, **Clipper**, **Xbase++**, **FlagShip**, **Recital Terminal Developer**, **CodeBase**, **MultiBase** a **Harbour/XHarbour**. Dohromady jsou označovány jako **xBase**.

[1] Kapitola 2.3.1. – 2.3.5. je převzata z: Wikipedia, **internetová encyklopedie**. Text je sdílen pod licencí Creative Commons Uveďte původ-Zachovejte licenci 3.0 Unported, případně za dalších podmínek.

2.3.4. SOUBOR DATABÁZE – OBECNÝ FORMÁT .DBF

Výchozím formátem pro **dBase** je soubor .DBF. Ten se hojně využívá v mnoha dalších aplikacích, kde je potřeba jednoduchý formát pro uchování strukturovaných dat. V současné době mu ale v tomto ohledu konkuruje SQLite.

Pro každou tabulku byl jeden soubor .**DBF**. Extra soubory **dBase** vytvářela pro sloupce typu memo (v souboru .**DBT**; viz dále) a pro index (.**IDX**) nebo indexy (.**MDX**). Organizace tabulek do databází na úrovni **SŘBD** neexistovala, stejně jako např. pohledy, cizí klíče, triggery a jiné prvky pozdějších databázových systémů. Přístup více uživatelů řešily jednotlivé aplikace na úrovni ope račního systému. Přístupové údaje pro přístup k tabulkám chyběly.

Jméno tabulky bylo určeno jménem souboru. Soubor na začátku obsahoval definici sloupců, následovanou jednotlivými řádky. Sloupce mohly mít název dlouhý maximálně 11 znaků a mohly být těchto typů (první písmeno je kód typu, používaný v souboru):

C – řetězec (v SQL typ char) se specifikovanou délkou (maximálně 255 znaků)
L – logická hodnota (s hodnotami Y nebo T pro pravdu, N nebo F pro nepravdu)
N – číslo, u kterého lze specifikovat maximální počet číslic
F – číslo, u kterého lze specifikovat maximální počet desetinných míst
D – datum (v SQL typ date) od roku 0000 do roku 9999
M – memo (v SQL typ text) text do délky 64 kB

Řádky měly stejnou délku (kratší řetězce se doplňovaly \0 zprava a čísla nulami zleva), takže šly rychle procházet. Každý záznam obsahoval indikátor platnosti. Mazání záznamů sestávalo samotné indikace neplatnosti. Tím ale vznikala fragmentace. Pro optimalizaci velikosti se tedy dal soubor zabalit/spakovat (program v něm prošel všechny záznamy a ty platné záznamy srotil k začátku souboru a zbytek souboru „usekl“).

Obsah souborů **.DBT** je rozsekaný na základní bloky o délce 512 bajtů. Tyto bloky jsou interně číslovány (počínaje nulou) a v souboru **.DBF** je u sloupců typu memo uloženo, kde (na bloku jakého čísla) začíná odpovídající text. Text je vždy doplněný na nejbližší násobek čísla 512.

2.3.5. VÝVOJ A SOUČASNOST

Vývoj formátu pokračoval pod křídly **Microsoftu** až do 2007, kdy byl vydán, jako součást programu **Visual FoxPro 9.0 SP2** – objektový nástroj k vytvoření a správě vysoce výkonných 32bitových databázových aplikací a komponent. Jeho další vývoj byl ukončen.

2.4. STRUKTURA SKRIPTU

Délka všech polí ve větě databáze je VŽDY stejná, i když neobsahuje žádnou hodnotu, proto jsou i jednotlivé věty databáze stejně dlouhé. Jsou to tzv. sekvence, odtud název sekvenční databáze.

Jednotlivé věty jsou skládány jedna za druhou, proto je snadné načítání jenotlivých vět a dle délky pole, definované v tabulce rozsekat větu na jednotlivá pole

Příklad věty struktury databáze:

Hlavička databáze 32 bytes (16 bytes dBase def. + 16 bytes eXtHead def.)									
DATUM Typ D Pozice 2 + Délka 8 =	**DAY** Typ C Pozice 10 + Délka 6 =	**RELE_1** Typ C Pozice 16 + Délka 21 =	**RELE_2** Typ C Pozice 37 + Délka 21 =	**RELE_3** Typ C Pozice 58 + Délka 21 =	**RELE_4** Typ C Pozice 79 + Délka 21 =	**RELE_5** Typ C Pozice 100 + Délka 21 =	**RELE_6** Typ C Pozice 121 + Délka 21 =	**RELE_7** Typ C Pozice 142 + Délka 21 =	**RELE_8** Typ C Pozice 163 + Délka 21 = celková délka věty

Náhled na hlavičku databáze (včetně tabulky) v Hexa editoru

Náhled na hlavičku souboru databáze odhalí fyzické uložení. Jak bylo již uvedeno, hlavička je dlouhá 32 bytes, z toho je 16 bytes **dBase** základní definice **+ 16 bytes eXtHead**. Za ní je uložena tabulka databáze. Je definována proměnnými, kterým je přiděleno pořadové číslo, název, typ, offset a délka.

2.5. ROZŠÍŘENÁ HLAVIČKA SOUBORU .DBF VER. 4.X.X.

Řídící skript obsahuje hlavičku, což je prostor na začátku souboru, kde prvních 32-bytes souboru obsahuje konfigurační informace o skriptu.

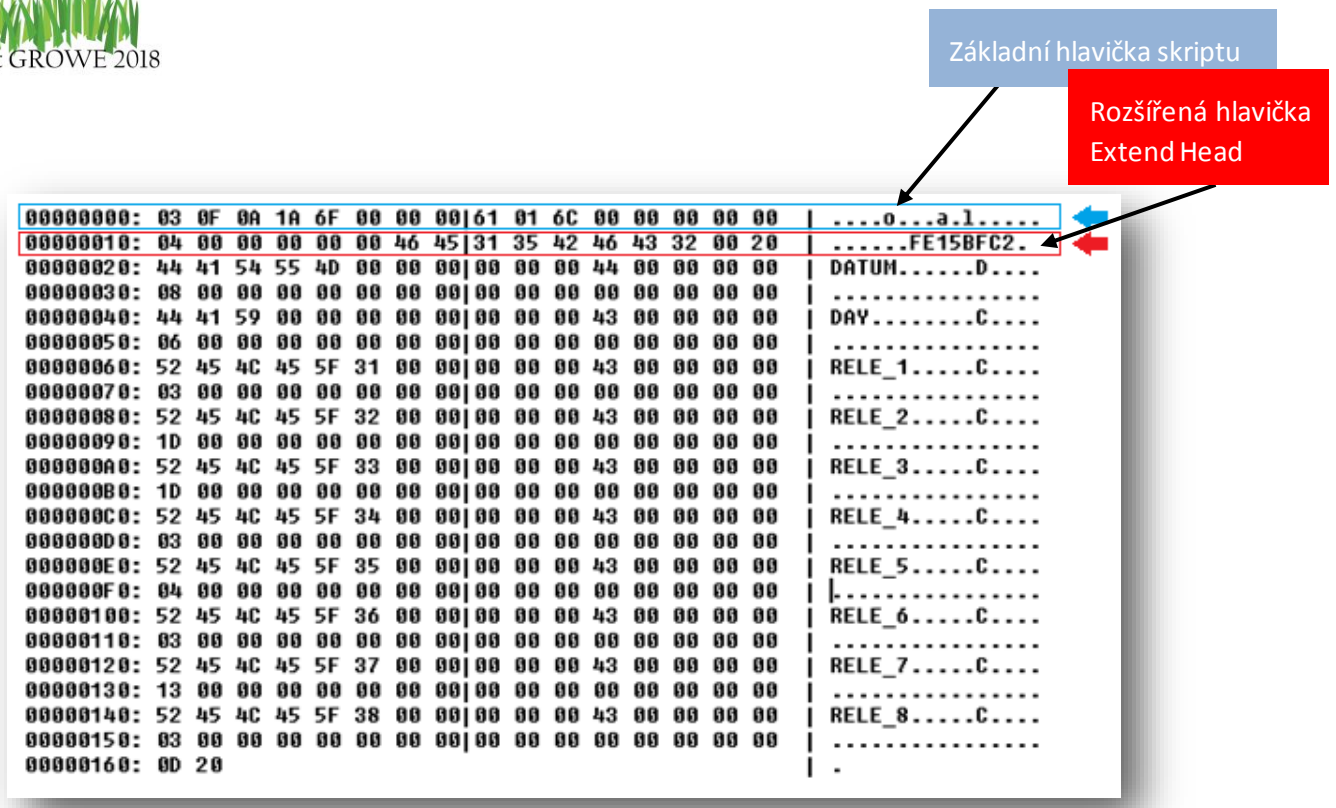
```
00000000: 03 0F 0A 1A 6F 00 00 00|61 01 6C 00 00 00 00 00  | ....o...a.l.....
00000010: 04 00 00 00 00 00 46 45|31 35 42 46 43 32 00 20  | ......FE15BFC2.
```

Hlavička souboru

Hlavička databáze dle specifikace GROWE:

Pozice	obsah
1	dBASE verze 3, nebo &H83 typ memo
2	Rok
3	Měsíc
4	Den posledního update
5-8	Celkový počet záznamů v souboru (long integer)
9-10	Počet záznamů, uložených v hlavičce (integer) [2]
11-12	Délka záznamu v souboru (integer)
13-32	Rozšířená hlavička - Extend Head

[2] Číslo vyjadřující počet záznamů (vět) v databázi, je uloženo v hlavičce a v určitých případech se může lišit od fyzického počtu v databázi. Proto na tento údaj není brán zřetel a vždy se počítá fyzický počet záznamů.



Hlavička včetně tabulky databáze

Hlavička skriptu	pozice	délka	typ proměnné	popis
Header.Version = ASC(Buffer$)	1	1 byte	Version AS INTEGER	Verze skriptu .DBF
Header.Rok = ASC(MID$(Buffer$, 2, 1))	2	1 byte	Year AS INTEGER	Datum souboru rok, formát rr
Header.Mesic = ASC(MID$(Buffer$, 3, 1))	3	1 byte	Month AS INTEGER	Datum souboru měsíc, formát mm
Header.Den = ASC(MID$(Buffer$, 4, 1))	4	1 byte	Day AS INTEGER	Datum souboru den, formát dd
Header.TRecs = CVL(MID$(Buffer$, 5, 4))	5-8	4 bytes	TRecs AS LONG	Počet vet v databázi
NEPOUZITO	9-10	2 bytes	TFields AS INTEGER	Počet polí ve větě
Header.RecLen = CVI(MID$(Buffer$, 11, 2))	11-12	2 bytes	RecLen AS INTEGER	Délka vety
REZERVOVÁNO pro zpětnou kompatibilitu	13-15	3 bytes	Memo AS INTEGER	Příznak není využit
Header.ExtendHEAD = MID$(Buffer$, 17, 16)	16-32	16 bytes	ExtendHead AS STRING*16	Rozšířená hlavička

Rozšířená hlavička **Extend Head** (**eXtHead**), obsahující dodatečné a rozšiřující informace je součástí základní hlavičky. Využívá nepoužívaných 16 bytes ze spodní části hlavičky.

Popis rozšířené hlavičky:

Hodnoty v hlavičce jsou rozděleny na příznaky a r_znaky (řídicí znaky. **Příznaky** určuji vlastnosti databáze, zatímco **řídící** (r_znaky) primo mění vlastnosti databáze (např. v tomto případě zamezuji zápis na úrovni modulu DBACCESS).

1. **příznak** - verze databáze
   volba CHR$(3) - příznak pro formát verze 3.00
   nebo CHR$(4) - příznak pro formát verze 4.xx
   jiny příznak - nastavení příznaku pro formát verze 3.00 kvůli zpětné kompatibilitě

2. **r_znak** - ochrana proti zápisu do databáze
   volba CHR$(240) - zápis ZAKAZAN
   jiná volba - zápis POVOLEN

3. příznak - kryptování obsah databáze, volba CHR$(1) - obsah JE zakrytován
jiná volba - obsah NENI zakrytován

4. příznak - inicializace skriptu, volba CHR$(1) - skript JE inicializován
jiná volba - skript NENI inicializován

5. příznak - originalita skriptu, volba CHR$(0) - originální skript
volba +=1 - zaměněny skript (hodnota zn. pořadí, např. záměna dálkovou správou)

6. příznak - použiti skriptu v procesu RIZENI, hodnota CHR$(1) - proces GROWE spuštěn/restartován
jiná hodnota - skript dosud nepoužít

7. příznak - ukončení CELEHO procesu GROWE (příznak ověřeného skriptu)
volba CHR$(0) - skript dosud nepoužít
volba CHR$ +=1 - počet dokončených cyklu

8.-15. znak - 8. Bytes CRC, kontrolního součtu

16. znak - **G** - databáze GROWE
A - (any) všeobecná databáze

Příklad eXtendHead:

04	00	00	00	00	00	46	45	31	35	42	46	43	32	00	20

POZNÁMKA:
Kontrolován je pouze obsah databáze, kdy může docházet ke změnám v hlavičce bez zneplatnění podpisu.

2.6. NASTAVENÍ OCHRANY ZÁPISU SKRIPTU

Do skriptu lze přidat příznak **POUZE ČTENÍ**, který aktivně zamezuje jakémukoli zápisu do obsahu. Tato ochrana je aktivní pouze u programů projektu GROWE, kde je tato ochrana na úrovni fyzického přístupu k souboru databáze. Toto však neplatí pro editaci na úrovni OS DOS/Windows98, kdy provedení fyzických změn v souboru prakticky není možné zamezit. Pro ochranu obsahu proto slouží digitální podpis, kdy jakákoli změna v datech jej zneplatní společně s příznakem pro zákaz zápisu. Jediná možnost, jak provést změnu v datech a vložit platný podpis, je použít některý z programů projektu GROWE, které ale akceptují příznak **POUZE ČTENÍ** a provedení změn tedy neumožní. Kombinace ochran obsahu skriptu je dostatečná pro zamezení použití náhodně přepsaného obsahu skriptu, pro který je i určena.

Header.ReadOnly – delka	2.byte - CHR$(240)	- aktivní příznak pouze čtení
	jiný znak	- zápis povolen

2.7. RUČNÍ EDITACE SKRIPTU

Program DBF_READ.EXE umožňuje i smazání již existující inicializace, kdy smaže datum v první větě, ostatní programy tento skript budou považovat za neinicializovaný, to znamená, že teoreticky je ve stavu prvotního vytvoření, kdy datumová řada ještě nebyla do skriptu vložena. Skutečnost, že se maže jen první záznam, umožňuje obnovení původní datumové řady, ale tato operace je natolik nestandardní, požadující jednoznačnou znalost toho "že uživatel ví, co dělá", že zde nebude podrobně popsána. Její poměrná nesnadnost, vyžadující nízko úrovňovou editaci v HEXA editoru a přesné vložení datumové řady do každého prvního záznamu ve větě do souboru. Alternativou je možnost použití utility DBEDIT.EXE, který je součástí programového balíku. Podrobný popis této utility je v samostatném souboru, v elektronické podobě na disku v adresáři (vyzkoušejte C:\GROWE\UTIL\DBACCESS\MAN). Existuje jistá pravděpodobnost, že se Vám tato operace nepodaří a dojde k porušení konzistence databáze. Prosím vytvořte si před "operací" zálohu souboru, ať neztratíte možnost návratu k funkční verzi skriptu. Při závažném porušení konzistence není zaručena, že bude možná obnova obsahu databáze, zvláště zranitelná v tomto je kryptovaný obsah (a to i v případu částečného zakryptování).

Pokud je inicializace skriptu smazána, nebo pokud ještě nebyla vložena, program nabídne její vložení. V tomto případě dojde k vložení celé datumové řady, nebo k jejímu přepisu a obnova v tomto případě již není možná.

2.8. PROGRAM DBF_READ

Pro proces inicializace skriptu se používá program **.DBF Reader Pro,** soubor **DBF_READ.EXE**. Tento program postupně zanalyzuje celý skript v několika krocích, kdy kontroluje platnost všech údajů v hlavičce souboru, dále syntaktická a logická správnost všech příkazů, pokud jsou všechny testy úspěšné, provede zálohu skriptu a jeho inicializaci. V opačném případě ohlásí nalezenou chybu a nabídne možnost opravy. Po uložení opravených údajů se celý kontrolní proces provede znovu. Pokud opět nalezne chybu, opět ji ohlásí, po opravě znovu spustí test a toto probíhá až do chvíle, kdy skript vyhoví testům. Teprve tehdy program umožní jeho inicializaci.

Při spuštění programu lze použít následující přepínače:

/file:nazev.DBF	- specifikuje .DBF soubor, který bude otevřen
/version	- vypíše informace o verzi a ukončí se
/clear	- smaže inicializaci otevřeného souboru .DBF
/info	- zobrazí podrobné informace o programu
/?	- zobrazí okno nápovědy, která popisuje možnosti spuštění programu

2.9. INICIALIZACE SKRIPTU

Inicializace skriptu je proces, při němž se do všech vět přidají k příkazům i jednotlivé datumy, v postupné řadě za sebou podle kalendáře. Podle nich se budou načítat příkazy pro tyto dny, v případě přerušení běhu hlavního programu, tento ví, kde má pokračovat. Procesu inicializace předchází

kompletní kontrola skriptu, pokud nejsou žádné chyby nalezeny, je provedení inicializace automaticky nabídnuto. V případě odmítnutí jsou další operace přerušeny a program ukončen. Pokud ale budete s provedením inicializace souhlasit, provede se nejdříve záloha původního souboru, poté se do souboru vloží již výše zmíněná datumová řada. První vložený datum je aktuální datum, kdy se inicializace provedla.

2.10. DIGITÁLNÍ PODPIS SKRIPTU

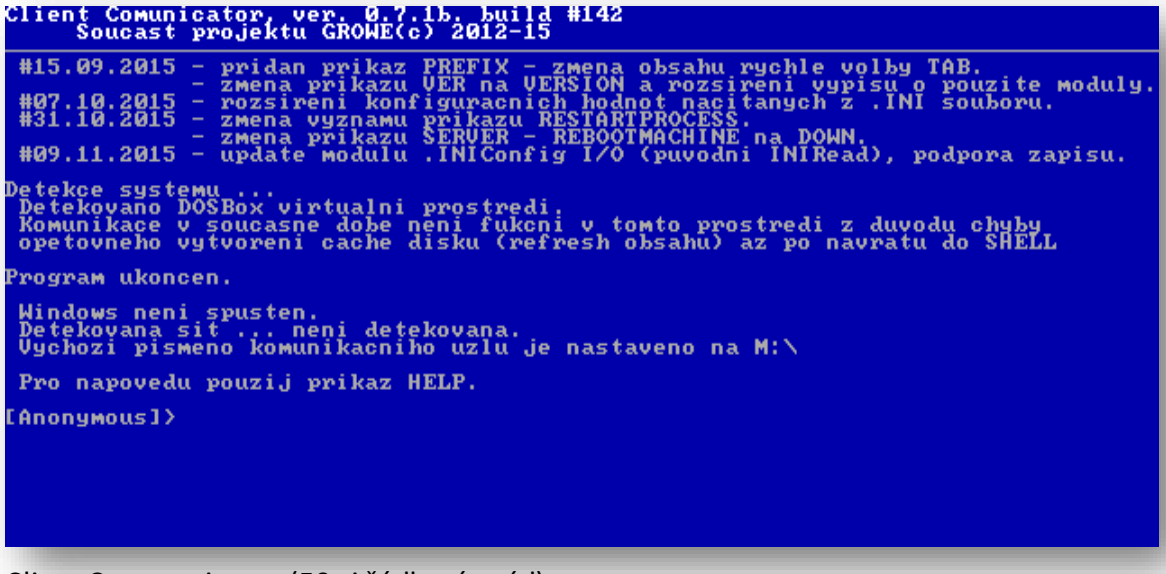
Digitální podpis je druh ochrany obsahu řídícího skriptu, kdy po jeho vytvoření se do hlavičky skriptu uloží tento podpis, obsahující CRC – cyklický redundantní součet obsahu. Cyklický redundantní součet, označovaný také CRC (zkratka z anglického názvu **Cyclic Redundancy Check**) je speciální hašovací funkce, používaná k detekci chyb během přenosu či ukládání dat. Výsledek tohoto kontrolního součtu je uložen v hlavičce, a pokud tento existuje (není ve výchozím stavu **00000000**), je programy pro práci se skriptem a hlavním řídícím programem čten, následně proveden nový kontrolní součet s aktuálními daty a výsledek je s tímto porovnán. Pokud se shoduje, je digitální podpis uznán jako platný. V opačném případě je zřejmé, že obsah byl po vložení podpisu pozměněn a digitální podpis je označen jako neplatný.

