Práce s paměťovou kartou v systémech Tecomat

TXV 003 43.01
třetí vydání
květen 2008
změny vyhrazeny

Historie změn

Datum	Vydání	Popis změn
Únor 2008	1	První vydání
	2	Opraveno objednací číslo modulu PM-1901
Květen 2008	3	Přiděleno objednací číslo dokumentu TXV 003 43 Změna názvu z původního „Ukládání dat na paměťovou kartu v systémech Foxtrot“ na nový název „Práce s paměťovou kartou v systémech Tecomat“

OBSAH

1 ÚVOD

Systémy Tecomat Foxtrot a TC700 s procesorovou jednotkou CP-7004 umožňují použití paměťových karet. Primárním úkolem karty v těchto systémech je uložení web stránek pro integrovaný web server. Otázkou je, zdali lze tyto karty využít i pro jiné účely, například pro ukládání dat z PLC programu. Odpověď zní ano, ale je potřeba splnit určité podmínky. Které podmínky to jsou a co je dobré si uvědomit při programování se snaží vysvětlit tento dokument.

1.1 Nástin problému

Nejprve několik obecně známých údajů. Paměťové karty prodělaly v posledních létech obrovský pokrok. Výsledkem je obrovská škála karet na trhu, nízká cena, velká kapacita dat. Karty se masově používají především ve fotografických přístrojích a mobilních telefonech což do značné míry určuje jejich vlastnosti. Pro uživatele těchto zařízení jsou nejdůležitějším kritériem při výběru paměťové karty následující parametry: kapacita karty, rychlost přístupu k datům na kartě (především pak při zápisu dat), standardní interface a cena karty v poměru k její kapacitě. Těmto kritériím se přizpůsobili výrobci karet. Jedny z nejvíce rozšířených karet jsou dnes karty typu SD, mikro SD a MMC. Například velká většina fotoaparátů dnes podporuje některý z těchto formátů, velmi často lze použít libovolnou uvedenou kartu. Což samozřejmě ovlivňuje množství prodávaných karet a tím i jejich cenu. Typické použití karet v mobilních telefonech a fotoaparátech pak určuje i další vlastnosti. Především je to používaný souborový systém. Uživatel chce mít možnost kartu ze zařízení vyjmout a přečíst ji v počítači. Takže výrobci zvolili FAT file systém. Ten je notoricky známý z operačního systému DOS a je podporován na všech počítačích standardu PC jak operačním systémem Linux, tak všemi variantami Windows. Otázku, zdali to byla nejlepší možná volba, přenecháme raději někomu jinému. Pokud chceme použít běžně dostupnou kartu, bude naformátovaná jako FAT16 nebo FAT32. Až potud je vše bez problémů. Kde je tedy rozdíl mezi použitím paměťové karty například ve fotoaparátu a použitím v PLC. První úskalí spočívá v parametru, o který se zpravidla uživatelé fotoaparátu vůbec nezajímají. Tímto parametrem je maximální garantovaný počet zápisů do karty, který se v současné době pohybuje někde kolem 100 000 zápisů do karty. Jinými slovy, životnost karty je dána především počtem zápisů. Pro použití ve fotoaparátu nebo v mobilním telefonu je to číslo dostatečně velké na to, aby uživatele prakticky nezajímalo. Zjednodušeně řečeno, až udělá 100 000 fotografií, tak si možná bude muset koupit novou kartu. V PLC systému je situace výrazně odlišná. Předpokládejme, že plánovaná životnost PLC v technologii je 10 let, technologie je s nepřetržitým provozem a požadavkem je ukládat na kartu data snímaná z řízené technologie například jako součást dokumentace výrobního procesu. Pak tedy zjednodušeným výpočtem zjistíme, že počet zápisů do karty denně = 100 000 : (365 x 10) = 27,397. To znamená, že pokud provedeme více jak 24 zápisů denně, karta nemusí vydržet plánovaných 10 let. Uvedený výpočet je značně zjednodušený, protože parametr 100 000 zápisů se týká každého segmentu v paměťové kartě, takže aby výpočet platil, museli bychom celých deset let přepisovat pouze jeden region v paměti karty. Při reálném použití bude situace s velkou pravděpodobností lepší, protože data nebudou ukládána pouze na jedno místo na kartě. Ale skutečností zůstává, že nelze do karty zapisovat každých 100 milisekund nová data, protože její životnost to prostě vylučuje. Další rozdíl souvisí s rychlostí zápisu do karty. Fotografa jistě zajímá, jak rychle se uloží celý snímek na kartu. Jestli během ukládání snímku trvají některé operace zápisu déle než jiné je pro něho zcela nezajímavé. Z tohoto faktu vychází také výrobce karty, který se snaží o co nejlepší čas uložení celého souboru. Takže je skutečností, že ne všechny operace při zápisu do karty trvají stejně

dlouho, převážná část zápisů probíhá rychle, ale jsou také okamžiky, kdy karta při zápisu relativně dlouho „přemýšlí“. Naopak programátora PLC bude především zajímat, jak dlouho může trvat jednotlivá operace zápisu, protože o tento čas se prodlouží doba cyklu PLC. A ty nejvíce důležité časy budou právě ty, kdy karta reaguje pomalu. Tyto časy se dají zjistit pouze experimentálně, výrobci karet je neuvádějí. Posledním výrazným rozdílem mezi použitím paměťové karty v mobilním zařízení a použitím v PLC je situace, kdy dojde k vypnutí zařízení. Napájení mobilních zařízení zajišťuje baterie, takže není problém při požadavku na vypnutí nejprve dokončit ukládání všech souborů a poté skutečně vypnout. Pokud je karta připojena k počítači, nelze kartu bezpečně vyjmout z počítače (resp. Čtečky paměťových karet) bez toho, že uživatel požádá operační systém o odpojení zařízení. V té chvíli operační systém dokončí zápisy na kartu a poté oznámí, že kartu lze bezpečně odejmout. Pokud dojde například k výpadku sítě, všichni uživatelé osobních počítačů jistě vědí, že při následném zapnutí počítače začne operační systém nejprve kontrolovat disk. Pokud v okamžiku výpadku probíhal zápis zápis do karty, soubor se v lepším případě neuloží a v horším případě dojde k porušení souborového systému. V nejhorším případě pak dojde ke ztrátě nejen právě ukládaného souboru, ale i některých dalších. To je obecně známo. Závěr tedy zní, že není možné odpojit napájení systému v libovolném okamžiku, jinak hrozí ztráta dat na kartě. U PLC systémů je situace o to komplikovanější, že se počítá s tím, že PLC systém lze bez rizika vypnout kdykoliv (nikoliv pouze tlačítkem Start). Procesor PLC je sice vybaven baterií, ale ta slouží k napájení obvodu reálného času a paměti SRAM, takže k tomu, aby si systém pamatoval datum, čas a zálohované proměnné označené v programu jako RETAIN. Baterie nemá dostatečnou kapacitu na to, aby udržela v chodu procesor systému a další obvody potřebné k tomu, aby byl dokončen zápis na kartu. Takže vzniká problém, jak zajistit, aby se souborový systém na kartě neporušil při výpadku napájení.