UPOZORNĚNÍ:
Současný proces tvorby a schvalování obsahu skriptu neumožňuje jednoprůchodový postup. Vložení digitálního podpisu nemá smysl hned po vytvoření skriptu, protože v něm dosud není vložena datumová řada (inicializace), její vložení zneplatní aktuální digitální podpis, který musí být následně aktualizován. Schvalovací proces při spuštění hlavního řídícího programu je nastaven tak, že nebude požadovat platný digitální podpis, pokud budete zaměňovat skript pomocí dálkové správy, bude platnost digitálního podpisu (z důvodu kryptování, přenosu a opětovného dekryptování) VŽDY vyžadována!!!

3. DÁLKOVÁ SPRÁVA

3.1. CLIENT COMMUNICATOR

Program **Client Comunicator** slouží ke vzdálené komunikaci se službou **SERVER Communicator**. Tato služba je svázaná programově s programem pro řízení procesu GROWE a umožňuje jeho dálkovou správu a monitoring. Pro komunikaci je využita síť MS-NET protokol TCP/IP s využitím technologie tzv. **Comm. Vertex** – komunikačního uzlu.



Client Communicator (50-ti řádkový mód)

Komunikační uzel je vytvořen jako **RAM** disk sdílený na straně **SERVERu**, o velikosti cca. **4 MB**. Na tento uzel se zasílají komunikační pakety, v nichž jsou předávány příkazy nebo žádosti o službu na straně **SERVER**. Komunikační paket je podrobně popsán v souboru **PACKET.TXT**, princip komunikace je v souboru **COMMUNIC.TXT**. Po odeslání příkazového paketu 'naslouchá' klient na komunikačním uzlu, dokud **SERVER** neodešle odpověď. Poté tuto zobrazí uživateli, popř. ji zpracuje do výstupních dat, které poté zobrazí.

Existuje několik úrovní komunikace. Nechráněna a chráněna, a tato se dále liší podle přístupových práv. Nechráněna komunikace umožňuje pouze zjištění existence komunikačního uzlu, stavu serveru ON Line/OFF Line, rychlost odezvy serveru a přihlášení pomoci uživatelského účtu. Teprve úspěšné přihlášení umožnuje využívat pokročilé příkazy, sloužící jednak ke správě služby SERVER a dále skrze tuto službu ovládat další, v tomto případě službu ŘÍZENÍ. Komunikace probíhá v krytované podobě, a to zašifrováním těla paketu. Po přihlášení k službě SERVER navíc dochází ke kontrole tzv. **CID – ClientID**. **ClientID** je řetězec 15-ti náhodně vygenerovaných znaků, který se vytvoří po startu programu a je platný po celou dobu jeho běhu. Po ukončení a opětovném spuštění se vytvoří nové **CID**. Toto **CID** slouží k ochraně komunikace po přihlášení, aby nemohlo dojit k převzetí vyšších oprávnění. Client odesílá **CID** v každém komunikačním paketu, Server kontroluje jeho shodu s **CID** použitým při přihlašování. Pokud **SERVER** nalezne neshodu, přeruší navázané spojení a provede vnitřní znovunastaveni do výchozího stavu. Každá taková zachycená neshoda se považuje za závažné narušení

ochrany komunikace, pokud se objeví opakovaně, předpokládá se, že probíhá pokus o převzetí nebo neoprávněného zásahu do komunikace třetí stranou a služba se trvale deaktivuje. Obdobně **SERVER** kontroluje vícenásobné neúspěšné přihlášení, kdy toto povazuje za tzv. **HAMMER ATTACK** (násilné prolomení založené na opakovaném zkoušení přihlašovacích údajů).

3.2. KONFIGURACE PROGRAMU

Pro správnou funkci dálkové správy je potřeba nakonfigurovat komunikační program i uzel. Konfigurační soubor programu **Client Communicator** používá formát **.INI**, který je až na obsah stejný, jako ten popsán v kapitole **1.8. Konfigurace programu**

```
[INI]
[Configuration]
colorfg=7
colorbg=1
vertexletter=m
prefixtab=server
clientlines=50
waitstateforserver=180
```

Popis jednotlivých konfiguračních proměnných hlavního programu:

colorfg – Barva pozadí, hodnota v rozsahu 0 až 15
colorbg – Barva popředí, hodnota v rozsahu 0 až 15
vertexletter – písmeno komunikačního uzlu
prefixtab – často se opakující příkaz, slovo, které se vloží do řádku po stisku klávesy TAB
clientlines – počet řádků, které zobrazuje program **Client Communicator**, hodnota je 25 nebo 50
waitstateforserver – časová prodleva, po kterou program čeká na odpověď serveru, výchozí je 180

3.3. PŘÍKAZY CLIENT&SERVER

Příkazová sada, která se používá pro obousměrnou komunikaci je dle specifikace SERVER COMMAND.

Podrobné informace o specifikaci:

Verze příkazové sady **SERVER COMMAND**, ver. **0.9.9. ALFA**
SYSTEM Client&Server GROWESYSTEM Communication
Značka příkazové sady: **SRV_Cmmnd_v0.99_D**

3.3.1. POPIS KOMUNIKAČNÍCH PŘÍKAZŮ

Tento soubor slouží zároveň jako nápovědný text, kdy při požadavku na server o zaslání informaci o specifikace příkazové sady, se uloží na komunikační uzel tento soubor.

Všechny příkazy jsou odesílány klientem na komunikační uzel serveru, kde jsou dále zpracovány. Příkazy jsou rozděleny do dvou sad – veřejné a chráněné, přístupné až po úspěšném přihlášení k službě server. Veřejné příkazy slouží k základnímu zjištění stavu služby serveru (stav-ONLINE/OFFLINE, test rychlosti odezvy, zvýšení uživatelských práv) a stavu komunikačního uzlu. Chráněné příkazy jsou dostupné až po úspěšném přihlášení k serveru, kdy cela přihlašovací procedura je na straně serveru a klient pouze přijímá textové výzvy k vložení už. jména a hesla a předává tyto získané údaje od uživatele. Pokud není uživatel přihlášen, je provedení všech chráněných příkazu odmítnuto serverem.

3.3.2. PŘÍKAZY NEVYŽADUJÍCÍ PŘIHLÁŠENÍ (NEKOMUNIKUJÍCÍ SE SLUŽBOU SERVER)

COMM - dotaz na konektivitu serveru
NETVER - informace o detekované verzi sítě LAN

3.3.3. PŘÍKAZY NEVYŽADUJÍCÍ PŘIHLÁŠENÍ

PING - ověření komunikace se serverem
SERVER LOGIN - příkaz pro přihlášení k serveru (získaní vyšších oprávnění pro možnost použiti chráněných příkazu)

3.3.4. PŘÍKAZY VYŽADUJÍCÍ PŘIHLÁŠENÍ (CHRÁNĚNÉ)

Server DATE	- vrací systémové datum serveru
Server TIME	- vrací systémový čas serveru
Server TEXT	- posle zprávu na obrazovku serveru
Server SYNCHRONIZE	- synchronizuje datum a čas serveru podle PC klienta
Server VERSION	- vrací informace o verzi serveru
Server INFO	- vrací podrobné informace o serveru
Server GETLOGFILES	- uloží .LOG soubory serveru na vertex
Server GETACTIVESCRIPT	- uloží aktivní .DBF skript na vertex
Server CHANGESCRIPT	- vymění aktivní skript za novy (dotaz na soubor)
Server RESTARTPROCESS	- restartuje proces GROWE
Server DOWN	- deaktivuje službu SERVER
Server KEYLOCK	- vypíná službu pro čtení klávesnice serveru
Server UNLOCK	- zapíná službu pro čtení klávesnice serveru
Server GETCOMMANDPACK	- odešle popis příkazové sady serveru na vertex
Server LOGOUT	- odhlášení od serveru (ztráta vyšších oprávnění)
Server AUTO_ON	- zapnuti auto řízení procesu GROWE
Server ATO_OFF	- vypnuti auto řízení procesu GROWE

Server GETTEMP	- vrátí poslední čtenou teplotu procesu GROWE
Server CONTROL X Y	- posle číslo X na Relé Y řídicí desky
Server TESTBOARD	- auto test řídící reléové desky

UŽIVATELSKÉ PŘÍKAZY (čísla 0-899)

písemná forma	číselná forma	datum vytvoření	platnost
Ping	#000	* 28. 12. 2014	**PLATNÝ**
Server Date	#001	* 01. 2015	**PLATNÝ**
Server Time	#002	* 01. 2015	**PLATNÝ**
Server Synchronize	#003	* 01. 2015	**PLATNÝ**
Server Version	#004	* 01. 2015	**PLATNÝ**
Server Info	#005	* 01. 2015	**PLATNÝ**
Server GetLogFile	#006	* 01. 2015	**PLATNÝ**
Server GetActiveAcript	#007	* 01. 2015	**PLATNÝ**
Server ChangeScript	#008	* 01. 2015	**PLATNÝ**
Server RestartProcess	#009	* 20. 06. 2015	**PLATNÝ**
Server RestartMachine	**#010**	*** 01. 2015**	**DEAKTIVOVÁN**
Server Down	#010	* 31. 10. 2015	**PLATNÝ**
Server KeyLock	#011	* 01. 2015	**PLATNÝ**
Server KeyUnLock	#012	* 01. 2015	**PLATNÝ**
Server GetCommandPack	#013	* 05. 2015	**PLATNÝ**
Server Auto ON	#014	* 10. 07. 2015	**PLATNÝ**
Server Auto OFF	#015	* 10. 07. 2015	**PLATNÝ**
Server GetTemp	#016	* 10. 07. 2015	**PLATNÝ**
Server Control X Y	#017	* 10. 07. 2015	**PLATNÝ**
Server TestBoard	#018	* 11. 07. 2015	**PLATNÝ**

SYSTÉMOVÉ PŘÍKAZY (rezervovaná část, čísla 900+)

písemná forma	číselná forma	datum vytvoření	platnost
Text	#900	* 04.2015	**PLATNÝ**
GetScreen	#910	* 11. 07. 2015	**PLATNÝ**
[test spojení]	#950	* 26. 06. 2015	**PLATNÝ**
Debug On/Off	**#980**	*** 11. 07. 2015**	**DEAKTIVOVÁN**
LogOut	#998	* 01. 02. 2015	**PLATNÝ**
LogIn	#999	* 01. 02. 2015	**PLATNÝ**

3.3.5. UŽIVATELSKÉ PŘÍKAZY SERVERU – PODROBNÝ POPIS

***Pozn**.: K zápisu: všechny argumenty uzavřené do závorek [] jsou nepovinné, argument Vertex musí být zapsán ve formátu písmena RAM disku + ":" -> M:*

3.3.5.1. PŘÍKAZ PING - #000

Ověření komunikace se serverem

Syntaxe příkazu:

PING vertex: [/C]

[/C] - cyklická operace, přerušení po ESC

(stejný příkaz se používá se pro udržování navázaného spojení se serverem)

NOPP vertex

Definice komunikace:

Klient odešle příkaz "000"

Server přijme příkazový paket, odešle odpověď "Communication OK".

Klient přijme paket odpovědi a zobrazí text.

Přepínač [/C] způsobí cyklické volaní, přerušení kl. ESC.

Příkaz NOPP cyklické volaní neumožnuje.

3.3.5.2. PŘÍKAZ SERVER DATE - #001

Vrací systémové datum PC serveru, vyžaduje přihlášení

Syntaxe příkazu:

Server DATE

Definice komunikace:

Klient odešle příkaz "001".

Server přijme příkaz, odešle systémové datum PC ve formátu "mm-dd-rrrr".

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.3. PŘÍKAZ SERVER TIME - #002

Vrací systémový čas PC serveru, vyžaduje přihlášení

Syntaxe příkazu:

Server TIME

Definice komunikace:

Klient odešle příkaz "002" .

Server přijme příkaz, odešle systémový čas PC ve 24 hod. formátu "hh-mm-ss".

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.4. PŘÍKAZ SERVER SYNCHRONIZE - #003

Synchronizuje systémový datum a čas serveru podle času klienta

Syntaxe příkazu:

Server SYNCHRONIZE

Definice komunikace:

Klient odešle příkaz "003" + systémový datum PC ve formátu "mm-dd-rrrr" + systémový čas PC ve 24 hod. formátu "hh-mm-ss".

Server přijme příkaz, aktualizuje sys. datum a čas dle příchozích informací a odešle textovou zprávu "Sync OK".

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.5. PŘÍKAZ SERVER VERSION - #004

Vrací informaci o verzi serveru

Syntaxe příkazu:

Server VERSION

Definice komunikace:

Klient odešle příkaz "004".

Server přijme příkaz, odešle textovou zprávu obsahující informaci o verzi služby server.

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.6. PŘÍKAZ SERVER INFO - #005

Server uloží podrobné informace o svém běhu na vertex v .LOG souboru

Syntaxe příkazu:

Server INFO

Definice komunikace:

Klient odešle příkaz "005".

Server přijme příkaz, zkopíruje soubor SERVER.LOG z lokálního adresáře na vertex a odešle textovou zprávu "Save File To Comm Vertex"

Klient přijme paket odpovědi a na dotaz zobrazí soubor odpovědi.

Klient na dotaz smaže .LOG soubor odpovědi z vertexu. Při opakovaném příkazu se .LOG soubor na vertexu přepisuje.

3.3.5.7. PŘÍKAZ SERVER GETLOGFILE - #006

Server uloží na vertex všechny nalezené .LOG soubory z lokálního adresáře

Syntaxe příkazu:

Server GETLOGFILES

Definice komunikace:

Klient odešle příkaz "006".

Server přijme příkaz, zkopíruje soubory *.LOG z lokálního adresáře na vertex a odešle textovou zprávu "Send xLOGFilesOK", kdy x označuje počet odeslaných souboru.LOG

Klient přijme paket odpovědi a zobrazí text odpovědi.

Soubory .LOG jsou stále uloženy na vertexu, pokud je příkaz volán opakovaně jsou soubory přepisovány (aktualizovány).

3.3.5.8. PŘÍKAZ SERVER GETACTIVEACRIPT - #007

Uloží aktivní řídicí .DBF skript na vertex

Syntaxe příkazu:

Server GETACTIVESCRIPT

Definice komunikace:

Klient odešle příkaz "007".

Server přijme příkaz, nalezne (podle informaci z řídicího systému) aktivní řídicí skript .DBF z lokálního adresáře, tento zkopíruje na vertex a odešle textovou zprávu "SendActivScr".

Poté server odešle textovou zprávu o výsledku operace restartu procesu.

Soubor skriptu .DBF je stále uložen na vertexu, pokud je příkaz volán opakovaně, je soubor přepisován.

Pokud server nenalezne soubor skriptu .DBF v aktivním adresáři odešle textovou zprávu "Not Available Script".

3.3.5.9. PŘÍKAZ SERVER CHANGESCRIPT - #008

Zamění aktivní řídicí .DBF skript na serveru

Syntaxe příkazu:

Server CHANGESCRIPT

Definice komunikace:

Klient zobrazí dotaz na název soubor, kterým nahradí původní skript na serveru.

Po vybrání souboru je provedena jeho kontrola, zda je skript validní (kontrola platnosti digitálního podpisu). Pokud není kontrola ukončena s chybou, klient zkopíruje skript na vertex, odešle příkaz "008"+ název skriptu, server přijme paket, přesune skript z vertexu do lokálního adresáře a předá pokyn řídicímu programu pro záměnu skriptu v procesu ŘÍZENÍ.

Poté server odešle textovou zprávu o výsledku operace záměny skriptu a restartuje se.

3.3.5.10. PŘÍKAZ SERVER RESTARTPROCESS - #009

Restartuje řídicí proces GROWE

Syntaxe příkazu:

Server RESTARTPROCESS

Definice komunikace:

Klient odešle příkaz "009"

Server přijme příkaz, předá požadavek o restart procesu ŘÍZENÍ.

Pote server odešle textovou zprávu o výsledku operace restartu procesu "System SERVER now restarting..."

SERVER provede interní restart, odpojí připojenou relaci a uvede se do výchozího stavu

Klient přijme paket s odpovědí, zobrazí text odpovědi a provede automatické odhlášení

3.3.5.11. PŘÍKAZ SERVER DOWN - #010

Deaktivuje službu SERVER (při běhu procesu ŘÍZENÍ)

Syntaxe příkazu:

Server DOWN

Definice komunikace:

Klient odešle příkaz "010"

Server přijme příkaz, předá požadavek o deaktivaci služby nadřazenému programu

Poté server odešle textovou zprávu o přijetí požadavku o deaktivaci služby.

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.12. PŘÍKAZ SERVER KEYLOCK - #011

Zamyká klávesnici serveru (SERVER odpojí službu čtení klávesnice)

Syntaxe příkazu:

Server KEYLOCK

Definice komunikace:

Klient odešle příkaz "011".

Server přijme příkaz, provede zamčení klávesnice a vrátí textovou zprávu "Server Keyboard Lock"

Klient přijme paket s odpovědí a zobrazí text odpovědi.

3.3.5.13. PŘÍKAZ SERVER KEYUNLOCK - #012

Odemyká klávesnici serveru (SERVER připojí službu čtení klávesnice)

Syntaxe příkazu:

Server KEYUNLOCK

Definice komunikace:

Klient odešle příkaz "012".

Server přijme příkaz, provede odemčení klávesnice a vrátí textovou zprávu "Server Keyboard Unlock"

Klient přijme paket odpovědi a zobrazí text odpovědi.

3.3.5.14. PŘÍKAZ SERVER GETCOMMANDPACK - #013

Server vyhledá v domácím adresáři soubor SRVCMMNT.TXT a uloží ho na vertex

Syntaxe příkazu:

Server GETCOMMANDPACK

Definice komunikace:

Klient odešle příkaz "013".

Server přijme příkaz, provede kontrolu existence souboru, pokud existuje, zkopíruje ho na vertex a vrátí textovou zprávu o úspěšném dokončení operace "Server Send Command Pack to Vertex", jinak odešle textovou zprávu o neúspěchu "Command Pack not Found".

3.3.5.15. PŘÍKAZ SERVER AUTO ON - #014

Příkaz předá procesu ŘÍZENÍ požadavek o zapnutí automatické regulace dle řídicího skriptu

Syntaxe příkazu:

Server AUTO ON

Definice komunikace:

Klient odešle příkaz "014".

Server přijme příkaz, zapne službu automatického řízení procesu GROWE. Vrátí textovou zprávu o úspěšném dokončení operace "Server Auto On Ready", jinak odešle textovou zprávu o neúspěchu "Server Auto ON Fail".

3.3.5.16. PŘÍKAZ SERVER AUTO OFF - #015

Příkaz předá procesu ŘÍZENÍ požadavek o vypnutí automatické regulace dle řídicího skriptu

Syntaxe příkazu:

Server AUTO OFF

Definice komunikace:

Klient odešle příkaz "015".

Server přijme příkaz, vypne službu automatického řízení procesu GROWE. Vrátí textovou zprávu o úspěšném dokončení operace "Server Automatic Control is OFF", jinak odešle textovou zprávu o neúspěchu "Server Auto OFF Fail".

3.3.5.17. PŘÍKAZ SERVER GETTEMP - #016

Vrátí hodnotu poslední měřené teploty v procesu GROWE

Syntaxe příkazu:

Server GETTEMP

Definice komunikace:

Klient odešle příkaz "016".

Server přijme příkaz a vrátí textovou zprávu s informací o aktuální platné teplotě. Jinak odešle textovou zprávu o chybě při čtení teploty.

3.3.5.18. PŘÍKAZ SERVER CONTROL X Y - #017

Příkaz předá procesu ŘÍZENÍ požadavek na reset, změnu stavu nebo vrátí aktuální stav všech relé.

Syntaxe příkazu:

Server CONTROL X Y

Definice komunikace:

Klient odešle příkaz "017", dále hodnoty X a Y.

Server přijme příkaz, předá procesu GROWE požadavek o změnu na relé X stav Y. Rozsah hodnoty relé X je 0–9, kdy 0 provede reset všech relé do stavu VYPNUTO, hodnota 9 vrátí pouze stav všech relé, bez toho, že by byly provedeny změny. V obou těchto případech je druhá hodnota Y ignorována. V případě použití hodnoty 1 až 8 tím se označí, na kterém relé má být provedena změna. V tomto případě druhá hodnota označuje, stav, na který má být současný změněn. Hodnota 0 znamená vypnuto, 1 zapnuto a 2 změna současného stavu na opačný. Jiné hodnoty jsou ignorovány.

UPOZORNĚNÍ:
Před tímto příkazem je nutno vypnout automatické řízení, příkazem SERVER AUTO_OFF, protože každá další událost dle řídicího skriptu může změnit požadovanou hodnotu na původní v něm uvedenou. K těmto kontrolám stavu dochází min. 1x za sec. a může dojit k tzv. procvaknutí relé, kdy se krátkodobě přeruší proud, což může narušit funkcionalitu některých spotřebičů v procesu GROWE.

3.3.5.19. PŘÍKAZ SERVER TESTBOARD - #018

Příkaz předá procesu ŘÍZENÍ požadavek na auto test řídící reléové desky, vrátí výsledek testu.

Syntaxe příkazu:

Server TESTBOARD

Definice komunikace:

Klient odešle příkaz "018".

Server přijme příkaz, předá procesu GROWE požadavek provedení auto testu. Při jeho průběhu jsou postupně spínána jednotlivá relé a kontroluje se vždy jejich stav. Poté je provedena kontrola zákazu souběhu sepnutých relé 4 a 5. Po ukončení testu SERVER vrátí informaci o výsledku testu.

3.3.6. SYSTÉMOVÉ PŘÍKAZY SERVERU

3.3.6.1. PŘÍKAZ SERVER TEXT - #900

Odeslaní textového řetězce a jeho zobrazeni na obrazovce serveru

Syntaxe příkazu:

SERVER TEXT

Definice komunikace:

Klient odešle příkaz "900"+ text zprávy

Server přijme příkaz, zobrazí text zprávy na obrazovku serveru a vrátí textovou zprávu "Incoming message accepted." v paketu odpovědi.

Klient přijme paket a zobrazí text odpovědi.

3.3.6.2. PŘÍKAZ (TEST SPOJENI) - #950

Automaticky příkaz, nelze vyvolat pomoci příkazu v SHELLu klienta, spouští automaticky klient po úspěšném přihlášení k serveru, v cyklu co 10 sec.

Syntaxe příkazu:

Syntaxe zápisu příkazu není definována – automatický příkaz systému

Definice komunikace:

Pro tento speciální příkaz je zkrácena obdoba čekání na odpověď a to na 1/3 této doby, pokud server neodpoví, předpokládá se, ze spojení bylo ztraceno a klient se uvede do stavu nepřipojeno a změní PROMPT na <[Anonymous]>

3.3.6.3. PŘÍKAZ SERVER LOGOUT - #998

odhlášení od serveru (ukončení konekce, ztráta vyšších oprávnění)

Syntaxe příkazu:

Server LOGOUT

Definice komunikace:

Klient odešle příkaz "998".

Server přijme příkaz, provede odhlášení klienta od serveru a vrátí textovou zprávu "User Xxxxx disconnect from server.", kdy Xxxxx je uživatelské jméno odhlašovaného uživatele.

Klient přijme paket a zobrazí text odpovědi a změní PROMPT na [Anonymous]>

3.3.6.4. PŘÍKAZ SERVER LOGIN - #999

Přihlášení uživatele k serveru (navázaní konekce, získaní vyšších oprávnění)

Syntaxe příkazu:

Server LOGIN

Definice komunikace:

Klient zobrazí dotaz na název písmene vertexu, dojde o ověření platnosti vertexu, klient odešle příkaz:

"999".+CCVer(Verze klienta)+CID(ClientID)

Server přijme příkaz, uloží CCVer a CID do paměti a nastaví příznak pro probíhající proces LOGIN. Server odešle dotaz na uživatelské jméno formou textové zprávy "User name:" v paketu odpovědi. Klient přijme paket, zobrazí text dotazu a čeká na zadaní už. jména, poté jej odešle v paketu serveru. Server přijme paket, uloží do paměti už. jméno a odešle dotaz na heslo formou textové zprávy "Password:" v paketu. Klient přijme paket, zobrazí text dotazu a čeká na zadání hesla, zadávaní probíhá v režimu skrytých znaků na obrazovce, poté jej odešle v paketu serveru. Server přijme paket, uloží do paměti heslo, zkontroluje platnost podpory klienta CCVer, shoda CID, dekryptuje LOGIN skript PASSWORD.TXT, zkontroluje platnost už. jména a hesla. Při všech shodách nastaví příznak pro přihlášení, odešle text odpovědi "User Xxxxx logged sucessfully.", kdy Xxxxx je uživatelské jméno přihlašovaného uživatele. Dále zpřístupní chráněné příkazy serveru, kdy při použití těchto příkazů jsou dále kontrolovány informace CCVer a CID aby nemohlo dojit k převzetí navázané komunikace s jiným klientem nebo cizím programem a s ní neoprávněnému navýšení práv k serveru. Po úspěšném přihlášení změní klient PROMPT na [Logged]>. Pokud se přihlašovací údaje (už. jméno nebo heslo) neshoduji, server odmítne konekci a odešle odpověď "Login fail."

Pokud se neshoduje CCVer, server odešle odpověď "Wrong Client or unsupported Client version. Login fail."

Pokud se neshoduje CID, což je závažná bezpečnostní chyba, která může nastat např. po přihlášení klientem k serveru, pote restartu klienta a pokusu užití statusu přihlášení (zvýšeni práv) k serveru. Server uloží do RUN.LOG souboru hlášeni o pokusu o narušení bezpečnosti "Pokus o neautorizovaný přístup k serveru " a ukončí veškeré konekce a restartuje se. Po restartu se uvede do výchozího stavu.

Na stranu klienta se neodesílají o tomto žádná chybová hlášení, aby nedocházelo k nechtěnému napomáhání při pokusu o násilné prolomení prozrazením reakcí systému.

4. KONFIGURACE PROSTŘEDÍ

4.1. DOPORUČENÝ HARDWARE A SOFTWARE

V současné době rozvoje počítačů v podstatě vyhovuje hardwarovým požadavkům každý počítač, na němž běží OS Windows 98. Mezi povinné komponenty patří **MAX**. jedno-jádrový procesor, alespoň 64 MB RAM, volný prostor na disku cca 100 MB, 1x COM a 1x LPT port, síťová karta TCP/IP.

UPOZORNĚNÍ:
Operační systém MS-DOS ani aplikace Projektu GROWE nepracují na více jádrových procesorech. I když je někdy možné, modifikací systému BIOS deaktivovat určitá jádra, nebo vypnout funkci Hyper-threading, obecně nelze tento hardware doporučit. Při provádění testů na více jádrových procesorech nebyl běh programů zcela bezproblémový a projevovalo se občasné mrznutí systému.

4.2. KONFIGURACE OS

Pro správný běh programu a všech jeho služeb je potřebná specifická konfigurace operačního systému (OS). Základem je systém Windows 95/98/98SE/ME. Doporučený (a zde dále popisovaný) systém je Windows 98 SE, obsahující síťovou podporu TCP/IP, spuštěnou službu pro vytvoření RAM Disku a jeho sdílení v rámci sítě pro plný přístup (zápis a čtení) o velikost cca. 5-10 MB. Tato velikost je zcela dostačující pro běh programu, nicméně při intenzivním používání dálkové správy je třeba dbát na "úklid" disku, tj. mazání dat uložených na něm. Lze smazat téměř celý obsah disku, mimo klíčových souborů, které jsou chráněny. Tyto soubory se odstraní automaticky při ukončení hlavního programu.

Pro správnou konfiguraci OS je třeba modifikovat následující soubory:

- AUTOEXEC.BAT
- MSDOS.SYS
- SYSTEM.INI

```
@ECHO OFF
PROMPT $P$G
PATH
C:\;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\GROWE;C:\GROWE\RIZENI;C:\GROWE\
RIZENI\BIN;C:\GROWE\RIZENI\MAN;C:\GROWE\RIZENI\UTIL
SET LIB=C:\VBDOS\LIB;%LIB%
SET INCLUDE=C:\VBDOS\INC;%INCLUDE%
SET HELPFILES=C:\VBDOS\HLP;%HELPFILES%
SET TEMP=C:\TEMP
C:\XMSDSK\XMSDSK 10240 m: /C1 /T /Y
```

Soubor AUTOEXEC.BAT - umístění v kořenovém adresáři disku

UPOZORNĚNÍ:
Pro maximální využitelnost operační paměti je potřeba, aby byly načteny pouze nezbytné ovladače OS. Zbytečně spuštěné služby a ovladače mohou být zdrojem nestability OS, která je v automatickém procesu nežádoucí.

V pokročilé konfiguraci OS je možno nastavit a (doporučuje se) vyřadit z běhu OS i systémový SHELL, proces EXPLORER.EXE. Při startu nebo restartu PC pak dojde k automatickému spuštění hlavního řídícího programu. Pokud tedy dojde např. k výpadku napájení, po jeho obnově se PC opětovně spustí a proces pokračuje tam, kde byl přerušen.

Údaj **shell** je klíčový, pro svou rozsáhlost je zde uvedena pouze úvodní část souboru, ostatní údaje zde uvedené se nemusí shodovat s obsahem souboru ve vašem počítači. Je třeba si uvědomit, že soubor

```
[Paths]
WinDir=C:\WINDOWS
WinBootDir=C:\WINDOWS
HostWinBootDrv=C
[Options]
BootMulti=0
BootGUI=1
BootKeys=0
BootMenu=0
BootMenuDefault=1
BootMenuDelay=1
AutoScan=0
LoadTop=1
Logo=1
SystemReg=0
DoubleBuffer=0
WinVer=4.10.2222
```

Soubor MSDOS.SYS - umístění v kořenovém adresáři disku

obsahuje konfiguraci OS a ta může být na různých počítačích odlišná.

```
[boot]
oemfonts.fon=vgaoem.fon
shell=explorer.exe                    <- záměna za   shell=c:\growe\rizeni\bin\mini.exe
system.drv=system.drv                                pro automatické spuštění při startu OS
drivers=mmsystem.dll power.drv
user.exe=user.exe
gdi.exe=gdi.exe
```

Soubor SYSTEM.INI - umístění – adresář obsahující systém Windows

4.3. KONFIGURACE VERTEXU – KOMUNIKAČNÍHO UZLU

Komunikační uzel, je RAM disk o optimální velikosti 4-10 MB, který je potřeba, aby byl nastaven jako sdílený, v sítí MS-Net pro plný přístup (zápis a čtení). Doporučuje se, aby tento disk byl vytvořen na počítači, kde běží služba SERVER. I když je možná situace, kdy se komunikační uzel nachází na straně klienta, ale tato konfigurace se DŮRAZNĚ nedoporučuje, protože skýtá několik úskalí. Program požaduje při spuštěné službě SERVER neustálý přístup ke komunikačnímu uzlu, proto by musel neustále

```
DEVICEHIGH=C:\WINDOWS\HIMEM.SYS /V
NUMLOCK=ON
LASTDRIVE=Z
FILES=50
DOS=HIGH,UMB
STACKS=9,256
DEVICEHIGH=C:\WINDOWS\EMM386.EXE RAM HIGHSCAN AUTO
SHELL=C:\COMMAND.COM /E:1024 /P
DEVICE=C:\WINDOWS\IFSHLP.SYS
DEVICEHIGH=C:\WINDOWS\RAMDRIVE.SYS 10240 512 1024 /E
```

Soubor CONFIG.SYS - umístění v kořenovém adresáři

běžet i klientský počítač, na němž by byl vytvořen. Tento navíc by musel běžet bez možnosti přechodu do úsporného režimu, který počítač by způsobil odpojení od sítě a znepřístupnění komunikačního uzlu. Toto odebrání uzlu při běhu služby SERVER, by způsobilo, že služba by tuto událost zachytila a automaticky by zastavila svoji činnost a ohlásila chybový stav. Restart služby je samozřejmě možný, ale je nutné ho provést stiskem Alt+R na řídícím počítači, čímž dálková správa ztrácí na logice svého použití.

Pro načtení ovladače a vytvoření RAM disku je třeba do souboru **CONFIG.SYS** řádek

DEVICEHIGH=C:\WINDOWS\RAMDRIVE.SYS 10240 512 1024 /E

nebo do souboru **AUTOEXEC.BAT** přidat řádek:

C:\XMSDSK\XMSDSK 10240 m: /C1 /T /Y[3]

UPOZORNĚNÍ:
Prosím zkontrolujete, zda se se soubor RAMDRIVE.SYS opravdu nachází v adresáři, kam je uvedená cesta. V případě, že OS nenajde uvedený ovladač, ohlásí chybu a RAM disk se nevytvoří.

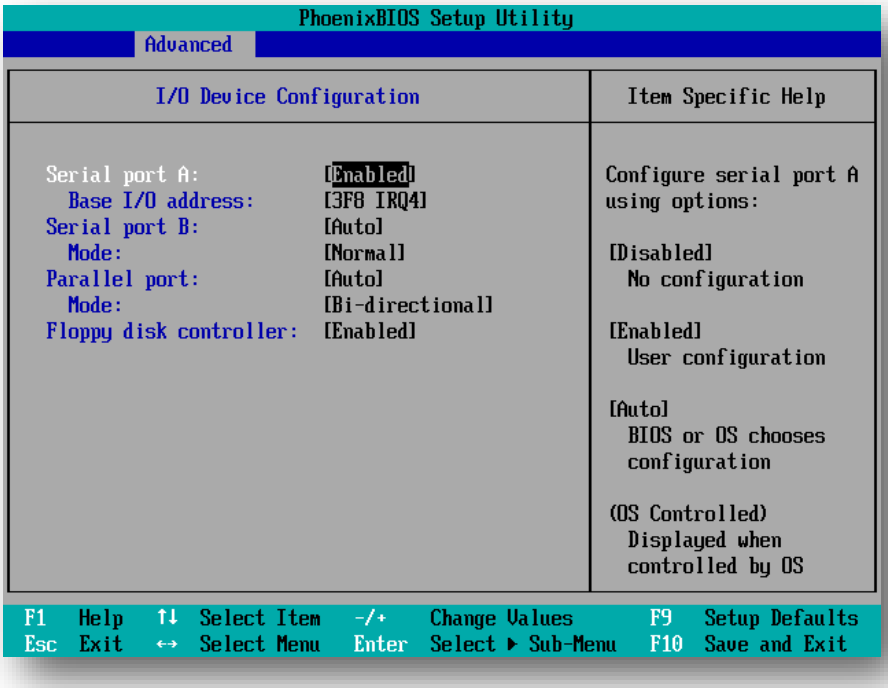
Sdílení RAM disku v síti MS-Lan je třeba nakonfigurovat tak, aby do něj měly práva čtení a zápisu obě strany, tedy PC se službou SERVER a další PC s programem **Client Commnunicator** . Doporučený OS, což je v tomto případě **MS Windows 98 SE** umožňuje zabezpečit přístup heslem, což i doporučujeme, pro ochranu před nežádoucím zásahem do něj. Veškerý obsah procházející uzlem je kryptován, přesto

[3] Pro vytvoření RAM disku slouží utilita **ReSizeable RAMDisk version 2.09**, která je ke stažení na WEBových stránkách projektu GROWE

ochranu heslem nelze podceňovat, pokud by došlo např. k zaplnění uzlu po nakopírování nějakého obsahu do něj, by opětovně zasáhla ochrana služby SERVER a problém by „vyřešila“ svým zastavením.

POZNÁMKA:
Požadavek na hlavní řídící program je, že si musí sám automaticky poradit s neočekávanými situacemi a to způsobem, že odpojí problematické části (např. služby) aby zachoval běh své hlavní funkce, tj. řízení procesu GROWE. Pokud by docházelo k dalším a dalším výpadkům v programu, bude se program pokoušet o regeneraci svým restartem. Pokud ani toto neuspěje, program přepne proces GROWE do udržovacího režimu vyhlásí stav nouze (např. zvukovou signalizací), uloží veškerý obsah paměti a dat do souboru .LOG a vypne se.

4.4. NASTAVENÍ COM PORTU V BIOS



Náhled nastavení COM portu v BIOS

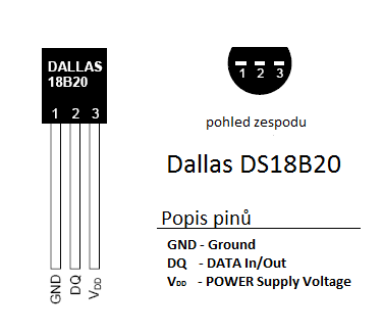
Nastavení parametrů pro konfiguraci COM (seriového) portu jsou:

Serial Port	[AUTO] nebo [Enabled]
Base I/O Address	[3F8, IRQ4]

POZNÁMKA:
*Volba **[AUTO]** nastaví všechny parametry automaticky, po této volbě se další parametry nastavení znepřístupní nebo se skryjí.*

4.5. TEPLOTNÍ ČIDLO

Čidlo DS18B20 vyrábí firma Maxim/Dallas. V sortimentu nalézt teploměry, A/D převodníky, paměti, čidla vlhkosti, měřiče napětí a další periferie, které komunikují po univerzální sběrnici 1-Wire® Bus.



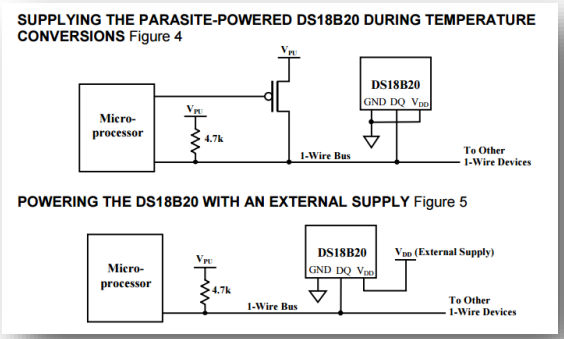
Pohled na čidlo a rozmístění kontaktů

Použitá teplotní čidla, DS18B20 patří z hlediska parametrů mezi nejlepší, a přitom cenově dostupné. Je velmi přesné (absolutní přesnost lepší než 0,5 °C), lze jej připojit pouze dvěma "dráty" a obsahuje řadu dalších funkcí, např. programovatelnou přesnost převodníku 9-12 bitů.