1.2 Zadání úlohy

Předpokládejme následující zadání. Úkolem pro PLC je sledovat vybrané veličiny řízené technologie a ukládat jejich hodnoty do souborů na kartu. Ukládání může být pravidelné (jednou za časovou jednotku) nebo na základě události (například když se změní sledovaná proměnná o zadanou hodnotu). Ukládání dat bude nedílnou součástí řízení. Soubory uložené na paměťové kartě musí být jednoduše přenositelné do počítače, ať už kvůli archivaci průběhu řízeného procesu nebo kvůli dalšímu zpracování, například vyhodnocení poruch v technologii apod. Pro přenos souborů do počítače se využijí standardní prostředky dostupné v každém počítači. Soubory přenesené do počítače musí jít na kartě smazat. Přístup k souborům na kartě může být podmíněn zadáním jména uživatele a hesla.

1.3 Návrh řešení

Z předchozích kapitol vyplývá, že při ukládání dat na paměťovou kartu je třeba věnovat pozornost následujícím problémům:

- frekvence zápisů souborů na kartu
- prodloužení doby cyklu PLC při zápisu souboru
- vypnutí napájení systému během ukládání souboru na kartu
- formát ukládaných dat
- struktura adresářů, do kterých budou data ukládána
- viditelnost uložených souborů pro web server

Frekvence zápisů souborů na kartu

Pro snížení frekvence zápisu souborů do karty je vhodné ukládat data nejprve do DataBoxu, což je paměť typu SRAM, jejíž obsah je v systému Foxtrot zálohován vestavěným akumulátorem a volitelně lithiovou baterií (CR2032). Akumulátor je schopen udržet obsat DataBoxu po dobu minimálně 500 hodin, s osazenou baterií je DataBox zálohován 20 000 hodin. Podrobnost viz TXV 00410. Systém je standardně dodávaný bez baterie. Baterii lze kdykoliv doplnit do připraveného držáku. Do souboru na kartu pak budou ukládaná data z DataBoxu typicky jednou až několikrát denně.

Prodloužení doby cyklu PLC při zápisu souboru

Při zápisu do souboru dochází prodloužení doby cykly o čas potřebný na zápis. Tento čas je závislý na rychlosti karty (viz příloha) a také na operaci, kterou se právě provádí. Z toho důvodu je potřebné některé operace rozložit na několik cyklů tak, aby výsledný čas byl co nejmenší. Takže popisované řešení nejprve během několika cyklů otestuje, zda existují zadané adresáře, pokud některý adresář neexistuje tak ho založí, pak otevře soubor pro zápis, poté do souboru postupně uloží data z DataBoxu a na závěr uzavře soubor. Operace zápisu souboru tedy trvá několik cyklů PLC.

Vypnutí napájení systému během ukládání souboru na kartu

Během fyzického zápisu do karty nelze vypnout napájení systému, neboť hrozí nebezpečí poškození souborového systému na kartě. Tento problém lze řešit následujícími způsoby.

Buď je možné zálohovat napájení PLC (použít UPS), monitorovat přechod na záložní napětí a v případě výpadku hlavního napájení ukončit zápis do souboru. Jinými slovy podmínit zápis do souboru přítomností hlavního napětí. Lze předpokládat, že při výpadku hlavního napájení nebude pracovat ani technologie, takže nebudou ani žádná data k uložení. V případě výpadku záložního napájení z UPS je už vše v pořádku, neboť do karty se nezapisuje a nemůže tedy dojít ke ztrátě dat na kartě.

Druhou variantou řešení je použít podpůrný modul napájení PM-1901 (objednací číslo TXN 119 01). Modul PM-1901 se zapojuje mezi napájecí zdroj a základní modul pro Foxtrot. Jeho úkolem je zajistit napájení základního modulu Foxtrot ve chvíli výpadku napájení z hlavního zdroje po dobu nezbytně nutnou pro úspěšné dokončení operací zápisu na kartu. Po dobu, kdy se do souboru ukládá, jsou všechna data stále uložena v DataBoxu, takže při vypnutí napájení se sice nedokončí zápis celého souboru, ale po opětovném zapnutí lze soubor znovu uložit. Při opakovaném uložení se na kartě přepíší původně uložená data.

Formát ukládaných dat

Teoreticky není žádné omezení formátu, v jakém budou data do souboru uložena. Data mohou být uložena v binární podobě, mohou být kryptována, komprimována, atd. Prakticky je ale výhodné zvolit formát podle způsobu dalšího zpracování souborů po jejich přenosu do počítače. Takovým hojně používaným formátem je CSV (Comma Separated Value == čárkami oddělené hodnoty), který se běžně používá pro přenos dat mezi programy. Tento formát je vhodný pro import souborů do tabulkových kalkulátorů a databázových programů. Data se ukládají textově po řádkách,

jednotlivé hodnoty na řádku jsou odděleny čárkami. Každý řádek v textu pak odpovídá jednomu řádku hodnot v tabulkovém kalkulátoru.

Příklad hodnot uložených v CSV formátu :

```
Time,Temperature,Pressure
13:00:46.90,21,101
13:10:46.91,22,122
13:20:46.91,23,153
```

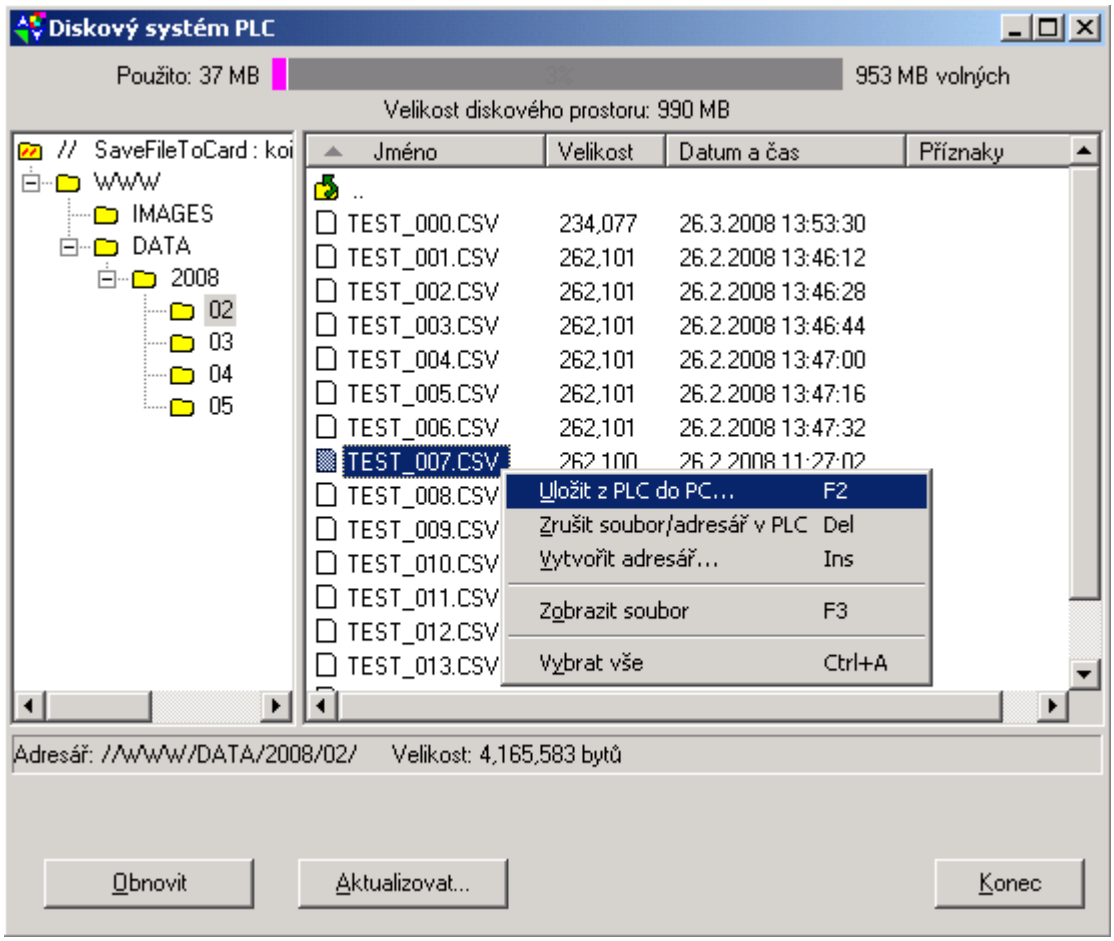
Pro ukládání dat v tomto formátu lze s výhodou použít datový typ STRING, do kterého lze snadno převádět proměnné jiných datových typů. Vytvoření jednoho řádku hodnot z výše uvedeného příkladu může vypadat například následovně

```
record := TIME_TO_STRING(GetTime());
record := CONCAT(IN1 := record, IN2 := ',');
record := CONCAT(IN1 := record, IN2 := INT_TO_STRING( temper));
record := CONCAT(IN1 := record, IN2 := ',');
record := CONCAT(IN1 := record, IN2 := INT_TO_STRING( press));
record := CONCAT(IN1 := record, IN2 := '$0D$0A');
```

Výhodou formátu CSV je jednoduchost, snadné vytvoření, čitelnost dat bez speciálních nástrojů (stačí jakýkoliv textový editor respektive prohlížeč) a snadná přenositelnost dat do dalších programů. Nevýhodou je větší spotřeba paměti a tím i místa na kartě například v porovnání s binárně uloženými daty.

Struktura adresářů na MMC/SD kartě

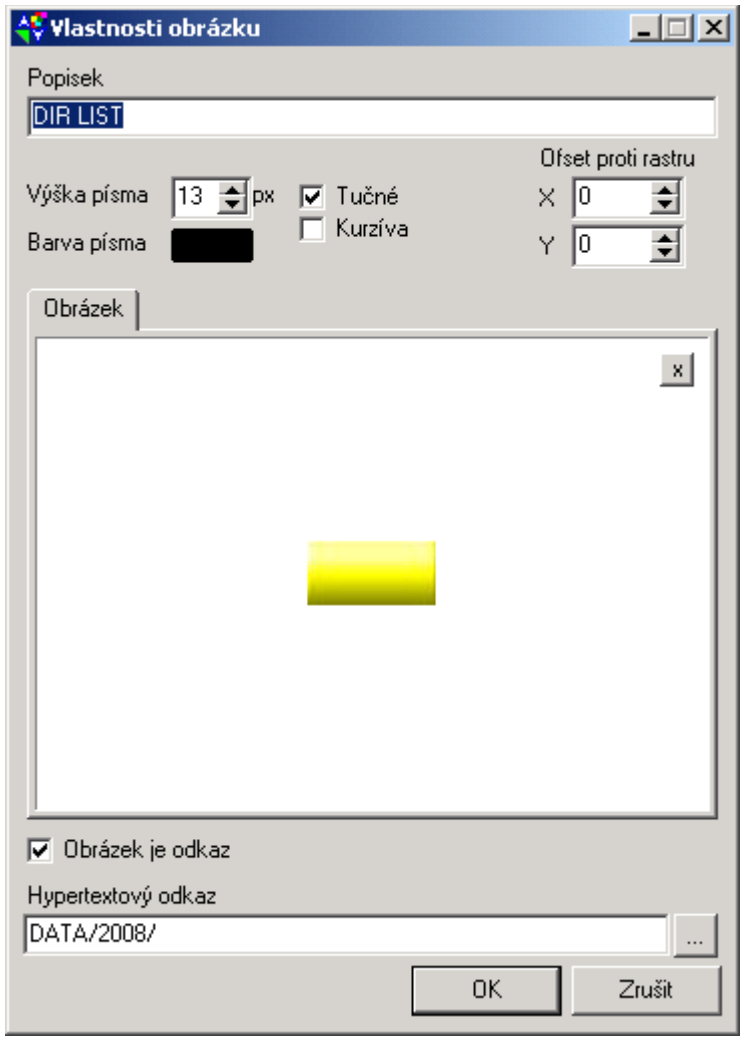
Kořenový adresář pro souborové operace v PLC systému se jmenuje ROOT. Programátor PLC systému může pracovat pouze s těmi soubory a adresáři, které jsou umístěny v adresáři ROOT. Ostatní soubory a adresáře nejsou z PLC programu dostupné. Adresář ROOT je tedy pracovním adresářem programátora PLC (to znamená, že všechny cesty zadané v PLC programu jsou cesty uvnitř adresáře ROOT). Pokud předpokládáme, že data budou ukládána dlouhodobě, je třeba řešit strukturu adresářů, do kterých budou data ukládána. Obecně lze doporučit, aby byly soubory ukládané aplikačním programem PLC oddělené od ostatních souborů na kartě, tedy v samostatném adresáři (například DATA/}. Pokud bude uložen jeden soubor denně, bude to za jeden rok 365 souborů. Je tedy vhodné vytvořit pro každý rok samostatný adresář a pro lepší přehled jestě adresáře pro jednotlivé měsíce. Příklad takové struktury adresářů je uveden na následujícím obrázku.