Part Number	Interface	Accuracy (±°C)	Supply Voltage Range (V)	Parasite Power	Operating Temp. Range (°C)	Temp. Thresh. (°C)	Meas. Res. (Bits)	Package
DS18B20	1-Wire®	0.5	3.0 to 5.5	Available	-55 to +125	1, Prog. NV	9 - 12	3/TO-92 8/µMAX 8/SO.150

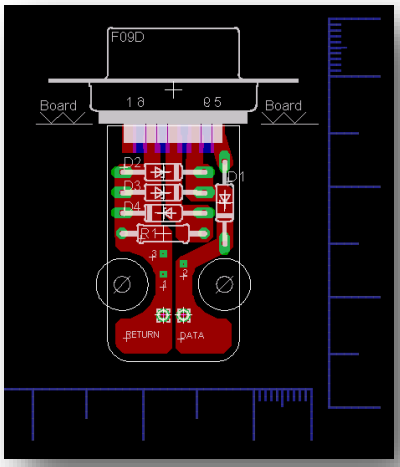
Základní informace o čidlu

Čidlo DS18B20 je napájeno přímo ze sběrnice, což znamená, že jsou vývody čidla 1 a 3 připojeny na RETURN, vývod 2 jsou DATA. Propojovací kabel pak stačí pouze dvouvodičový, připojený CINCH konektorem. Délka 1-Wire® sběrnice může být až několik desítek metrů.



Náhled diagramu zapojení napájení čidla

Takovéto napájení se nazývá parasite-power, prakticky to znamená, že pro napájení čidla i posílání informace o naměřené teplotě „chodí“ po jednom drátu. Čidlo se nejdříve jakoby nabije, pak se na přesně stanovenou dobu vyzkratuje a pokud se dodrží stanovená doba, která je v desítkách milisekund, čidlo vrátí informaci o aktuální teplotě. V praxi je celá problematika složitější, datasheed čidla v rozsahu cca 20 stran (soubor **ds18b20-datasheet.pdf**) je přiložen v adresáři s dokumentací.



Pohled na plošný spoj (ze strany součástek)

Použité součástky:

Odpor	R1 - 1k5
Dioda	D1 - BZX83 3,9V
Dioda	D2 - BZX83 6,2V
Dioda	D3, D4 - 1N5818
Teplotní čidlo	IC1 - DS18B20
Konektor	CAN9-Z
Krytka na konektor	CAN9
Zásuvka	CINCH SCJ-0359

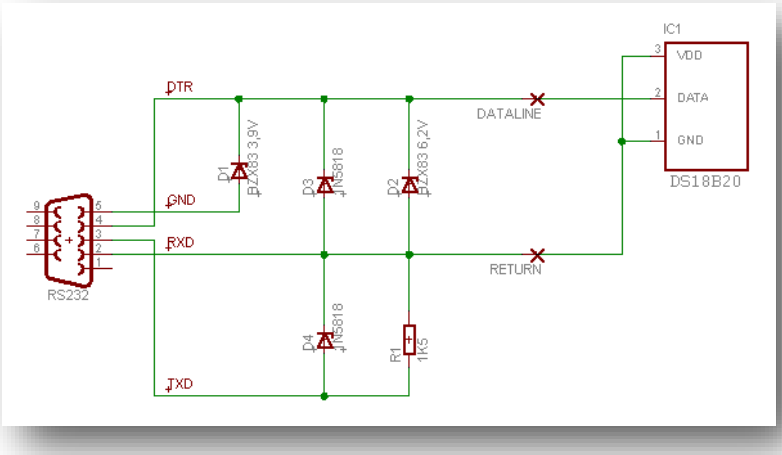


Schéma zapojení čidla

4.6. TEPLOTNÍ ČIDLO – SOFTWARE

O komunikaci s teplotním čidlem se stará program **DigiTemp** soubor **DIGITEMP.EXE**, který je spuštěn jako nezávislý proces, který cyklicky 1x za minutu čte aktuální teplotu z čidla nebo čidel a tu ukládá do souboru **TEMP.LOG**. Pro bezchybnou funkci tohoto procesu je třeba mít vytvořený konfigurační soubor DIGITEMP.CFG, který musí být umístěn ve stejném adresáři jako program. Protože ruční vytvoření tohoto souboru není právě snadné, je třeba dodržen striktní zápis, včetně formátovacích znaků. V opačném případě program ignoruje tuto konfiguraci a nastaví se do výchozího stavu, kdy neprovádí zápis do souboru, ale na obrazovku. Hlavní řídící program očekává aktuální teplotu v souboru **TEMP.LOG**, pokud nalezne 2x nebo víckrát poslední záznam neaktualizovaný, vyhlásí chybu čtení teploty.

```
08.02.2016 19:35:28 Sensor_0 C° 28.75
08.02.2016 19:36:27 Sensor_0 C° 28.75
08.02.2016 19:37:26 Sensor_0 C° 28.75
08.02.2016 19:38:25 Sensor_0 C° 28.50
08.02.2016 19:39:24 Sensor_0 C° 28.50
08.02.2016 19:40:23 Sensor_0 C° 28.75
08.02.2016 19:41:22 Sensor_0 C° 28.75
08.02.2016 19:42:21 Sensor_0 C° 28.75
08.02.2016 19:43:20 Sensor_0 C° 28.75
08.02.2016 19:44:19 Sensor_0 C° 28.75
08.02.2016 19:45:18 Sensor_0 C° 28.75
08.02.2016 19:46:17 Sensor_0 C° 28.75
08.02.2016 19:47:16 Sensor_0 C° 29.00
08.02.2016 19:48:15 Sensor_0 C° 28.75
08.02.2016 19:49:14 Sensor_0 C° 28.75
08.02.2016 19:50:13 Sensor_0 C° 28.75
08.02.2016 19:51:12 Sensor  0 C° 28.75
```

Záznamy o čtení teploty v souboru TEMP.LOG

Soubor **TEMP.LOG** obsahuje záznamy o čtení teploty ve formátu DD.MM.RRRR HH:MM:SS, název sensoru, znak stupeň Celsia, naměřená hodnota. Poslední záznam na konci souboru je vždy ten nejaktuálnější.

```
Turning off all DS2409 Couplers
.
Searching the 1-Wire LAN
28984EB401000071 : DS18B20 Temperature Sensor
ROM #0 : 28984EB401000071
```

Záznam o inicializaci čidla, soubor DIGITEMP.NFO

Soubor **DIGITEMP.CFG** obsahuje informace, které odpovědělo teplotní čidlo nebo čidla (max. 2) při svém prvotním oslovení.

Soubor **DIGITEMP.TXT** obsahuje rychlou nápovědu, která se vypíše při spuštění programu bez jakýchkoli parametrů. Protože je text rozsáhlejší, než je velikost jedné obrazovky, dochází při výpisu textu k odřádkování textu mimo. Proto je v celkové podobě uložen v souboru v textové formě.

INFORMACE:
Program je určen pro Windows 95-98, ve Windows 2000 a vyšších OS je již problematické jej spustit, pokud není spuštěn v režimu kompatibility.

```
DigiTemp v1.7                Copyright 1996-2002 by Brian C. Lane
All Rights Reserved - http://www.brianlane.com

Usage: digitemp [-i -a -t -w] [-s -d -l -r -v -n -c -o]
-i                           Initalize digitemp.cfg file
-w                           Walk the full device tree
-a                           Read all Sensors
-s1                          Set serial port
-r500                        Read delay in mS
-v                           Verbose output
-l/var/log/temperature       Send output to logfile
-t0                  Read Sensor #
-d5                  Delay between samples (in sec.)
-c/path/configfile           Configuration file
-q                           Quiet, no copyright banner
-n50                 Number of times to repeat
-o2                  Output format for logfile
-o"output format string"     See description below

Logfile formats:     1 = One line per sensor, time, C, F (default)
2 = One line per sample, elapsed time, temperature in C
3 = Same as #2, except temperature is in F
#2 and #3 have the data seperated by tabs, suitable for import
into a spreadsheet or other graphing software.
```

Možnosti spuštění programu DIGITEMP.EXE – nápovědný text

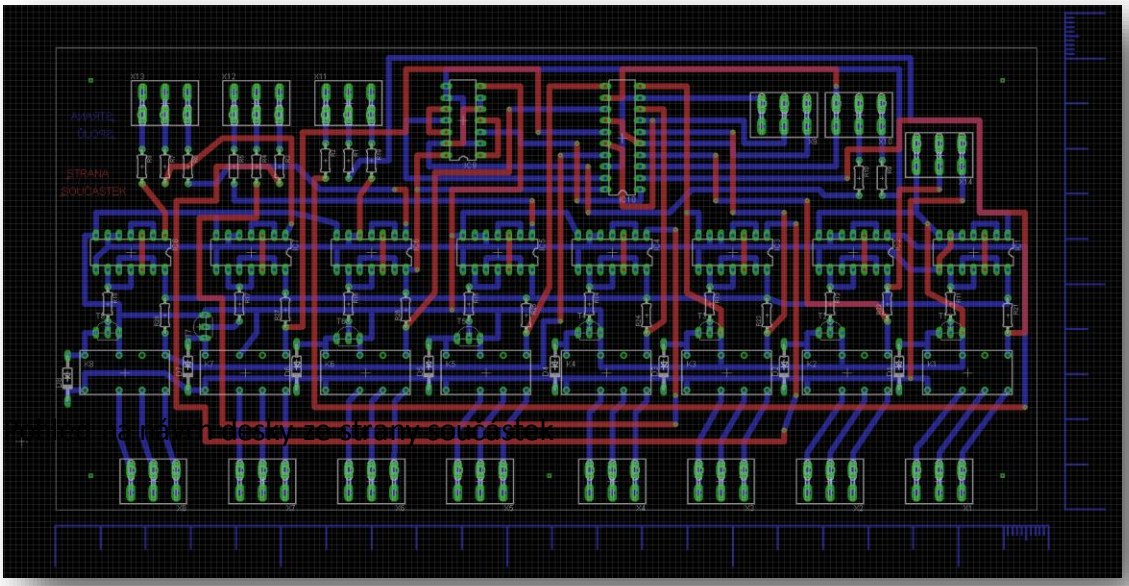
UPOZORNĚNÍ:
Klíčový je konfigurační soubor DIGITEMP.CFG, který společně se souborem DIGITEMP.EXE a spouštěcím souborem DIGITEMP.BAT musí být nakopírován do adresáře obsahující hlavní řídící program.

Tip:
Pro jednoduché vytvoření konfiguračních souborů slouží program popsán v kapitole **6.5. DigiTemp Config File Creator**.

5. ŘÍDÍCÍ RELÉOVÁ DESKA

5.1. POPIS ŘÍDÍCÍ DESKY

Řídící deska je klíčová komponenta pro celý proces **GROWE**, protože je v podstatě výkonným hardwarovým modulem, provádějící příkazy počítače. Je připojena pomocí **LPT** portu, komunikuje tedy paralelně, tz., že dostává příkazy pro všechna spínací relé „najednou“. Deska je napájená 5 a 12 V, z interního zdroje.



Návrh plošných spojů řídící desky oboustranná (pohled ze strany součástek)

Relé a kontakty 240 V ~ jsou na vzdálenější straně. Počítají se z levé strany na pravou - 1 až 8.

Zapojení napájení a svorkovnic do kabelu LPT:

1. svorkovnice z levé strany - napájení:
   - leva zdířka +12V
   - prostřední zdířka +5V
   - pravá zdířka 0V - zem napájení, zem z konektoru
   - Tato napájecí napětí jsou použita ze zdroje PC.
   - **piny 19, 20 nespojovat se stíněním kabelu!!!**

2. svorkovnice
   - levý kontakt řízeni čtení - konektor LPT, pin 17 **SELIN**
   - střední řízení čtení - konektor LPT, pin 14 **AUTO**
   - pravý čtení stavu - konektor LPT, pin 13 **SELECT**

3. svorkovnice

levý kontakt	čtení stavu	- konektor LPT, pin 12	**PE**
střední	čtení stavu	- konektor LPT, pin 10	**ACK**
pravý	čtení stavu	- konektor LPT, pin 11	**BUSY**

4. svorkovnice

levý kontakt	řízení zápisu DAT	- konektor LPT, pin 1	**STROBE**
střední	DATA - bit 7	- konektor LPT, pin 9	**DATA-7**
pravý	DATA - bit 6	- konektor LPT, pin 8	**DATA-6**

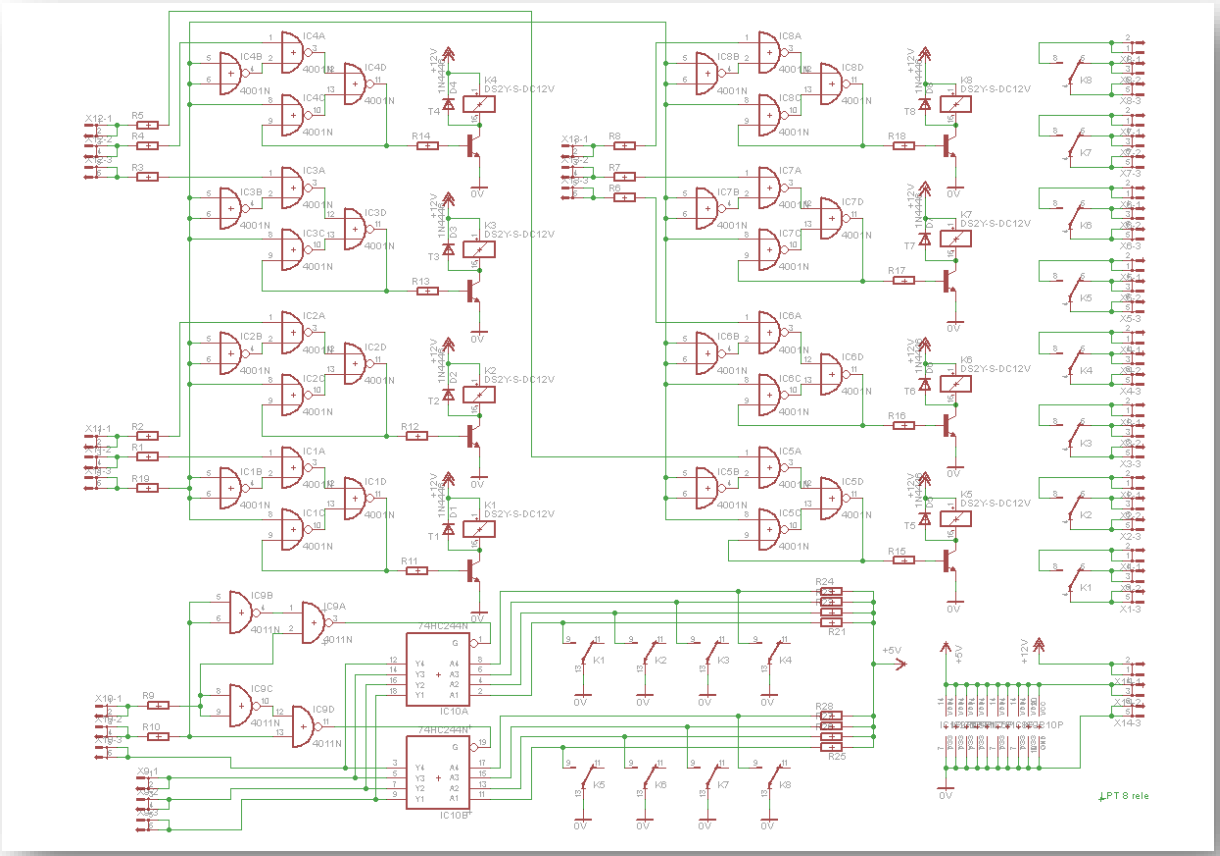
5. svorkovnice

levý kontakt	DATA - bit 5	- konektor LPT, pin 7	**DATA-5**
střední	DATA - bit 4	- konektor LPT, pin 6	**DATA-4**
pravý	DATA - bit 3	- konektor LPT, pin 5	**DATA-3**

6. svorkovnice

levý pin	DATA - bit 2	- konektor LPT, pin 4	**DATA-2**
střední	DATA - bit 1	- konektor LPT, pin 3	**DATA-1**
pravý	DATA - bit 0	- konektor LPT, pin 2	**DATA-0**

5.2. SCHÉMA ZAPOJENÍ

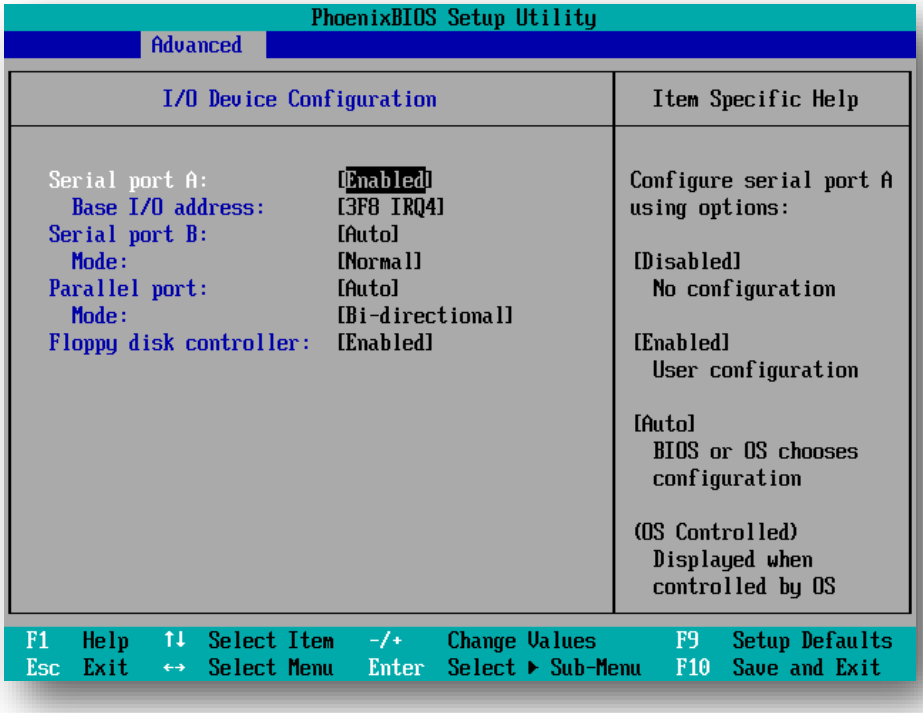


Obvodové schéma řídící desky

5.3. KOMUNIKAČNÍ PROTOKOL

Paralelní port přenáší 8 bitů současně, kdy je vždy informace vysílána po bajtech – paralelní přenos. Užívá 8-bitovou rozšířenou (extended) **ASCII** sadu znaků. Zpětně se pak odesílají do počítače informace o stavu jednotlivých relé na řídící desce.

5.4. NASTAVENÍ LPT PORTU V BIOS



Nastavení parametrů pro konfiguraci LPT (paralelního) portu jsou:

Onboard Paralel Port	[**AUTO**] nebo [**3F8H/IRQ4**]
Parralel Port Mode	[**Output Only**] nebo [**Bi-directional**]

POZNÁMKA:
*Volba **[AUTO]** nastaví všechny parametry automaticky, po této volbě další parametry nastavení znepřístupní nebo se skryjí.*

5.5. ŘÍDÍCÍ SPÍNACÍ MODUL

Spínací modul je v podstatě taková chytrá prodlužovačka, do které se připojují jednotlivé spotřebiče do zásuvek 1 až 7. Pod každou ze zásuvek je zelená LED dioda indikující, že je tato zásuvka pod proudem. Pokud by v této chvíli bylo na ní zapojeno nějaké zařízení, bylo by v provozu.

Modul je připojen kabelem LPT do počítače, napájecím kabelem ze zdroje počítače a kabelem 240 V do zásuvky, ze které bude napájen celý proces. Samotný počítač MUSÍ být napájen ze stejné větve, nejlépe ze stejné dvoj-zásuvky. Odběr počítače je cca 20-80W, s čímž je třeba v celém systému počítat. Ideální, ale ne nutná je samostatná větev s jističem 16 A. V bytech většinou tato větev existuje a slouží k napájení energeticky náročnějších spotřebičů např. pračka.