Viditelnost uložených souborů pro web server

Web server PLC používá jako kořenový adresář ROOT/WWW. Pokud mají být soubory dostupné přes web rozhraní, musí být uloženy v této cestě (např. v adresáři ROOT/WWW/DATA). Chceme-li zobrazit obsah adresáře ve web stránce, stačí do stránky zadat odkaz DATA/2008/. Maximální počet takto zobrazených souborů je 64. Dialog nástroje WEB Maker pro nastavení odkazu na adresář je vidět na následujícím obrázku.

Naopak budou-li soubory uloženy na stejné úrovni jako adresář WWW, nebudou z web serveru přístupné. Takové soubory lze přenést do počítače pouze z programovacího prostředí Mosaic (viz předcházející obrázek).



Další podrobnosti jsou uvedeny popisu příslušných centrálních jednotek PLC a v popisu knihovny pro souborové operace TXV00341_01.

Uvedené zásady jsou použity v následujícím příkladu.

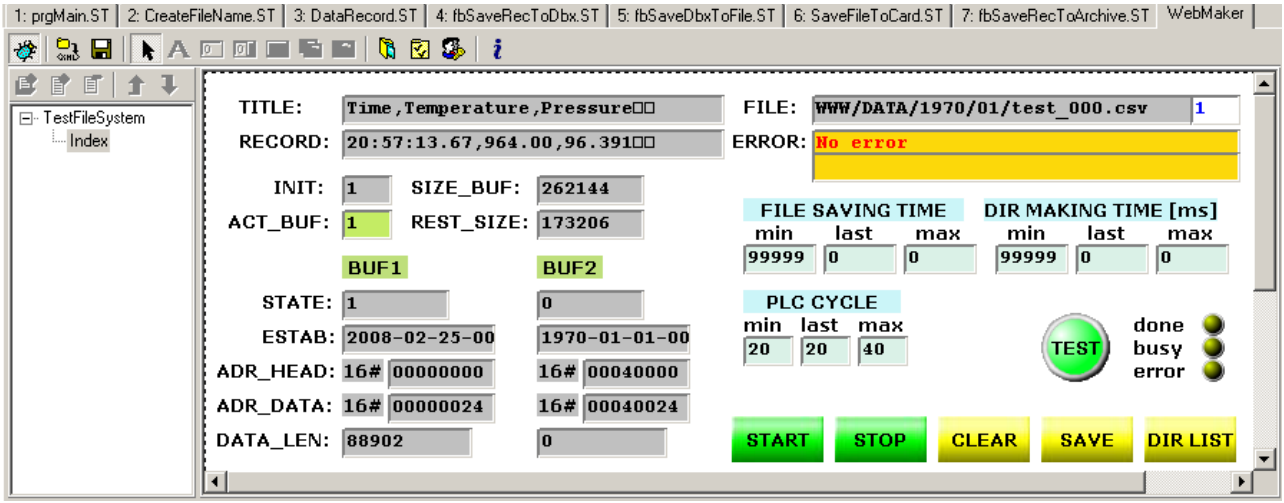
1.4 Příklad řešení

Příklad řešení úlohy definované v kap.1.2 s použitím zásad z kap.1.3 je možno najít v projektové skupině MMC_SD_Card v projektu SaveFileToCard. Příklad lze použít jako základ pro řešení úloh podobného typu jakož i pro testování parametrů konkrétní paměťové karty.

Hlavní program ukládá do CSV souboru hodnoty proměnných *tempVal* a *presVal* společně s časovou značkou. Pro ukládání záznamů je použit funkční blok typu *fbSaveRecToArchive*, který ukládá data do bufferu v paměti *DataBox* odkud je po zaplnění bufferu přepisuje do souboru. Pro přípravu záznamu je použita funkce *PrepareRec*, která sestavuje jednu větu ukládaného záznamu. Další funkce *CreateTestFileName* připravuje jméno souboru, do kterého budou záznamy uloženy. Jméno souboru je sestaveno včetně cesty, do které bude soubor uložen.

Testovací program je možné ovládat z okna nástroje WebMaker, které slouží jak pro vytvoření web stránek pro PLC, tak pro ladění a ovládání programu v prostředí Mosaic. Tlačítka v pravé spodní části okna mají následující význam:

- START spuštění testu
- STOP pozastavení testu
- CLEAR vynulování a inicializace bufferů v DataBoxu
- SAVE ruční uložení aktivního bufferu do souboru
- DIR LIST odkaz na adresář, pro zobrazení obsahu adresáře ve web stránce (toto tlačítko je funkční pouze při zobrazení stránky v internetovém prohlížeči)



Význam dalších polí v okně Web Makeru je následující:

- TITLE hlavička ukládaného záznamu
- RECORD aktuální ukládaný záznam (časová značka, hodnota proměnné *tempVal*, hodnota proměnné *přesVal*)
- FILE jméno naposledy uloženého souboru
- ERROR chybové hlášení

- INIT buffery v DataBoxu jsou inicializovány
- SIZE BUF velikost bufferu pro ukládání záznamů v bytech
- ACT BUF číslo bufferu, do kterého se aktuálně ukládají záznamy
- REST SIZE zbývající prostor, který je v aktivním bufferu k dispozici pro ukládání záznamů

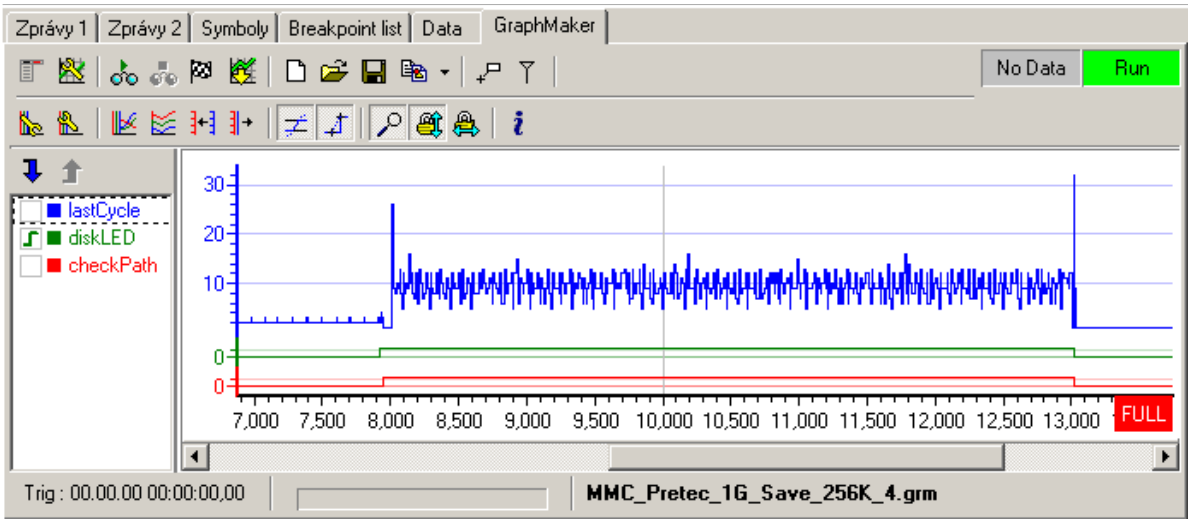
Ve sloupcích BUF1 a BUF2 se pak signalizuje stav obou ukládacích bufferů v DataBoxu:

- STATE stav bufferu (0 = FREE, 1 = ACTIVE, 2 = CLOSED)
- ESTAB datum a čas aktivace bufferu
- ADR_HEAD adresa v DataBoxu, kde jsou uloženy řídící informace bufferu
- ADR_DATA adresa v DataBoxu, kde jsou uloženy záznamy
- DATA_LEN aktuální délka uložených záznamů

Pole FILE SAVING TIME zobrazuje minimální a maximální čas potřebný pro uložení souboru. Pole DIR MAKING TIME zobrazuje časy potřebné pro založení adresářů. A konečně pole PLC CYCLE ukazuje časy cyklů PLC při testu. Rozsvícení pole TEST indikuje

spuštěný test, indikace BUSY signalizuje právě probíhající zápis do souboru a indikace ERROR svítí při chybě zápisu do souboru.

Okno nástroje Graph Maker je připraveno zaznamenat průběh ukládání souboru na kartu. Sleduje se hlavně čas potřebný na ukládání v jednotlivých cyklech PLC (stopa *lastCycle*, čas v milisec). Další zobrazené průběhy ukazují kdy je aktivován zápis do karty (stopa *diskLed*) a kdy jsou vytvořeny potřebné adresáře a probíhá vlastní zápis dat do souboru (stopa *checkPath*) Záznam analyzátoru se spustí stiskem ikony „praporek“ v horní ovládací liště nástroje. Poté je třeba vyčkat, až se uloží soubor na kartu a naplní se buffer analyzátoru v centrální jednotce PLC. Po naplnění analyzátoru se automaticky objeví dialog s výzvou ke stažení nasnímaných dat z PLC do počítače.

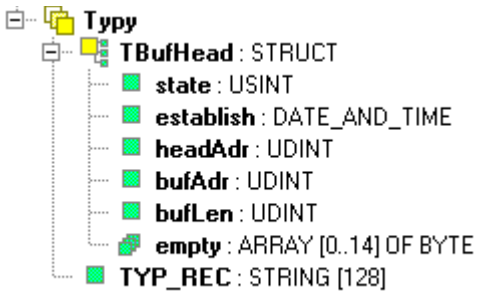


Pro vlastní testy je možné udělat identickou kopii celého projektu volbou Projekt | Kopírovat projekt a zadáním jména pro kopii projektu. Kopii pak lze libovolně modifikovat s tím, že původní projekt zůstane zachován a lze se k němu kdykoliv vrátit.

Následující kapitoly popisují všechny důležité části programu.

2 DATOVÉ TYPY

V příkladu SaveFileToCard jsou definovány následující datové typy:



2.1 Typ TYP_REC

TYP_REC je datový typ odvozený ze základního typu STRING a používá se pro záznam ukládaný do CSV souboru. Jeho velikost lze měnit podle potřeby, maximální velikost je 255 znaků.

2.2 Typ TBufHead

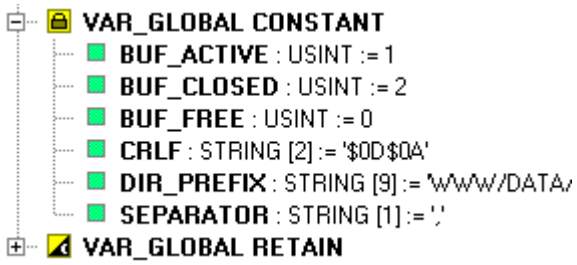
TBufHead je struktura, která je uložena na začátku každého bufferu v DataBoxu a obsahuje informace o aktuálním stavu bufferu.

Význam jednotlivých položek struktury *TBufHead* je následující:

- *state* stav bufferu
- *establish* datum a čas založení bufferu (uložení prvního záznamu)
- *headAdr* adresa této struktury v DataBoxu
- *bufAdr* adresa prvního uloženého záznamu
- *bufLen* aktuální délka uložených záznamů
- *empty* rezerva pro další informace

3 KONSTANTY

V knihovně FileLib jsou definovány následující konstanty:



3.1 Konstanty BUF_ACTIVE, BUF_CLOSED a BUF_FREE

Konstanty *BUF_ACTIVE, BUF_CLOSED a BUF_FREE* jsou typu USINT a používají se jako stav bufferu pro ukládání záznamů v DataBoxu.

Význam a hodnoty konstant jsou následující:

BUF_FREE	*0,* buffer je volný, záznamy z bufferu jsou uloženy v souboru
BUF_ACTIVE	*1,* buffer je aktivní, tj, jsou do něho právě ukládány záznamy
BUF_CLOSED	*2,* buffer je uzavřen, záznamy jsou připraveny pro zápis do souboru

3.2 Konstanta CRLF

Konstanta *CRLF* je typu STRING[2], obsahuje ASCII znaky “návrat vozu” a “nový řádek” a používá se jako oddělovač řádků v záznamech ukládaných do souboru.

3.3 Konstanta SEPARATOR

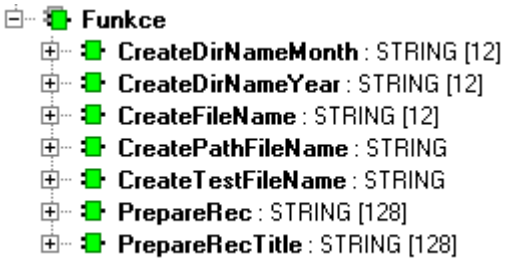
Konstanta *SEPARATOR* je typu STRING[1], obsahuje znak “čárka” a používá se jako oddělovač hodnot v záznamech ukládaných do souboru. Tato konstanta může být v případě potřeby změněna na jiný oddělovač, například znak „tabelátor“.

3.4 Konstanta DIR_PREFIX

Konstanta *DIR_PREFIX* je typu STRING[9], má hodnotu “WWW/DATA/” a udává počátek cesty ve jménech ukládaných souborů. Také tuto konstantu lze změnit podle potřeby.