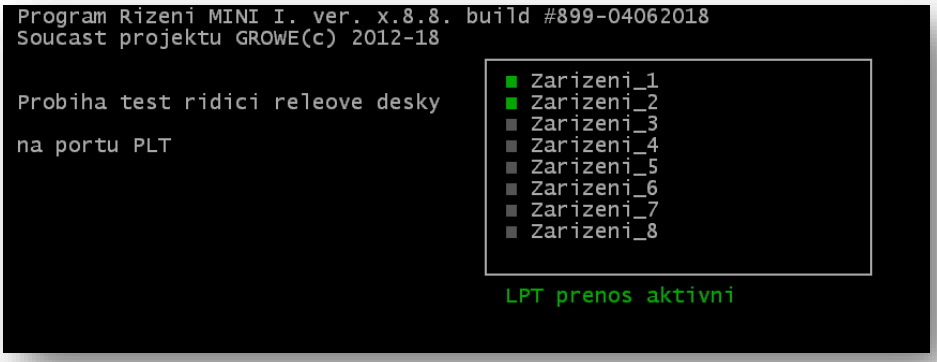
UPOZORNĚNÍ:
Stav LED diod, který indikuje, že je zásuvka pod proudem, svítí plným světlem. Stav relé zapnuto/vypnuto, pokud tedy bude vypnut hlavní vypínač napájení, budou LED diody indikovat sníženou intenzitou svitu.

5.6. PARAMETRY ŘÍDÍCÍHO MODULU PRO SPÍNÁNÍ 240 V

Řídící modul je navržen na spínání napětí **240 V** / max. proud **7 A**, což je teoreticky **1600 W** na spotřebič. Samozřejmě, že nelze takto zatížit všechny zásuvky, reálný počítaný odběr všech spotřebičů je okolo **1500/1800 W**. Pokud by bylo potřeba spínat vyšší výkon, je třeba použít např. výkonový stykač ovládaný touto zásuvkou. V tomto případě spínaný výkon je omezen velikostí vašeho jističe.

5.7. TEST ŘÍDÍCÍHO MODULU

Správnou funkci řídícího modulu lze prověřit pomocí programu Řízení MINI I., který je potřeba spustit s parametrem **/TESTBOARD**, celý příkaz pak vypadá takto: **MINI.EXE /TESTBOARD.** Program se spustí v testovacím režimu, kdy prověří jednotlivá spínací relé na funkčnost. Protože při testu dochází k rychlým změnám na jednotlivých relé, není vhodné modul testovat s připojenými spotřebiči. Po dokončení testu se zobrazí jeho výsledky a program se ukončí.



Probíhající test řídícího modulu

Pokud dojde během testu ke zjištění chybového stavu, zobrazí se podrobná informace o závadě, s možnostmi nápravy.

Možností opravy je několik:

- Zkontrolujte správné připojení řídícího modulu k portu LPT
- Zkontrolujte připojení řídícího modulu k napájení a zapnuté a hl. spínači.
- Zkontrolujte nastavení portu PLT v BIOSu řídícího počítače. Komunikační mód portu musí být v režimu [**Output Only**] nebo [**Bi-directional**].

UPOZORNĚNÍ:
*V žádném případě nenastavujte komunikační mód [**ECP**], [**EPP**] nebo jejich kombinaci [**ECP+EPP**]. V těchto módech řídící modul **nekomunikuje**.*

5.8. PODMÍNKY BEZPEČNÉHO POUŽITÍ A PROVOZ

Pro bezpečné použití, je potřeba se řídit několika důležitými kroky. Z hlediska bezpečnosti je v prvé řadě mít na paměti bezpečnost uživatele. Ta je zajištěna svou konstrukcí, obvody jsou galvanicky odděleny, pokud nebude uživatel zasahovat do zapojení, prakticky žádné nebezpečí nehrozí. Poněkud horší je ochrana hardware počítače, a to konkrétně portů, kdy je třeba při spuštění systému dodržovat určitý postup.

Vždy je potřeba nejprve všechna zařízení zapojit a teprve poté je spouštět. Důležité je zejména zapojení napájení řídícího modulu **5** a **12 V** ze zdroje počítače, kdy port **LPT** není na rozdíl od portu **COM** chráněn a pokud by na něj bez napájení byly poslány řídící příkazy, mohlo by dojít k trvalému poškození tohoto portu.

Hlavní spínač na těle řídícího modulu slouží k sepnutí hlavního přívodu **240 V** a pokud jsou sepnuta určitá relé, tyto ho propojí až na zásuvku. Pokud je tedy hlavní přívod vypnut, funkcionalita řízení a komunikace mezi počítačem a reléovou deskou zůstává zachována. Pravidlem pro bezpečné spuštění je, že tento spínač se sepne až po spuštění počítače, připojení modulu do sítě **240 V**, zapojení všech spotřebičů do něj. Rovněž ho lze použít jako bezpečnostní vypínač v případě nebezpečí nebo potřeby okamžitého vypnutí.

5.9. ES PROHLÁŠENÍ O SHODĚ

ES prohlášení o shodě znamená, že výrobek nebo zařízení je v souladu s předpisy a normami. Je to prohlášení o tom, že výrobek splňuje požadavky technických předpisů platných v EU (tedy i ČR). Postup při posouzení shody stanoví zákon 22/1997 Sb. v platném znění a nařízení vlády č. 176/2008 Sb., které odpovídá směrnici Evropského parlamentu a Rady 2006/42/ES o strojních zařízeních.

5.10. ELEKTROMAGNETICKÁ KOMPATIBILITA A PROVOZ

Tento výrobek nebo jeho část/i (dále jen zařízení) je/jsou v souladu s ochrannými požadavky směrnice rady Evropské unie 2004/108/ES o sbližování právních předpisů členských států vztahující se k elektromagnetické kompatibilitě. Zařízení bylo testováno a splňuje limity pro zařízení informačních technologií třídy B podle evropské normy.

POZNÁMKA:

Zařízení bylo testováno a splňuje limity pro digitální zařízení třídy B, pravidel FCC. Tyto limity jsou navrženy pro zajištění přiměřené ochrany před škodlivým rušením v obytném zařízení. Zařízení vytváří, používá a může vyzařovat vysokofrekvenční energii. Pokud se nenainstaluje a nepoužívá podle pokynů, může působit škodlivé rušení rádiové komunikace.

Neexistuje žádná záruka, že při konkrétní instalaci a provozu nedojde k rušení. Pokud zařízení skutečně způsobuje rušení rádia nebo televizoru, což lze zjistit vypnutím a zapnutím zařízení, doporučujeme k odstranění rušení vyzkoušet následující kroky:

- Přesměrujte nebo přemístěte přijímací anténu.
- Zvětšete odstup mezi zařízením a přijímačem.
- Zařízení zapojte do jiné zásuvky na jiném okruhu, než je připojen přijímač.

Zařízení třídy B splňuje směrnici CAN ICES-3 (B).

VÝSTRAHA:

Neměňte ani neupravujte konstrukci zařízení, a to včetně úpravy frekvence, na které tento výrobek pracuje, protože tak můžete znemožnit jeho volné použití.

6. PODPŮRNÉ PROGRAMY A UTILITY

6.1. ADRESÁŘOVÁ STRUKTURA PROGRAMŮ

C:\GROWE	hlavní adresář
C:\GROWE\RIZENI	adresář s programy hlavního řídícího procesu
C:\GROWE\RIZENI\BIN	adresář se zdrojovými soubory
C:\GROWE\RIZENI\MAN	adresář s manuály
C:\GROWE\RIZENI\UTIL	adresář s pomocnými programy a utilitami
C:\GROWE\RIZENI\SCRIPTS	adresář s již vytvořenými řídícími skripty

6.2. SEZNAM POMOCNÝCH PROGRAMŮ:

Skript Builder Pro	program pro tvorbu řídícího skriptu
Manual List	program nápovědy
DigiTemp Config File Creator	program pro konfiguraci teplotního čidla
DBF_Reader Pro	program pro kontrolu a inicializaci řídícího skriptu
Cryptor eXtended	program pro šifrování souborů
Spotřeba (modul)	modul pro sledování spotřeby energie
New Licence Requester	program vytvářející .RQS soubor pro vygenerování licence
Client Communicator	program dálkové správy
Sniffer Communicator	program pro sledování komunikace dálkové správy (pouze jako součást programátorské verze)

6.3. PROGRAM SKRIPT BUILDER PRO

Úvodní okno programu

Program slouží pro snadnou tvorbu a editaci řídícího skriptu procesu **GROWE**. Tento skript obsahuje příkazy, kterými se řídí spínání jednotlivých zařízení připojených na centrální spínací jednotku. Tato řídící jednotka je připojena k počítači přes rozhraní **LPT**, přes který je i ovládána.

Příkazy pro celý cyklus jsou uloženy ve skriptu v databázi dBase po jednotlivých větách. Věta je seznam příkazů pro určitý den. Příkazy ve větě jsou uloženy v polích, které mají specifické vlastnosti jako pozice ve větě, délka, typ informace, který obsahují a jiné. Této definici polí se říká tabulka databáze a fyzicky je v souboru uložena v hlavičce. Protože pro všechny věty platí stejná definice, jsou i všechny stejně dlouhé. Umožňuje to tedy snadné listování, snadnou úpravu vět i jednotlivých polí, přidávání, záměna,

třídění, mazání, a to jak nevratně, tak s možností obnovy. Ne všechny uvedené vlastnosti a možnosti jsou využity, pro možnost, aby databáze nesla "další rozšiřující informace", byl tento formát upraven a rozšířen. Zpětná kompatibilita byla zachována, ale pouze pro prohlížeče obsahu dBase ver. 3.xx kompatibilní. Pokud použijete dBase ver. 3.xx **editor**, po uložení databáze tomto editoru se nezachovají rozšiřující informace, proto se použití těchto programů pro editaci skriptu důrazně **nedoporučuje**.

Rozšiřující informace, které jsou do skriptu vloženy, jsou např. ochrana skriptu proti zápisu, vložení kontrolního součtu CRC, kryptování obsahu databáze a další. Popis rozšiřující hlavičky je popsán v sekci **FORMAT DBF.**

Program je vytvořen pro rychlou tvorbu tohoto skriptu, kdy pro pochopení, "jak na to" je třeba vysvětlit, jak tento skript chápe řídící program.

Skript obsahuje tedy hlavičku databáze, v níž je vložena tabulka plus další informace, dále jednotlivé věty, v počtu dnů procesu **GROWE**. Věta obsahuje 10 polí:

pozice	**název**	**typ**	**pozice**	**délka**
1.	DATUM	D	2	8
2.	DAY	C	10	6
3.	RELE_1	C	16	21
4.	RELE_2	C	37	21
5.	RELE_3	C	58	21
6.	RELE_4	C	79	21
7.	RELE_5	C	100	21
8.	RELE_6	C	121	21
9.	RELE_7	C	142	21
10.	RELE_8	C	163	21

První pole obsahuje datum (typ D), fyzický formát je **RRRRMMDD**, **RRRR** – rok, **MM** – měsíc a **DD** - den. Druhé pole je pořadové číslo dne procesu **GROWE**. Tyto první dvě pole jsou do skriptu doplňovány automaticky při následných procesech, které budou popsány později. Dalších osm polí je určeno pro příkazy. Z výše uvedené tabulky databáze vyplývá, že prostor pro příkazy má předimenzovanou délku, i když jejich fyzická délka může být i 2 znaky. Do plné délky jsou doplněny mezerami na konci, které jsou při zpracování ignorovány a při ukládání opět doplňovány, není potřeba je ručně dopisovat.

Program je navržen tak, aby tvorba skriptu a ukládání jednotlivých příkazů byla co nejjednodušší, dotazové dialogy kontrolují zadané údaje, pokud se rozhodnete psát příkazy přímo, bude kontrolovat obsah po zadání příkazu. Změní barvu textu na červenou a při ukládání věty na chybný příkaz upozorní. Přesto pokud budete na chybném obsahu trvat, program umožní tento uložit, v následných procesech budete na tuto chybu upozorněni a bez opravy nebude možné skript tzv. inicializovat, tj. vložení finalizačních informací a vložit kontrolní součet. Bez těchto informací, nebo při jejich neplatnosti hlavní řídící program takový skript odmítne vykonat. Proto, a i ve snaze o maximální stabilitu běhu procesu by mělo být v zájmu uživatele o bezchybné vytvoření skriptu. Pokud jej budete vytvářet pomocí dialogového menu, je možnost vytvoření chyby prakticky vyloučena.

Pro snadnou tvorbu skriptu je třeba podrobněji vysvětlit, jak jej chápe řídící program. Ten jej kontroluje na několika úrovních (úroveň ochrany lze nastavit v konfiguraci tohoto programu). Při prvotním otevření souboru nejdříve čte hlavičku souboru a dále hledá rozšiřující informace. Dále kontroluje platnost CRC součtu, pak kontroluje logickou platnost nejprve prvního a pak druhého pole v každé větě. Poté hledá větu s aktuálním datem, tuto načte a započne s vykonáváním jednotlivých příkazů.

Příkazy (podle popisu tabulky) jsou nazvány RELÉ 1–8. Tyto (pro jednoduchost nazvěme) relé, které spínají napětí na zásuvkách hlavního napájecího rozvaděče. Jednotlivá zařízení mají přesně stanovenou zásuvku pro zapojení, z této konfigurace je potřeba vycházet při zadávání příkazů.

Vycházejme z konfigurace, kdy na rozvaděči jsou zapojeny zařízení takto:

1. zásuvka	zvlhčovač
2. zásuvka	pomocné osvětlení
3. zásuvka	hlavní osvětlení
4. zásuvka	základní odsávání
5. zásuvka	přídavné odsávání
6. zásuvka	vnitřní cirkulace vzduchu
7. zásuvka	zálivka (čerpadlo nebo jiné zařízení)

Relé 8 slouží pouze k systémovým účelům, proto pro něj nelze zadat jiný příkaz, než vypnuto – OFF. Nicméně, rozšíření konfigurace o další zásuvku je teoreticky možné, v konečném návrhu s ním ale není počítáno.

Některá zařízení, v tomto případě uveďme odsávání je řešeno jedním spotřebičem se dvěma nezávislými napájeními, nebo dvěma nezávislými zařízeními. V prvním případě, kdy ventilátor má dva režimy odsávání, pomalý s nízkým výkonem a hlučností a maximální s vysokým výkonem a hlučností. Oba režimy nemohou jet současně, a to z důvodu, že motor ventilátoru má dva druhy vinutí, které nemohou být napájeny současně. Program má ochranu na úrovni řídícího modulu, kdy nedovolí jejich současné sepnutí, a to jak v automatickém, tak manuálním režimu ovládání. Proto lze pro relé 5 možno použít pouze příkaz ON, OFF nebo AUTO. Toto relé (zařízení) má ve své funkci v procesu **GROWE** vyšší prioritu, pokud použijete příkaz ON, sepne se přídavné odsávání a tím se odpojí odsávání základní. Tento stav se nezmění, dokud relé 5 nedostane příkaz OFF a nedojde k jeho vypnutí. Teprve potom bude relé 4 akceptovat příkaz pro sepnutí.

Pro jednotlivá relé lze použít pouze určité příkazy:

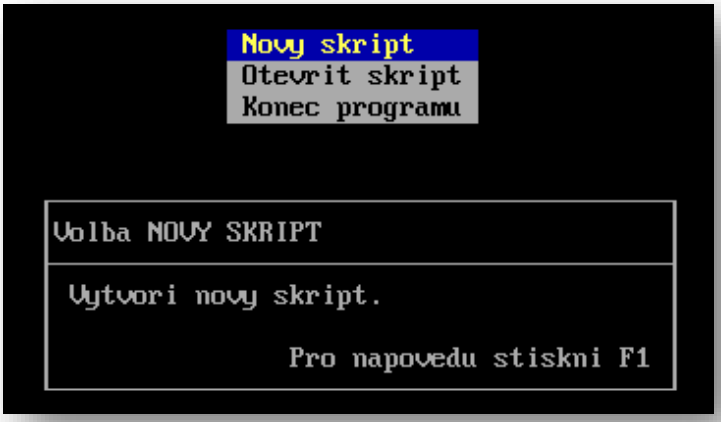
Relé 1	OFF, ON, TO, WHIT
Relé 2	OFF, ON, TO, WHIT
Relé 3	OFF, ON, TO, WHIT
Relé 4	OFF, ON
Relé 5	OFF, ON, AUTO
Relé 6	OFF, ON, TO, WHIT
Relé 7	OFF, ON, TO, WHIT
Relé 8	OFF

A teď samotnému návrhu skriptu. Nejdříve je si třeba ujasnit proces **GROWE**, druh a perioda světla, typ závlahy, délka jednotlivých fází procesu. Předpokládejme, že pro různé fáze procesu požadujeme jinou světelnou periodu, s jiným druhem světla, s jinou hodnotou vlhkosti.

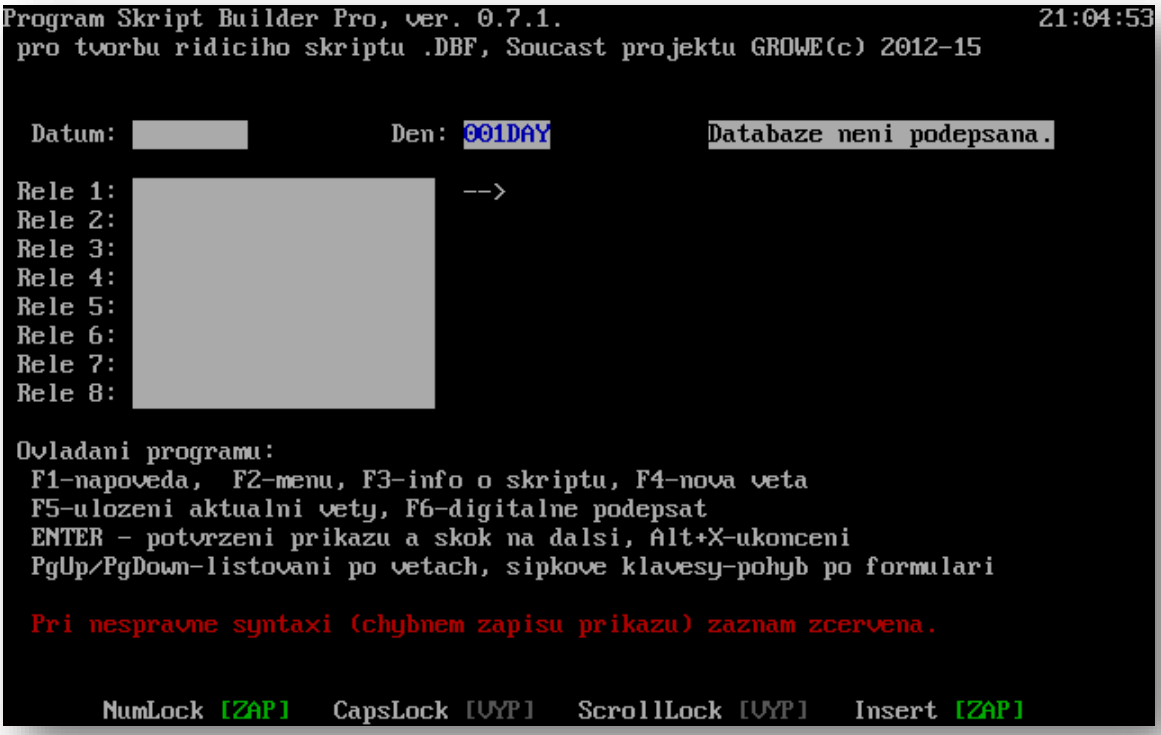
Pro první dny proto navrhneme pro I. fázi 18-ti hodinovou světelnou periodu, základní druh světla a maximální vlhkost. Postupně přidáváme druhé osvětlení až do plného nahrazení základního světla. Po celou dobu přizpůsobujeme požadavkům další funkcionality jako zálivka, vnitřní ventilace...