4 FUNKCE

V příkladu SaveFileToCard jsou definovány následující funkce:



Pro vytváření jmen souborů a adresářů obsahuje příklad následující funkce:

- *CreateDirNameMonth* vytvoří jméno adresáře podle měsíce
- *CreateDirNameYear* vytvoří jméno adresáře podle roku
- *CreateFileName* vytvoří jméno souboru podle datumu
- *CreatePathFileName* vytvoří jméno souboru včetně cesty podle datumu
- *CreateTestFileName* vytvoří jméno testovacího souboru

Pro přípravu ukládaných záznamů obsahuje příklad následující funkce:

- *PrepareRec* připraví záznam ve formátu CSV
- *PrepareRecTitle* připraví hlavičku záznamu ve formátu CSV

Všechny funkce vrací STRING a mohou být modifikovány podle potřeby.

4.1 Funkce PrepareRec

Tuto funkci je nutné přepsat podle konkrétní potřeby.

Funkce **PrepareRec** připraví záznam ve formátu CSV podle aktuálních hodnot vstupních proměnných *temp* a *pres*.

Funkce **PrepareRec** vrátí řetězec začínající časovým údajem, za kterým následují hodnoty proměnných *temp* a *pres*. Jednotlivé údaje jsou odděleny znakem podle konstanty *SEPARATOR*. Celý řetězec je zakončen konstantou *CRLF*.

Funkce PrepareRec :

```
FUNCTION PrepareRec : STRING[128]
(*
   return data record
*)
  VAR_INPUT
    temp    : REAL;         // temperature
    pres    : REAL;         // pressure
  END_VAR
  VAR_TEMP
    xxx     : STRING[16];
    rec     : TYP_REC;
  END_VAR

  // time stamp
  rec := TIME_TO_STRING(GetTime());
  rec := MID(IN := rec, L := 11, P := 3);
  // value separator
  rec := CONCAT(IN1 := rec, IN2 := SEPARATOR);
  // temperature
  xxx := REAL_TO_STRING(temp);
  xxx := DELETE(IN := xxx, L := 4, P := FIND(IN1 := xxx, IN2 := '.') + 2);
  rec := CONCAT(IN1 := rec, IN2 := xxx);
  // value separator
  rec := CONCAT(IN1 := rec, IN2 := SEPARATOR);
  // pressure
  xxx := REAL_TO_STRING(pres);
  xxx := DELETE(IN := xxx, L := 3, P := FIND(IN1 := xxx, IN2 := '.') + 3);
  rec := CONCAT(IN1 := rec, IN2 := xxx);
  // line separator
  PrepareRec := CONCAT(IN1 := rec, IN2 := CRLF);
END_FUNCTION
```

4.2 Funkce PrepareRecTitle

Tuto funkci je nutné přepsat podle konkrétní potřeby.

Funkce **PrepareRecTitle** připraví hlavičku pro záznam ve formátu CSV.

Funkce **PrepareRecTitle** vrátí řetězec: „**`Time,Temperature,Pressure`**“ zakončený konstantou *CRLF*.

Funkce PrepareRecTitle :

```
FUNCTION PrepareRecTitle : STRING[128]
(*
   return record header
*)
  VAR_TEMP
    title  : TYP_REC;
  END_VAR

  // record header
  title := 'Time' + SEPARATOR + 'Temperature' + SEPARATOR + 'Pressure' + CRLF;
  PrepareRecTitle := title;
END_FUNCTION
```

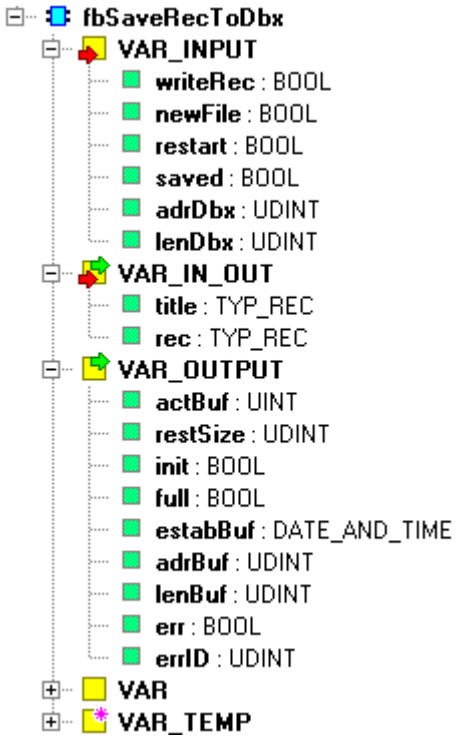
5 FUNKČNÍ BLOKY PRO UKLÁDÁNÍ SOUBORU

V příkladu SaveFileToCard jsou definovány následující funkční bloky:

- *fbSaveRecToDbx* uložení záznamu do bufferu v DataBoxu
- *fbSaveDbxToFile* uložení bufferu z DataBoxu do souboru
- *fbSaveRecToArchive* uložení záznamu do souboru s využitím bufferu v DataBoxu, tento funkční blok využívá oba předchozí bloky

Funkční bloky využívají základní souborové funkce z knihovny FileLib a rozkládají čtení resp. zápis souborů na více cyklů tak, aby se optimalizovala časová náročnost těchto operací.

5.1 Funkční blok fbSaveRecToDbx



Funkční blok **fbSaveRecToDbx** uloží záznam z proměnné *rec* do aktivního bufferu v DataBoxu. Záznamem může být libolný řetězec a jeho uložení se provede, je-li v proměnné *writeRec* hodnota TRUE. Záznam je uložen vždy na konec bufferu (za poslední uložený záznam). Při uložení se zvýší výstupní proměnná *lenBuf* o délku ukládaného řetezce.

V proměnné *title* se očekává hlavička (první záznam) pro formát CSV, který je uložen na začátku bufferu. Je-li tento řetězec prázdný, záznamy budou uloženy bez hlavičky.

Hodnota TRUE v proměnné *newFile* způsobí, že se nejprve uzavře aktivní buffer v DataBoxu a nastaví se výstupní proměnné následovně: proměnná *full* na hodnotu TRUE (indikace, že buffer je připraven na zápis do souboru), proměnná *estabBuf* se nastaví datum a čas založení bufferu, proměnná *adrBuf* obsahuje adresu bufferu, na které začínají uložené záznamy a proměnná *lenBuf* říká aktuální délku uložených záznamů. Poté se jako aktivní buffer označí druhý z dvojice bufferů v DataBoxu. Do hlavičky tohoto bufferu se zapíše datum a čas jeho založení, dále se na začátek bufferu uloží záznam z proměnné *title* a případně se připojí záznam z proměnné *rec* (pokud je proměnná *writeRec* TRUE). Uzavřený buffer čeká na uložení do té doby než se ve vstupní proměnné *saved* objeví hodnota TRUE. Poté se stav uzavřeného bufferu změní na volný buffer a buffer je opět připraven na ukládání záznamů.