Po požadované délce fáze I. nastavíme přechod na 12-ti hodinovou světelnou periodu, (fáze II.), změníme četnost zvlhčování, které se postupně může úplně vyřadit. Jako ukončení lze nastavit všechny příkazy posledního dne na OFF, proces se neukončí přechodem přes půlnoc. Při ukončování procesu **GROWE** dochází k uložení veškerých dat disk, pro pozdější analýzu, proto nemusí být vždy žádoucí končit proces uprostřed noci.

Dá se říct, že vytvořit skript lze pouze přidržením klávesy **ENTER**. Ve skutečnosti to tak jednoduché není, pokud si ale jeho návrh dobře promyslíte, bude Vám program **Skript Builder Pro** dobrým pomocníkem při jeho tvorbě.



Úvodní menu



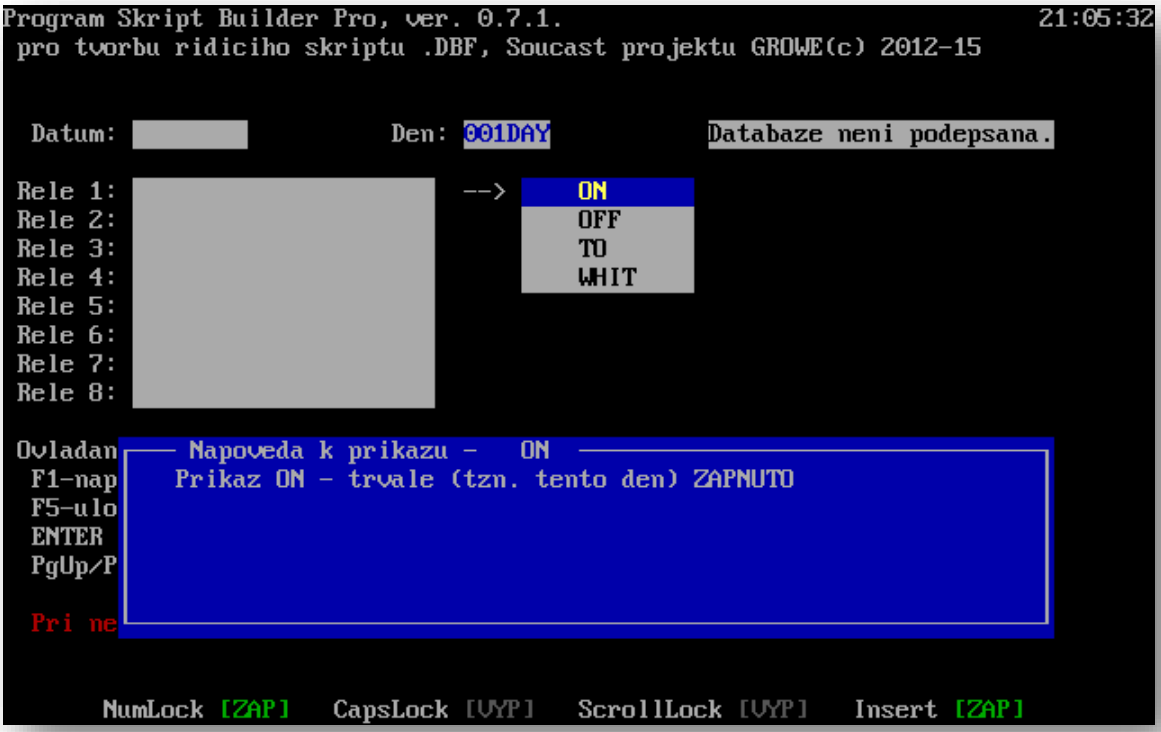
Hlavní obrazovka programu

Program po spuštění zobrazí úvodní menu, kdy nabízí vytvoření nového skriptu, nebo otevření již existujícího.

Při výběru otevření existujícího skriptu se zobrazí dialog pro výběr souboru, jinak program zobrazí hlavní obrazovku. V levé části je editační část pro zápis, v pravé jsou doplňující informace. Ve spodní části je nápovědný text, úplně dole je zobrazen stav systémových funkcí klávesnice. Pro okamžitou nápovědu lze kdykoli v programu použít klávesu F1, která zobrazí text vztahující se k aktuálně prováděné činnosti.

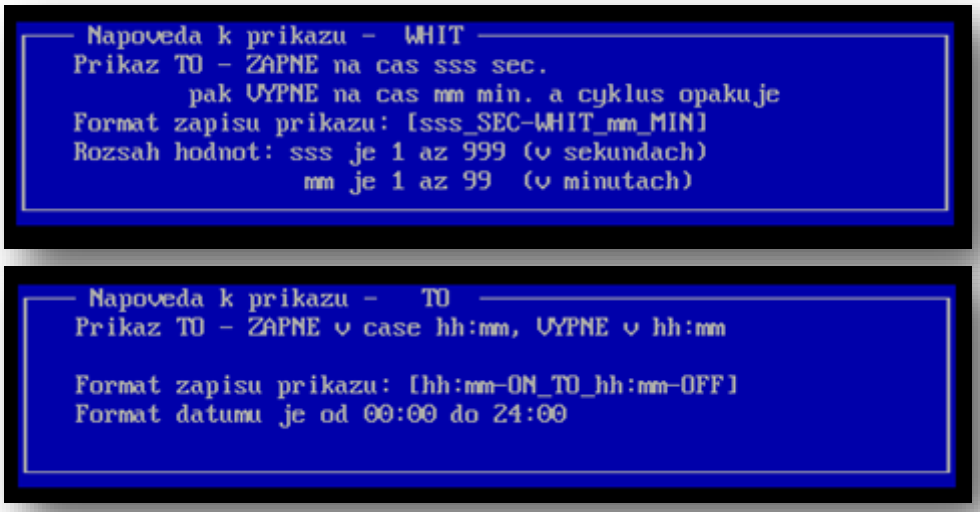
Editační okna Datum a Den nejsou přístupná pro zápis, jak bylo již dříve uvedeno, tyto informace se doplní automaticky. Pole Den se vkládá automaticky jako číselná řada +1 pro každý nový den, pole Datum se naplní datumovou řadou při inicializaci skriptu. Datumová řada je posloupnost datumů, kdy jako první se vloží aktuální ve chvíli inicializace skriptu, další následují dle kalendáře. Veškeré odchylky jako např. přestupný rok jsou akceptovány a program s nimi počítá.

Po výběru **Nový skript** je kurzor umístěn v poli Relé 1 na první pozici a je možno hned zadávat příkazy. Po stisku **ENTER** se zadaný příkaz zkontroluje, pokud je správný, přesune se kurzor na první pozici druhého okna Relé 2. Pokud není příkaz syntakticky správný, změní se barva textu na červenou. Pokud je příkaz psán malými písmeny, změní se na velká. Mezi jednotlivými editačními okny se lze pohybovat šipkovými klávesami, nebo kl. **TAB** (další) a **SHIFT+TAB** (předchozí). Mezi jednotlivými větami se lze pohybovat klávesami **PGUp** a **PGDown**.



Hlavní obrazovka s výběrovým menu příkazů

Klávesou **F2** lze vyvolat menu, které nabídne dostupné příkazy pro pole, na kterém je umístěn kurzor. Výběrem a potvrzením **ENTER** se příkaz vloží do pole. Pokud je potřeba doplnit další údaje do příkazu, přesune se kurzor na toto místo a očekává jeho vložení. Do příkazu lze vložit pouze povolené znaky (např. nelze vložit písmena do údaje čas), v opačném případě program tyto znaky smaže a zvukovou signalizací upozorňuje na chybné vložení. Pokud je údaj akceptován, přesune se kurzor na další údaj. Při ukládání se dále kontroluje logická správnost zadávaných údajů, např. není možné ve stejný čas provést zapnutí i vypnutí relé, druhý zadávaný údaj se v tomto případě smaže a požaduje vložení jiného údaje (příkaz TO). Po vložení příkazu do pole relé 8 se zobrazí dotaz na uložení právě vytvořené věty a vytvoření další v pořadí.



Varianty nápovědy menu příkazů

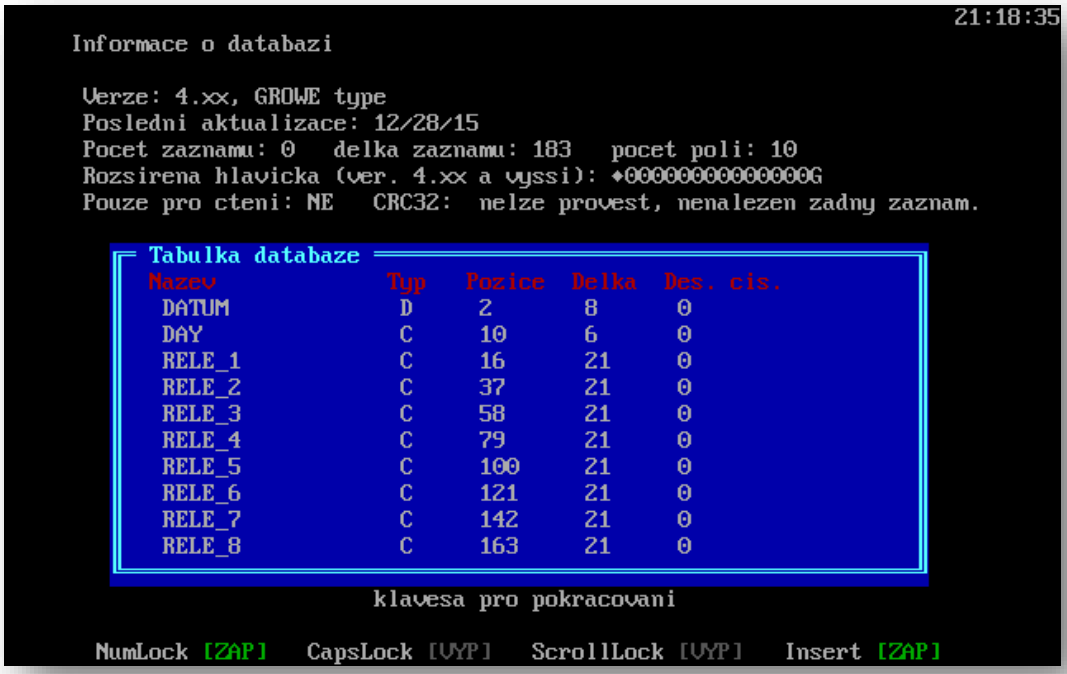
Věta se dá kdykoliv uložit klávesou **F5**, pokud ale obsahuje chybné příkazy označené červenou barvou, uložení se neprovede a zobrazí se chybové hlášení.

Po vytvoření celého skriptu lze klávesou **F6** vložit digitální podpis. Klávesa **F3** zobrazí podrobné informace o vytvářeném skriptu, verze, datum vytvoření, tabulka databáze, **CRC** součet a jiné.

Informace o příznaku zákazu zápisu

Program lze ukončit stiskem **Alt+X**, pokud během editace došlo ke změně v databázi, zobrazí se dotaz na uložení změn do souboru před ukončením.

Pokud je ale nastaven příznak Pouze čtení, není zápis změn do databáze možný. Proto neměňte obsah souboru takto označený (nejdříve přesvědčte se o možnosti zápisu, informace získáte pomocí klávesy **F3**), změny se neuloží a po ukončení programu budou ztraceny. Je důležité nejprve odstranit příznak **Pouze čtení**, teprve poté provádět změny v obsahu.



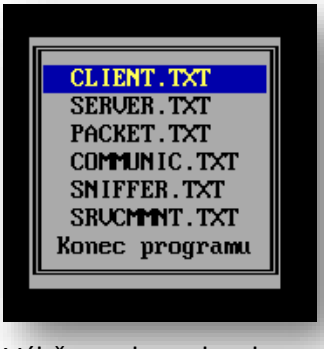
Informace o databázi

Součástí programu je několik již hotových skriptů, navržených pro univerzální použití. Tyto můžete jakkoli zkoumat, upravovat. Nicméně doporučujeme uchovávat použité skripty prověřené provozem, kdy každý úspěšně dokončený proces **GROWE** je do skriptu zapsán a tato informace pak slouží jako rozlišení mezi ověřeným a neověřeným skriptem. Tato informace je pouze informativní, její hodnota slouží pouze pro uživatele. Toto značení může být z důvodu ochrany soukromí zakázáno.

UPOZORNĚNÍ:
Pokud budete zaměňovat skript pomocí dálkové správy, je platný digitální podpis vyžadován ***VŽDY!!!***

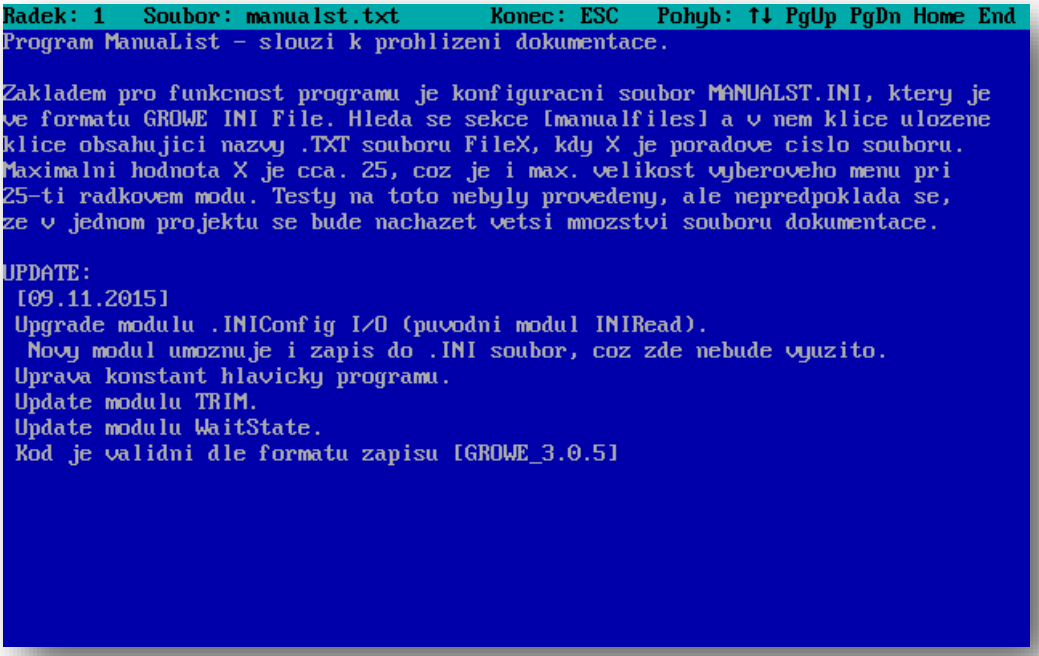
6.4. PROGRAM MANUAL LIST

Program slouží k prohlížení vývojářských poznámek k jednotlivým částem programu (moduly, služby apod.). Poznámky obsahují návrhy funkcionalit programů, na jejich definici byly poté vytvářeny.



Výběr souboru k zobrazení

Obsahují velké množství „neutříděných“ informací, které neprošly žádnou korekcí, jsou předkládány, tak jak jsou. Přes to obsahují důležité informace pro snazší orientaci ve zdrojových kódech.



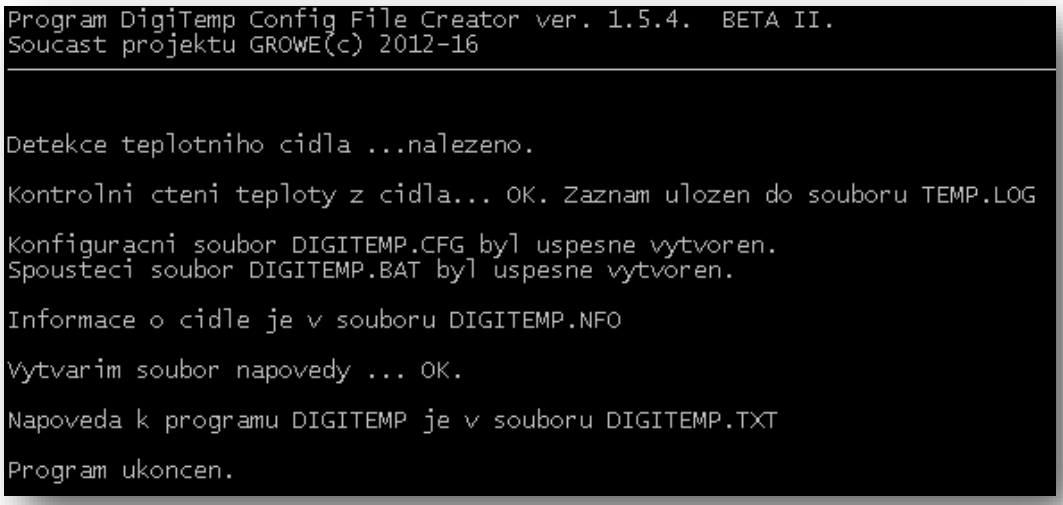
Příklad výpisu souboru elektronického manuálu

UPOZORNĚNÍ:

Rozšířený manuál je k dispozici ve vývojářské verzi, kdy tento je přiložen do archivu společně se zdrojovými kódy.

6.5. PROGRAM DIGITEMP CONFIG FILE CREATOR

Pro jednoduché vytvoření konfiguračních souborů slouží program **DigiTemp Config File Creator**,



Výpis hlášení při činnosti programu

soubor **DIGI_CFG.EXE**, který se nachází v adresáři **C:\GROWE\RIZENI\UTIL**.

Program pro úspěšné vytvoření konfiguračních souborů potřebuje, aby byl ve stejném adresáři se souborem **DIGITEMP.EXE**. Pokud tomu tak je, program po spuštění krátce pracuje, kdy několikrát osloví teplotní čidlo, čímž od něj získá důležité informace jako typ čidla, sériové číslo. Všechny získané informace se ukládají do textových souborů, vytvoří se i krátký nápovědný text (anglicky) s možnostmi použití programu.

Vytvořené konfigurační a informační soubory:

DIGITEMP.BAT	spouštěcí soubor pro cyklické čtení teploty z čidla nebo čidel
DIGITEMP.CFG	konfigurační soubor programu
TEMP.LOG	soubor, do kterého se ukládají informace o čtení teploty
DIGITEMP.NFO	informační soubor o teplotním čidlu

```
cls
@echo off
@echo Spusten proces mereni teploty. NEZAVIREJTE TOTO OKNO PRI BEHU PROCESU GROWE !!!
:begin
digitemp.exe -a -q -r55500
goto begin
```

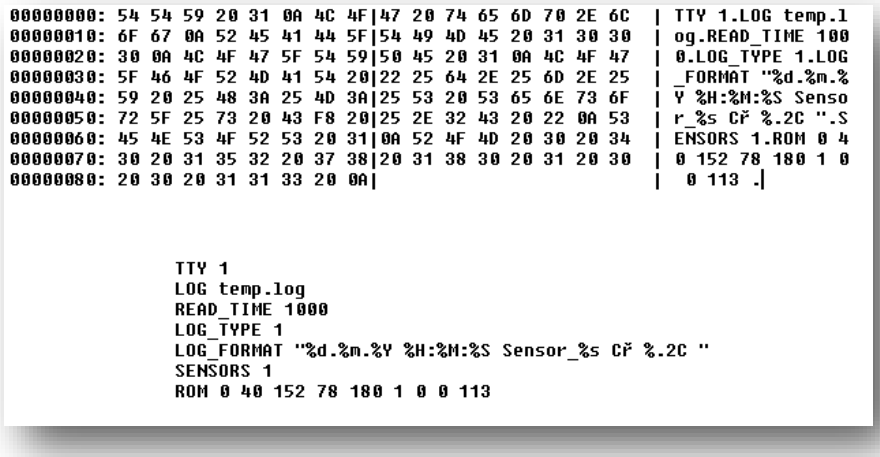
Obsah spouštěcího souboru DIGITEMP.BAT

DIGITEMP.TXT	nápovědný text o použití programu DIGITEMP.EXE
DIGITEMP.BAT	slouží k cyklicky se opakujícímu měření teploty.
DIGITEMP.EXE	utilita pro měření teploty

```
Turning off all DS2409 Couplers
.
Searching the 1-Wire LAN
28984EB401000071 : DS18B20 Temperature Sensor
ROM #0 : 28984EB401000071
```

Podrobné informace o čidle v souboru DIGITEMP.NFO.

Soubor **DIGITEMP.CFG** obsahuje konfiguraci. Je zde uveden ve dvou formátech, v textovém a hexa. Obsah souboru se jeví jako klasický text, ale obsahuje nestandardní formátovací znaky **&H0A**, pokud je editován klasickým textovým editorem, dojde k nahrazení těchto znaků klasickými **CL RF**. Tento formát program neakceptuje a načte výchozí hodnoty. Na posledním řádku je sériové číslo čidla v binární podobě. Každé čidlo obsahuje toto jedinečné sériové číslo, které nikdy není u dvou čidel stejné. Proto i tento řádek se pokaždé liší, a proto musí být konfigurační soubor vygenerován, nikoli pouze použit již existující.



```
00000000: 54 54 59 20 31 0A 4C 4F|47 20 74 65 6D 70 2E 6C  | TTY 1.LOG temp.l
00000010: 6F 67 0A 52 45 41 44 5F|54 49 4D 45 20 31 30 30  | og.READ_TIME 100
00000020: 30 0A 4C 4F 47 5F 54 59|50 45 20 31 0A 4C 4F 47  | 0.LOG_TYPE 1.LOG
00000030: 5F 46 4F 52 4D 41 54 20|22 25 64 2E 25 6D 2E 25  | _FORMAT "%d.%m.%
00000040: 59 20 25 48 3A 25 4D 3A|25 53 20 53 65 6E 73 6F  | Y %H:%M:%S Senso
00000050: 72 5F 25 73 20 43 F8 20|25 2E 32 43 20 22 0A 53  | r_%s Cř %.2C ".S
00000060: 45 4E 53 4F 52 53 20 31|0A 52 4F 4D 20 30 20 34  | ENSORS 1.ROM 0 4
00000070: 30 20 31 35 32 20 37 38|20 31 38 30 20 31 20 30  | 0 152 78 180 1 0
00000080: 20 30 20 31 31 33 20 0A|                        |  0 113 .
```

```
TTY 1
LOG temp.log
READ_TIME 1000
LOG_TYPE 1
LOG_FORMAT "%d.%m.%Y %H:%M:%S Sensor_%s Cř %.2C "
SENSORS 1
ROM 0 40 152 78 180 1 0 0 113
```

Obsah souboru DIGITEMP.CFG v textovém a Hexa zobrazení

6.6. PROGRAM DBF_READER PRO

Program kontroluje řídicí skript (ve formátu .DBF) ve třech fázích. Dále dle požadavků inicializuje databázi nebo inicializaci maže. Při inicializaci doplní do všech záznamů do prvního pole s názvem DATUM číselnou datumovou řadu, počínaje aktuálním. Při mazání inicializace smaže pole DATUM z prvního záznamu. Ostatní pole v dalších záznamech zůstávají zachována pro možnou obnovu.

Kontrola skriptu databáze:

1. **fáze** – kontrola inicializace skriptu
2. **fáze** – kontrola vyplnění všech slov (příkazů) ve větách
3. **fáze** – kontrola syntakticky správných zápisů všech příkazů

Nalezené chyb při testu **fáze 1.:**

- nejsou doplněny datumy do pole DATE, nebo jsou nalezeny v tomto poli nepovolené znaky
- chybové hlášení o nalezení neplatné inicializace skriptu

Nalezení chyby při testu **fáze 2.:**

- nejsou vyplněny příkazy ve všech polích
- chybového hlášení o nalezení prázdných polí

Nalezení chyby při testu **fáze 3.:**

- neplatné syntaxe řídících příkazů
- nelogičnost skladby příkazů
- nepovolené příkazy nebo kombinace příkazů pro určitá relé
- hlášení o nalezené chybě

Hlášení o probíhajícím testu se postupně zobrazují na obrazovce a zároveň se ukládají do souboru. Tento slouží jako popis historie skriptu od jeho vytvoření, přes výsledky provedených testů, provedené nebo smazané inicializace, přes informace o nasazení v procesu **GROWE**.

Pokud není skript inicializován, je možné toto provést pomocí přepínače při spouštění programu: **DBF_READ.EXE /INICIALIZE**

Při inicializace se vytvoří kopie skriptu do souboru **ZALOHAxx.DBF**, kde **xx** je číselná řada od **0** do **99**

- provede se kontrola existence, zda zálohový soubor s tímto názvem již existuje, pokud ano, použije se první volný název, (název + volné číslo z řady)
- do skriptu se doplní datumová řada, počínaje aktuálním datem

V programu jsou použity uživatelsky definovaná chybová hlášení, která lze libovolně definovat. Jsou uloženy v textovém souboru **ERRORCOD.TXT**. Číslo chyby rovněž vrací program při svém ukončení ve formě návratového (**ERRORLEVEL**) kódu.

Návratový **ERRORLEVEL** kód informuje, o výsledku kontroly řídícího skriptu, jedná se tedy jak o běhové chyby programu, tak o chyby ve struktuře skriptu, které bez opravy znemožňuje jeho nasazení v řídícím procesu **GROWE**.

Návratové kódy – **ERRORLEVELs**:

#0	Všechny operace proběhly v pořádku
#501	Volaní programu s parametrem **/INFO**
#502	Voláno s parametrem **/VERSION**
#510	Neznámy formát skriptu
#520	Soubor nenalezen
#530	Skript není inicializován
#540	Skript obsahuje neopravené chyby
#596	Příliš mnoho zálohových souboru (max. je 99)
#597	Spuštěno s neznámým parametrem
#598	Spuštěno bez povinného parametru
#599	Přerušeno uživatelem
#600	Smazaní inicializace řídicího skriptu

Všechny ostatní chyby jsou předány interní funkci ERR, zobrazující defaultní hlášení prostředí IDE a OS.

6.7. PROGRAM CRYPTOR EXTENDED

Program slouží k šifrování a dešifrování souborů. Někdy se také uvádí termín kryptování a dekryptování, je možné, že se setkáte s oběma termíny. Obecně se tato funkce využívá pro skrytí obsahu souboru, zabezpečení obsahu souboru proti změně a podobně.

Úvodní hlášení při spuštění programu

V Projektu **GROWE** se proces šifrování používá na několika úrovních. Šifrovaný je jednak datový přenos dálkové správy, kdy data, která „tečou“ z jednoho počítače do druhého jsou takto zabezpečena proti přečtení, nebo podvržení neoprávněného příkazu „bokem“ (odchycení komunikačního paketu s příkazem a jeho záměna). Zašifrován je soubor hesel pro přihlášení k dálkové správě, i všechny

Dialog výběru souboru

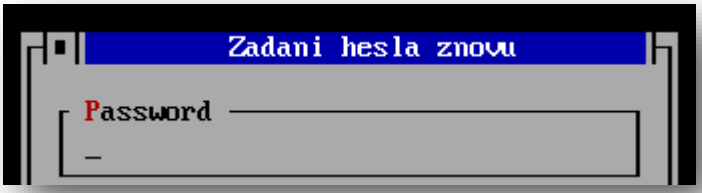
soubory, které si pomocí této služby můžete vyžádat z běžícího procesu **GROWE**.

K dešifrování do čitelné podoby slouží právě program **Cryptor eXtended**. Jeho použití je snadné, ale má i svá úskalí a v případě neodborného použití, nebo ztráty hesla může vést i **k trvalé ztrátě obsahu souborů**.

Po spuštění se zobrazí dialog pro výběr souboru. Program automaticky detekuje, zda soubor je již zašifrovaný, nebo se jej teprve chystáte šifrovat a podle toho automaticky vybere požadovanou funkci.

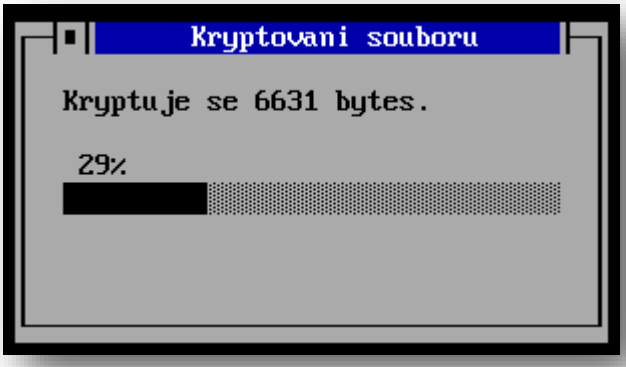
Dotaz na zadání hesla

V případě šifrování souboru se 2x dotazuje na heslo, pokud soubor dešifrujete, dotaz na heslo se zobrazí pouze jednou. Pokud je heslo správné, soubor se dešifruje a v této podobě se uloží do aktuálního adresáře. V případě šifrování se nejprve provede porovnání prvního a druhého hesla, pokud jsou stejná, soubor se zašifruje a uloží.



Dotaz na opětovné zadání hesla

V případě provedení dešifrování obsahu je použité heslo „zapomenuto“, při opětovném šifrování budete opět 2x dotazováni na vložení hesla.



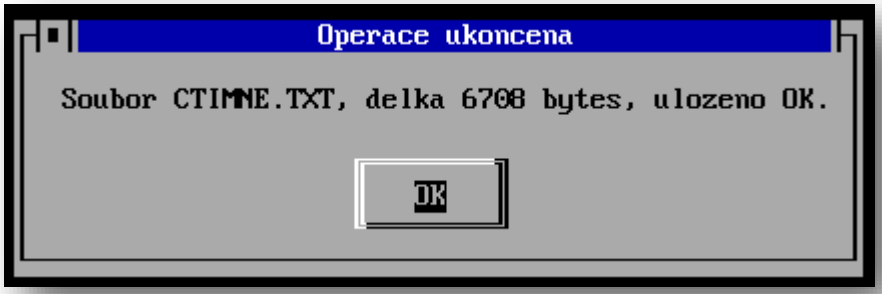
Průběh šifrování souboru

UPOZORNĚNÍ:
Heslo se nikde neukládá, pokud ho zapomenete, je obsah souboru trvale ztracen!!!

Program obsahuje několik ochran proti tzv. „nevratným" omylům při jeho požití:

- Při zobrazení nabídky souborů pro šifrování nenabízí název sama sebe - **CRYPTORX.EXE**. Je to ochrana proti "sebezničení" programu, kdy zašifrováním sama sebe se program přepíše do šifrované podoby, čímž se změní jeho spustitelná podoba. Výsledný zašifrovaný soubor **.EXE** již není spustitelný, při tomto pokusu může **OS** přestat reagovat a počítač musí být restartován.
- Program automaticky kontroluje obsah souboru, zda již není v šifrované podobě. Pokud ano, neprovede se vícenásobné šifrování, ale provede se po zadání správného hesla jeho dešifrování.

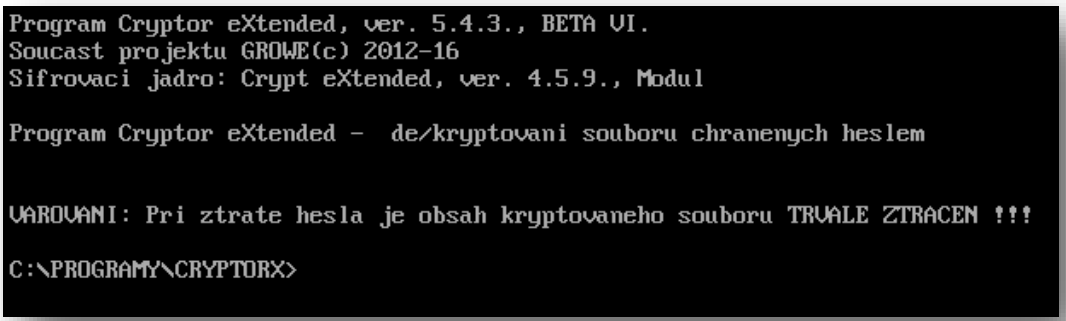
Funkce šifrování splňuje základní požadavky na ochranu obsahu souboru, ovšem za splnění několika podmínek:



Hlášení o úspěšném ukončení operace

- Heslo by mělo být přiměřeně dlouhé a složité. **Program rozlišuje mezi malými a VELÝMI písmeny**, je vhodné jejich kombinace společně s použitím čísel a speciálních znaků. Čím bude heslo delší, tím bude ochrana účinnější.
- Vyhněte se jednoduchým a často používaným heslům jako: 1234, heslo, admin, password a podobně.
- Nezapisujte si hesla, pokud ano, tak na nesnadno dostupná místa a nenápadným způsobem.
- Vícenásobné (víceprůchodové) šifrování není podporováno. Pokud požadujete vyšší stupeň ochrany obsahu, můžete soubor před šifrováním např. zkomprimovat (popř. i s ochranou heslem).

TIP: Pokud potřebujete zjistit informace o programu, spusťte jej s parametrem **/HELP**, příklad:



Informace o verzi programu

CRYPTORX.EXE /HELP. Velikost písma při zadávání příkazu není v tomto případě důležitá.

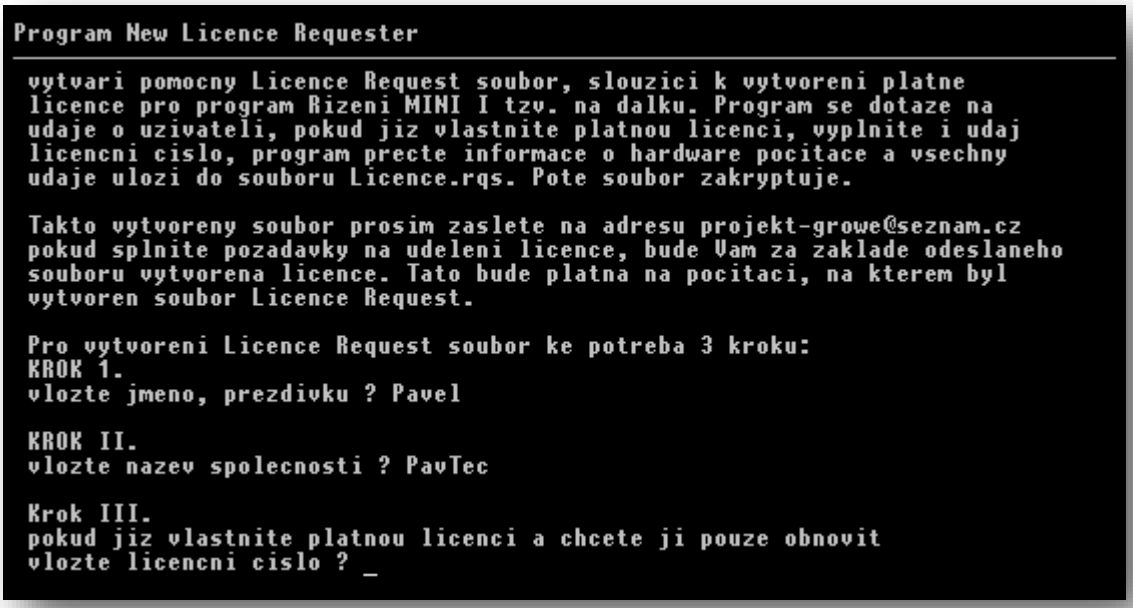
UPOZORNĚNÍ:
Při dešifrování systémových souborů procesu GROWE, např. soubor hesel pro přihlášení k dálkové správě, je potřeba, aby po provedení změn v něm byl opět zašifrován, a to STEJNÝM heslem, kterým byl předtím dešifrován. Je to z důvodu, že při přihlašovacím procesu dálkové správy program ŘÍZENÍ provádí dešifrování tohoto souboru a pokud mu změníme heslo, operace neproběhnou úspěšně a přihlášení se nezdaří.

6.8. MODUL SPOTŘEBA

Modul pro sledování spotřeby elektrické energie. V současné době se modul nachází ve fázi vývoje, o jeho dokončení budete informováni na webových stránkách projektu GROWE.

6.9. PROGRAM NEW LICENCE REQUESTER

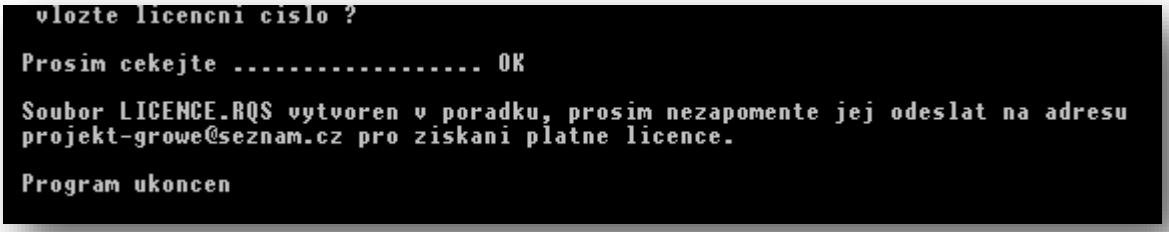
Program slouží k vytvoření **.RQS** (request-požadavek) souboru, na jehož základě se generuje licenční klíč k programu **Řízení MINI I.** Program se dotáže uživatele na už. jméno (respektive přezdívku), název



Náhled běhu programu

společnosti, pokud již vlastní licenční klíč a je požadována pouze obnova existující licence, vloží uživatel po dotazu i tento. Program dále provede rychlou analýzu počítače, výsledky uloží do souboru **LICENCE.RQS**.

Uživatel musí tento soubor odeslat společně s požadavkem jako přílohu e-mailu na adresu **Projektu GROWE**, uvedenou v kapitole **11. Kontaktní spojení**



Informace o odeslání výsledného souboru LICENCE.RQS

6.10. PROGRAM SNIFFER COMMUNICATOR

Program slouží ke sledování síťového provozu v ladícím módu, kdy je záměrně snížena její rychlost, aby bylo možno jednotlivé pakety odchytávat a analyzovat. Program je stále ve vývoji, v nejbližší době bude přidána i analýza TCP komunikačních paketů. Bližší informace jsou součástí programátorské verze manuálu.

POZNÁMKA:
Podrobný popis je součástí programátorské verze manuálu.

7. INSTALACE A AKTUALIZACE PROGRAMŮ

7.1. INSTALACE PROGRAMŮ

Instalace je naprosto triviální operace, kdy po stažení balíku programů v komprimované podobě **.RAR** se celý obsah „rozbalí“ do kořenového adresáře. Všechny programy jsou již v příslušných adresářích, jak je popsáno v kapitole **6. Podpůrné programy a utility**, část nazvaná **Adresářová struktura programů**.

7.2. AKTUALIZACE PROGRAMŮ

Aktualizace programů se provádí vždy jako manuální zásah uživatele, tedy neprobíhá automaticky. Aktualizační balíčky jsou k dispozici na Webovém serveru projektu. Adresa je uvedena v kapitole **11. Kontaktní spojení**. K aktualizačním souborům je přiložen podrobný popis aktualizace v textové formě. Instalační a aktualizační balíčky jsou v komprimované podobě ve formátu **WinRAR**.

Jako aktualizace nejvyšší priority jsou ty, které opravují chyby nebo dis-funkcionality hlavního nebo pomocných programů. Doporučuje se, je nainstalovat, jsou označeny písmeny **RC** (**R**elease **C**andidate) v kombinaci s číslicí 1 popř. 2 – např.: Program Řízení MINI I. Ver. 1.0.3. **RC1**.