Pokud je aktivní buffer naplněn, funkční blok **fbSaveRecToDbx** automaticky tento buffer uzavře stejným způsobem, jako kdyby byla nastavena řídící proměnná *newFile*.

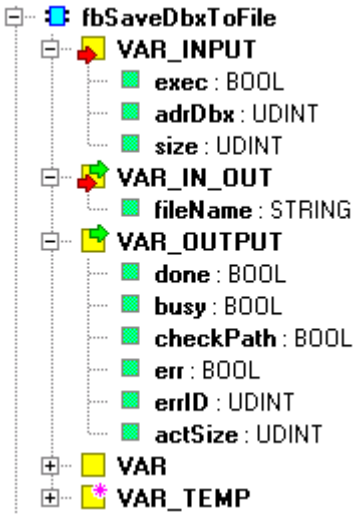
Hodnota TRUE v proměnné *restart* způsobí inicializaci řídících struktur pro buffery v DataBoxu, jako aktivní je označen první buffer, druhý buffer je označen jako volný, data uložená v bufferech jsou ztracena. V DataBoxu jsou založeny dva buffery, první z nich leží na adrese uvedené v proměnné *adrDbx*, proměnná *lenDbx* udává celkovou délku prostoru, který bude pro oba buffery použit. To znamená, že každý z bufferů bude mít velikost *lenDbx/2*.

Při prvním volání instance **fbSaveRecToDbx** po zapnutí napájení se provede inicializace řídících struktur pro buffery v DataBoxu podle aktuálního stavu DataBoxu, takže ukládání záznamů bude pokračovat tam, kde se před vypnutím s ukládáním skončilo.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	writeRec	BOOL	Pořadavek na zápis záznamu do aktivního bufferu v DataBoxu
	newFile	BOOL	Požadavek na uzavření aktivního bufferu a přípravu uzavřeného bufferu na zápis do souboru
	restart	BOOL	Pořadavek na reset bufferů v DataBoxu
	saved	BOOL	Uzavřený buffer byl uložen do souboru
	adrDbx	UDINT	Adresa paměti, kde v DataBoxu leží buffery pro záznamy
	lenDbx	UDINT	Velikost prostoru pro buffery v bytech
VAR_IN_OUT			
	title	TYP_REC	Hlavička ukládaných záznamů
	rec	TYP_REC	Ukládaný záznam
VAR_OUTPUT			
	actBuf	BOOL	Číslo aktivního bufferu (1 nebo 2)
	restSize	UDINT	Velikost volného místa v aktivním bufferu
	init	BOOL	TRUE znamená, že řídící struktury pro buffery byly inicializovány a funkční blok je připraven k použití
	full	BOOL	TRUE indikuje, že některý z bufferů v DataBoxu byl uzavřen a je připraven pro zápis do souboru
	estabBuf	DT	Datum a cas založení uzavřeného bufferu Při *full* = *FALSE* má *estabBuf* hodnotu DT#1970-01-01-00:00:00
	adrBuf	UDINT	Adresa DataBoxu, odkud leží záznamy v uzavřeném bufferu Při *full* = *FALSE* má *adrBuf* hodnotu 0
	lenBuf	UDINT	Délka záznamů uložených v uzavřeném bufferu Při *full* = *FALSE* má *lenBuf* hodnotu 0
	err	BOOL	Příznak chyby při ukládání záznamu do bufferu v DataBoxu
	errID	UDINT	Číslo chyby (0 znamená bez chyby)

5.2 Funkční blok fbSaveDbxToFile



Funkční blok **fbSaveDbxToFile** zapíše na náběžnou hranu proměnné *exec* blok paměti z DataBoxu do souboru. Adresu začátku bloku paměti udává proměnná *adrDbx*, délka zapisovaných dat je dána proměnnou *size*. Jméno souboru (včetně cesty), do kterého budou data uložena, udává proměnná *fileName*. Zápis dat do souboru neproběhne v jediném cyklu PLC, počet cyklů závisí na délce zapisovaných dat. Pokud neexistují adresáře uvedené v proměnné *fileName*, funkční blok **fbSaveDbxToFile** tyto adresáře založí. Pokud je na kartě uložen soubor stejného jména jako soubor v proměnné *fileName*, obsah původního souboru je přepsán novými daty. Během zápisu do souboru je nastavena výstupní proměnná *busy* na hodnotu TRUE. Proměnná *actSize* udává kolik bytů už bylo do souboru zapsáno. Pokud jsou všechna data úspěšně zapsaná do souboru a soubor je uzavřen nastaví se výstupní proměnná *done*.

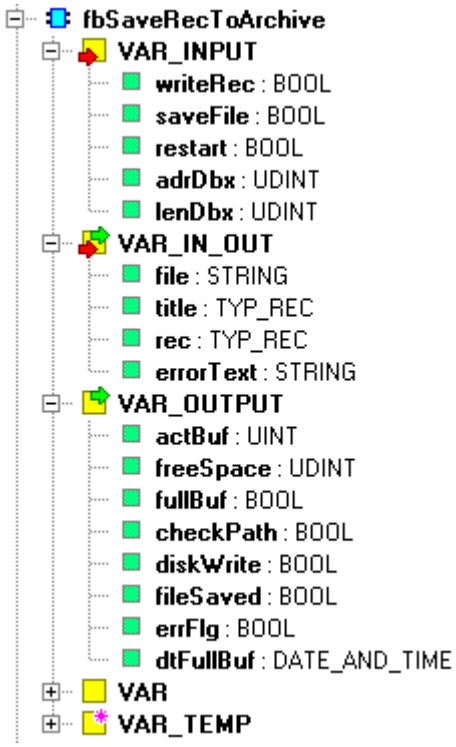
Instance funkčního bloku **fbSaveRecToDbx** je použita ve funkčním bloku **fbSaveRecToArchive.**

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	exec	BOOL	Řídící proměnná. Náběžná hrana (přechod z hodnotu FALSE na hodnotu TRUE) zahájí zápis do souboru
	adrDbx	UDINT	Adresa DataBoxu, odkud jsou uložena data
	size	UDINT	Velikost zapisovaných dat (počet bytů)
VAR_IN_OUT			
	fileName	STRING	Jméno souboru, do kterého budou data uložena (včetně cesty)

	Proměnná	*Typ*	*Význam*
VAR_OUTPUT			
	done	BOOL	Má hodnotu TRUE v okamžiku dokončení zápisu do souboru. Jinak vrací FALSE
	busy	BOOL	Má hodnotu TRUE během zápisu do souboru
	checkPath	BOOL	Nastaví se na hodnotu TRUE v okamžiku, kdy existují všechny potřebné adresáře pro zápis souboru (pro účely testování)
	err	BOOL	Příznak chyby při zápisu do souboru Pokud operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	errID	UDINT	Chybový kód. Pokud operace dopadla úspěšně, errID = 0. V případě chyby errID <> 0.
	actSize	UDINT	Průběžný počet skutečně zapsaných bytů

5.3 Funkční blok fbSaveRecToArchive



Funkční blok **fbSaveDbxToArchive** používá bloky **fbSaveRecToDbx** a **fbSaveDbxToFile** pro ukládání záznamů do souboru s využitím paměti DataBox pro přechodné uložení dat. Vlastnosti tohoto bloku jsou tedy shodné s vlastnostmi použitých bloků.