Další v žebříčku priorit jsou aktualizace, přinášející nová rozšíření a funkce, tzv. **BETA** verze. U nich byl vývoj nových funkcionalit již ukončen a nyní probíhá jejich testování. V případě, že budete tyto aktualizace použít a podílet se na tomto **BETA** testu, je potřeba se striktně držet návodu pro aktualizaci, který je součástí aktualizačního archivu. Je pravděpodobné, že tyto novinky budou s sebou přinášet i změny v klíčových částech řídícího programu (např. modifikace komunikačního protokolu software-hardware, změna nebo přidání příkazů dálkové správy, změna příkazu v řídícím skriptu apod.). V tomto případě je doporučeno provést úplnou zálohu (popsáno v kapitole **7.3. Zálohování**)

Nejníže co se týče priorit, nejníže jsou aktualizace přinášející nové funkce **ve fázi vývoje**. Programy, které obsahují, jsou označovány jako **ALFA** verze. Jsou určeny pro testovací účely, nedoporučuje se tyto programy používat v ostrém provozu. Funkcionality, které obsahují, mohou být nasazeny pouze experimentálně, ve finální verzi se mohou objevit v pozměněné podobě, nebo mohou být vypuštěny úplně.

7.3. ZÁLOHOVÁNÍ

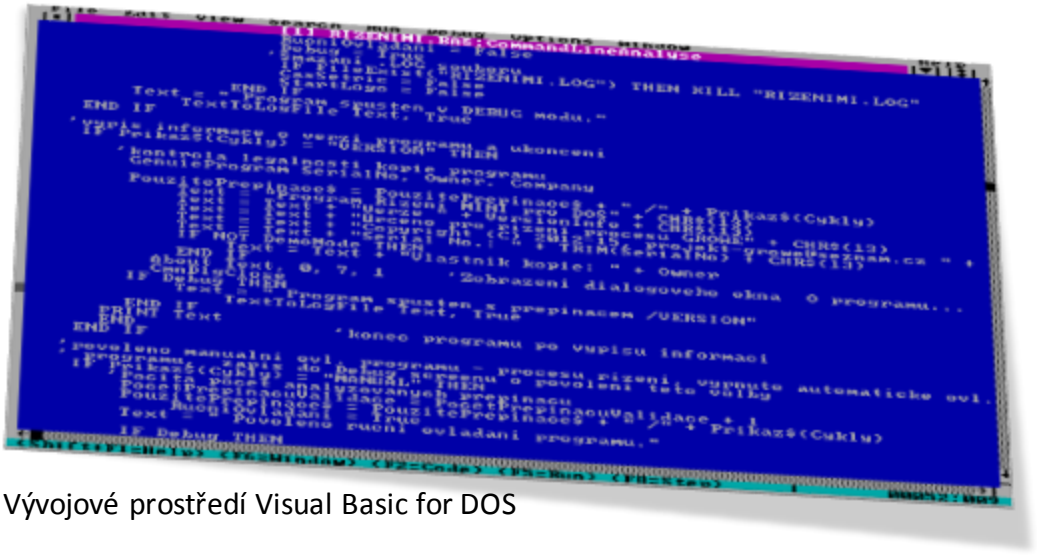
Zálohování je vzhledem k velikosti balíku programů cca. 5-6 MB poměrně snadná operace. Zkomprimujte celý adresář (složku) procesu Řízení **C:\GROWE** , pokud jej chcete uložit mimo řídící počítač, nebo jej prostě přejmenujte např. s koncovkou **.OLD** (**OS** umožňuje takto přejmenovat i adresáře) a proveďte instalaci do výchozího adresáře znovu. Poté, pokud potřebujete, překopírujte z původní zálohy potřebné soubory, např. řídící skripty, konfigurační soubory apod. Takových záloh můžete mít na disku v podstatě neomezené množství, které je ve své podstatě limitováno jen kapacitou disku. Každá ze záloh, pokud není zkomprimovaná do souboru, ale pouze provedena přejmenováním

adresáře je stále spustitelná a použitelná, pokud se nachází v kořenovém adresáři disku, např. pokud se provede záloha přejmenováním adresáře **C:\GROWE** na **C:\GROWE.ZALOHA1**.

UPOZORNĚNÍ:
*Pokud spouštíte programy ze zálohy, ujistěte se před tím, že nemáte spuštěny programy z hlavního adresáře, protože např. program Řízení nepodporuje vícenásobné spuštění programu a služba **SERVER** bude aktivní pouze v první instanci (v prvním spuštěném programu).*

8. ZDROJOVÝ KÓD

8.1. VÝVOJOVÉ PROSTŘEDÍ IDE



Vývojové prostředí Visual Basic for DOS

Jako jazyk pro vytvoření celého programového balíku byl zvolen **Visual Basic for DOS Pro, ver. 1.00.** Program je vytvořen pomocí programových modulů, obsahující jednotlivé procedury a funkce. Deklarace všech proměnných odpovídají explicitním deklaracím dle příkazu **OPTION EXPLICIT**. Některé procedury jsou vytvořeny v **Microsoft Macro Assembler - MASM for DOS 6.11 PDS**, zkompilovány do **.OBJ** a přidány do knihovny **QuickLibrary .QLB.**

8.2. VEŘEJNÁ ČÁST KÓDU

Určité části kódu, které jsou klíčové pro chod, komunikaci a ochranu programu jsou neveřejné. Všechny ostatní části lze označit jako veřejnou část kódu a za podmínek uvedených v kapitole **9.9. Podmínky pro získání zdrojového kódu** je lze získat.

UPOZORNĚNÍ:
Podrobné informace jsou součástí programátorské verze manuálu.

9. LICENCE A AUTORSKÁ PRÁVA

9.1. AUTORSKÁ PRÁVA

Autorská práva, pokud nejsou výslovně uvedena jinak, náleží společnosti **Projekt GROWE**. Autorská práva náležející k programovým modulům, které jsou použity, jsou ve zdrojových kódech uvedeny, včetně jejich vlastníků.

9.2. LICENČNÍ UJEDNÁNÍ

Toto licenční ujednání je právní smlouvou mezi Vámi, koncovým spotřebitelem (dále jen uživatel) softwarového a hardwarového produktu **Projekt GROWE** (dále v kapitole **9. Licence a autorská práva** označován jako **produkt**) a společností **Projekt GROWE Group**. Součástí produktu je hlavní řídící program **Řízení MINI I.**, nástroje pro analýzu a správu a pomocné programy. Produkt jako takový se neprodává, pouze se uděluje licence k jeho užívání. Instalací a používáním produktu uživatel potvrzuje, že souhlasí s podmínkami tohoto ujednání a zavazuje se jimi řídit.

NA ZÁKLADĚ TÉTO LICENČNÍ SMLOUVY SE UDĚLUJÍ NÁSLEDUJÍCÍ PRÁVA:

Instalace kopie produktu se vztahuje na jeden počítač. Počet kopií je omezen dle počtu získaných licencí, které je prokázáno počtem držení platných licenčních klíčů. Licence neumožňuje vícenásobné instalace produktu na dalších počítačích. Každá licence je pevně svázaná s hardware počítače, na kterém je produkt provozován. Vícenásobné instalace a/nebo zálohy programu na počítači, k němuž se vztahuje platná licence, jsou povoleny bez omezení.

POPIS DALŠÍCH PRÁV A OMEZENÍ:

Uživatel není oprávněn produkt prodávat, pronajímat, půjčovat nebo poskytovat sublicence jiným osobám či firmám. Zpětná analýza produktu není povolena. Licence se vydává na produkt jako celek. Jeho komponenty nelze oddělovat pro použití na více než jednom počítači, a to pouze na tom, k němuž se vztahuje licence. Je zakázáno, upravovat chod programu prostřednictvím produktů třetích stran. Zasahovat do databáze (import/export nebo zpracovávání dat) je povoleno, avšak s přihlédnutím na možné porušení autorských práv jejich výrobců. Projekt GROWE si vyhrazuje právo na změnu licenční politiky bez předchozího upozornění.

9.3. ZÍSKÁNÍ NOVÉ NEBO OBNOVENÍ EXISTUJÍCÍ LICENCE

Uživatel, který má zájem o získání platné licence, může o ni zažádat elektronickou formou, vyplněním formuláře Žádost o získání licence na Webových stránkách a vytvořením tzv. request souboru, který je třeba odeslat společně se žádostí v tomto formuláři. Vytvoření souboru je popsáno v kapitole **6.9. Program New Licence Requester**. Stejným způsobem lze získat i již existující a ztracenou licenci, ve

formě licenčního klíče (souboru **.KEY**). Obnovení se vztahuje pouze na platné licence a na počítač, pro který byla vydána.

9.4. PŘEVOD LICENCE

Uživatel smí převést veškerá svá práva vyplývající z uzavřené smlouvy trvale na jiného majitele (dále nabyvatel) a to za předpokladu, že předá společně se softwarovým produktem i hardware, na který je licence vázána. Dále se uživatel zavazuje plně odstranit vytvořené kopie produktu, neponechání si žádných kopií a převedení celého produktu (včetně všech hardwarových komponentů, medií, tištěného materiálu a všech verzí produktu) na nabyvatele, pokud tento souhlasí se všemi body tohoto ujednání.

9.5. OMEZENÍ ZÁRUKY

Podmínkou vzniku nároku na záruku je neporušení licenční smlouvy. Záruční doba je 24 měsíců od data převzetí produktu, pokud není definováno smlouvou jinak. Záruka se vztahuje na bezvadný chod programu, čímž se rozumí chod programu v souladu s dokumentací a jejími dodatky, vady produktu budou v rámci záruky řešeny bezplatným zasláním nejbližší další distribuční aktualizací programu. Reklamace s přesným popisem vady a žádostí o uplatnění záruky přijímá Projekt GROWE či distributor výhradně **elektronickou formou**. S výjimkou těchto záruk je produkt poskytován bez jakýchkoli dalších záruk. Za závadu produktu nelze označit skutečnost, že produkt v sobě neobsahuje případné legislativní změny, které nebyly výrobci známy v okamžiku výroby produktu, nebo že nepracuje na hardwaru, pro který není určen nebo s ním není svázán licencí. **Projekt GROWE nezaručuje bezvadný chod produktu, pokud:**

a) je provozován spolu s programy jiných výrobců, které znemožňují bezvadný chod programu
b) je provozován na chybně nakonfigurovaném počítači, nebo na chybně nastavené počítačové síti
c) uživatel provedl zásah do produktu nebo jeho souborů pomocí jiných prostředků, než je samotný program nebo další nástroje, které jsou součástí programového vybavení.

Pokud dojde ke ztrátě funkčnosti základní desky řídícího PC, se kterým je licence svázána, bude takto zneplatněná licence obnovena elektronicky **zdarma.**

9.6. OMEZENÍ ODPOVĚDNOSTI

Projekt GROWE nenese odpovědnost za případné následné škody, ať jsou jakékoli, které by vznikly na základě použití jeho produktů. **Projekt GROWE** nenese odpovědnost za použití hardwarového nebo softwarového vybavení k jiným účelům, než je výslovně uvedeno.

9.7. UKONČENÍ PLATNOSTI LICENCE

Licence produktu je platná pouze do svého ukončení. Licence produktu bude ukončena s okamžitou platností, pokud bude porušen kterýkoli bod tohoto ujednání.

Ukončením platnosti licence se program stává demonstrační verzí s omezenou funkčností. Omezení spočívá ve změně komunikace mezi řídícím programem a hardwarovým modulem na simulovaný.

9.8. DEMONSTRAČNÍ VERZE

Demonstrační verzi programu je možné volně kopírovat, rozmnožovat a šířit za podmínky, že každá takováto kopie bude obsahovat všechny soubory programového balíku **Řízení** včetně kompletní a nepozměněné dokumentace, označení názvu programu, projektu a kontaktu na autora.

9.9. PODMÍNKY PRO ZÍSKÁNÍ ZDROJOVÉHO KÓDU

Pro možné získání zdrojového kódu programu je třeba kontaktovat autora programu, a to vždy elektronickou formou. Do žádosti uveďte účel, pro který chcete tyto zdroje získat. Jako důvod uznání může být např. zájem o spolupráci při dalším rozvoji programů, studijní účely apod. Žádost bude posouzena, pokud bude schválena, pak do 30-ti dnů obdržíte odpověď, společně s odkazem na elektronickou smlouvu **DEVELOP_#001-17**. Po jejím úplné a pravdivé vyplnění a odsouhlasení, které nahrazuje podpis, se vytvoří na serveru vývojářský účet, který umožní přístup k archivům zdrojových kódů. Pokud bude žádost zamítnuta, nemusíte být o tomto informován a **nevzniká vám žádný nárok pro jejich získání**. Jedinou možností je opětovná žádost.

Pokud máte zájem o získání kódu ze studijních důvodů, máte možnost si vyžádat rozšířenou verzi tohoto manuálu, tzv. verzi **SOURCE Code**, obsahující rozsáhlý popis modulů a příklady použití pro správu databází, čtení a zápisu .INI souborů, šifrování, dálkové správy a řízení in-time procesů a dalších. Určité části jsou vytvořeny autory třetích stran, poskytnutí zdrojových kódů v tomto případě není možné, procedury/moduly jsou dostupné pouze v kompilované podobě jako .OBJ, nebo jako součást knihoven **Quick Library .QLB** .

Některé zdrojové kódy programů mohou být uvolněny k volnému použití. Většinou jsou poskytnuty „tak jak jsou“, případně (pokud existuje) i s deníkem vývoje, el. poznámkami, testovacími pomocnými programy apod. Odkazy ke stažení archivů **.ZIP** nebo **.RAR** naleznete na WEB stránkách **Projektu GROWE**. S takto uvolněným kódem můžete libovolně nakládat, modifikovat, zahrnovat je i jejich části do vlastních projektů a neomezeně dále šířit.

9.10. SPOLUPRÁCE PŘI VÝVOJI

Pro vývoj programů, jejich rozšíření, přidání analytických nástrojů dat, rozšíření funkcionalit procesů řízení, konverzi programů pro platformu RaspberryPI a BETA testování hledáme spolupracovníky.

Bližší informace naleznete na WEB stránkách projektu **GROWE** nebo nás kontaktujte na e-mailu development@projekt-growe.cz

10. SLOVNÍK POJMŮ

Projekt GROWE
Proces GROWE
Řídící program
Řídící skript
Řídící příkazy
Služba Server
Služba Řízení
Služba ŠetřičObrazovky
Služba ČteníTeploty
Služba Validace
Služba Keyboard
Licence
Moduly
dBase
Databáze
Hlavička databáze
Tabulka databáze
Program Skript Builder Pro
Program DBF Reader Pro
Program Client Communicator
Dálková správa
Příkazy Client&Server
Konfigurace
.INI soubor
.LOG soubor
OS – operační systém
Hardware
Software
BIOS
Port COMM
Port LPT
Teplotní čidlo
Řídící reléová deska
Řídící modul
Program Cryptor eXtended
Program Manual List
Program Sniffer Communicator
Program Script Studio
Aktualizace programu
Zálohování
Zdrojový kód
Vývojové prostředí IDE

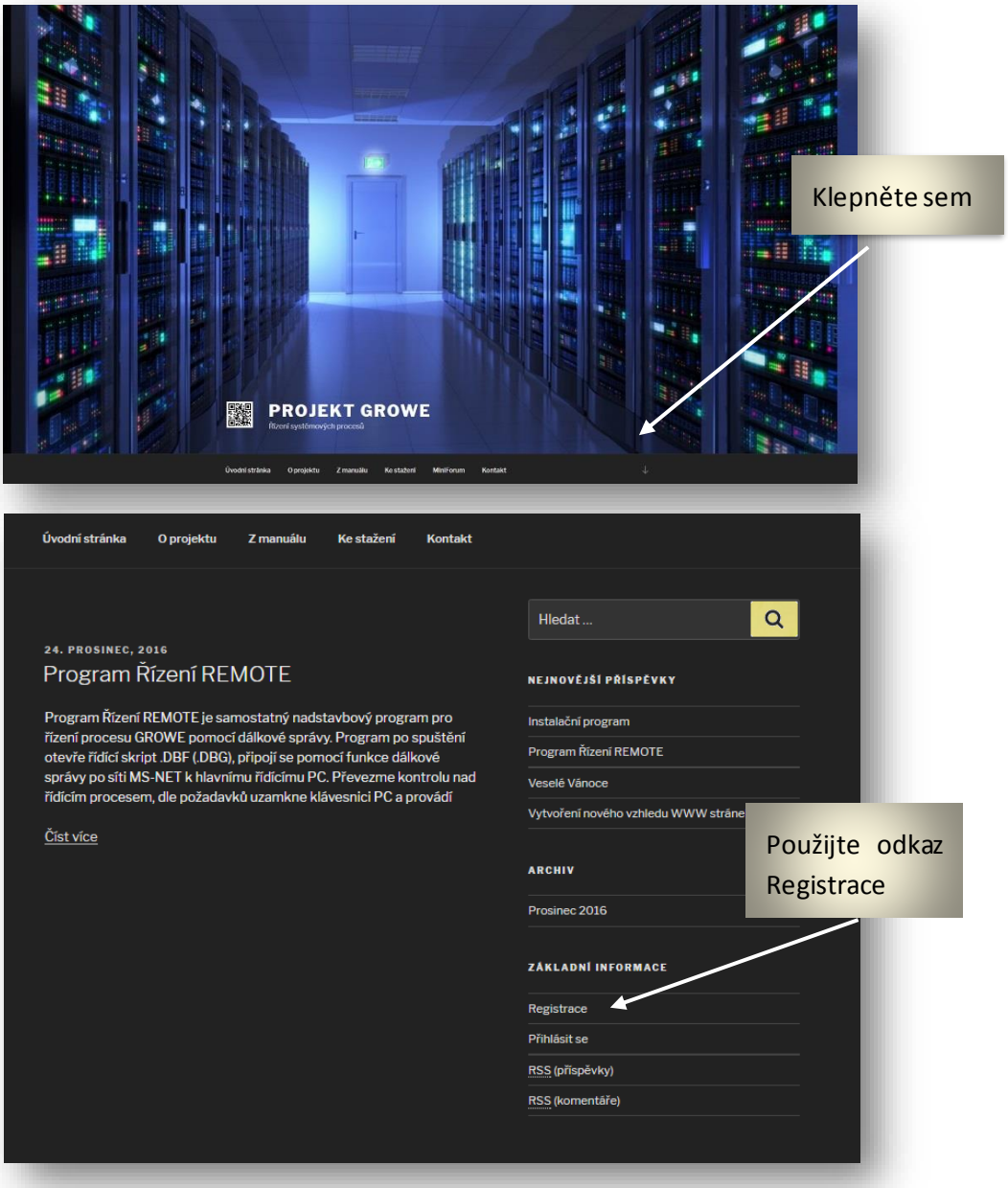
11. WEBOVÉ STRÁNKY A KONTAKTY

11.1 KONTAKTNÍ SPOJENÍ

WEB stránky projektu:	www.projekt-growe.cz
E-mail:	support@projekt-growe.cz
E-mail (dotazy ke zdrojovým kódům):	development@projekt-growe.cz

11.2. REGISTRACE NOVÉHO UŽIVATELE

Máte–li zájem o zasílání informací o novinkách e–mailem, prosím zaregistrujte se, nebo se přihlaste



k odběru na stránkách projektu, v menu **ZÁKLADNÍ INFORMACE**, odkaz **REGISTRACE**.

K výhodám registrace patří získávání informačních e-mailů o novinkách v **Projektu GROWE**, možnost přidávat komentáře k vybraným článkům, přístup na forum podpory, možnost stahování testovacích BETA verzí programů, přístup ke zdrojovým kódům programů uvolněných k volnému použití a šíření a další.