Funkční blok **fbSaveDbxToArchive** uloží záznam z proměnné *rec* do aktivního bufferu v DataBoxu v případě, že v proměnné *writeRec* je hodnota TRUE. Záznam je uložen vždy na konec bufferu (za poslední uložený záznam). Při uložení se sníží výstupní proměnná *freeSpace* o délku ukládaného řetezce.

V proměnné *title* se očekává hlavička (první záznam) pro formát CSV, který je uložen na začátku bufferu. Je-li tento řetězec prázdný, záznamy budou uloženy bez hlavičky.

Hodnota TRUE v proměnné *newFile* způsobí, že se nejprve uzavře aktivní buffer v DataBoxu a nastaví se výstupní proměnné následovně: proměnná *fullBuf* na hodnotu TRUE (indikace, že buffer je ukládán do souboru) a proměnná *dtFullBuf* obsahuje datum a čas založení bufferu. Vlastní zápis do souboru je zahájen se zpožděním jednoho cyklu po nastavení *fullBuf,* aby hlavní program mohl připravit jméno souboru do proměnné *fileName*. Jako aktivní buffer je označen druhý z dvojice bufferů v DataBoxu. Do hlavičky tohoto bufferu se zapíše datum a čas jeho založení, dále se na začátek bufferu uloží záznam z proměnné *title* a případně se připojí záznam z proměnné *rec* (pokud je proměnná *writeRec* TRUE). Číslo právě aktivního bufferu je v proměnné *actBuf*.

Pokud je aktivní buffer naplněn, funkční blok **fbSaveDbxToArchive** automaticky tento buffer uzavře stejným způsobem, jako kdyby byla nastavena řídící proměnná *newFile*.

Hodnota TRUE v proměnné *restart* způsobí inicializaci řídících struktur pro buffery v DataBoxu, jako aktivní je označen první buffer, druhý buffer je označen jako volný, data uložená v bufferech jsou ztracena. V DataBoxu jsou založeny dva buffery, první z nich leží na adrese uvedené v proměnné *adrDbx*, proměnná *lenDbx* udává celkovou délku prostoru, který bude pro oba buffery použit. To znamená, že každý z bufferů bude mít velikost *lenDbx/2*.

Funkční blok **fbSaveDbxToArchive** automaticky zapíše uzavřený buffer do souboru. Jméno souboru (včetně cesty), do kterého budou data uložena, udává proměnná *fileName*. Zápis dat do souboru neproběhne v jediném cyklu PLC, počet cyklů závisí na délce zapisovaných dat. Pokud neexistují adresáře uvedené v proměnné *fileName*, funkční blok **fbSaveDbxToArchive** tyto adresáře založí. Pokud je na kartě uložen soubor stejného jména jako soubor v proměnné *fileName*, obsah původního souboru je přepsán novými daty. Během zápisu do souboru je nastavena výstupní proměnná *diskWrite* na hodnotu TRUE. Pokud jsou všechna data úspěšně zapsaná do souboru a soubor je uzavřen nastaví se výstupní proměnná *fileSaved*.

Při prvním volání instance **fbSaveDbxToArchive** po zapnutí napájení se provede inicializace řídících struktur pro buffery v DataBoxu podle aktuálního stavu DataBoxu, takže ukládání záznamů bude pokračovat tam, kde se před vypnutím s ukládáním skončilo.

Pokud se napájení systému vypnulo během ukládání dat z uzavřeného bufferu do souboru, je při opětovném zapnutí napájení tento buffer znovu celý uložen do souboru. Aplikační program musí v tomto případě zajistit, aby jméno souboru v proměnné *file* bylo shodné. K tomu lze využít výstupní proměnnou *dtFullBuff*, která obsahuje datum a čas založení právě ukládaného bufferu.

V případě chyby při zápisu do souboru je nastavena proměnná *errFlg* na hodnotu TRUE a v proměnné *errorText* je chybové hlášení.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	*writeRec*	BOOL	Pořadavek na zápis záznamu do aktivního bufferu v DataBoxu
	*newFile*	BOOL	Požadavek na uzavření aktivního bufferu a zápis uzavřeného bufferu do souboru
	*restart*	BOOL	Pořadavek na reset bufferů v DataBoxu
	*adrDbx*	UDINT	Adresa paměti, kde v DataBoxu leží buffery pro záznamy
	*lenDbx*	UDINT	Velikost prostoru pro buffery v bytech
VAR_IN_OUT			
	*file*	STRING	Jméno souboru včetně cesty
	*title*	TYP_REC	Hlavička ukládaných záznamů
	*rec*	TYP_REC	Ukládaný záznam
	*errText*	STRING	Chybové hlášení

	Proměnná	*Typ*	*Význam*
VAR_OUTPUT			
	actBuf	BOOL	Číslo aktivního bufferu (1 nebo 2)
	freeSpace	UDINT	Velikost volného místa v aktivním bufferu
	fullBuf	BOOL	TRUE indikuje, že některý z bufferů v DataBoxu byl uzavřen a probíhá zápis do souboru
	checkPath	BOOL	Nastaví se na hodnotu TRUE v okamžiku, kdy existují všechny potřebné adresáře pro zápis souboru (pro účely testování)
	diskWrite	BOOL	Hodnotu TRUE indikuje probíhající zápis do souboru
	fileSaved	BOOL	Hodnotu TRUE indikuje ukončený zápis do souboru
	errFlg	BOOL	Příznak chyby při ukládání záznamu do souboru
	dtFullBuf	DT	Datum a cas založení uzavřeného bufferu
	err	BOOL	Příznak chyby při ukládání záznamu do bufferu v DataBoxu

6 PŘÍLOHY

6.1 Testování MMC a SD karet

Podmínky testování

Délka zapisovaného souboru 256 KB. Soubor byl ukládán do cesty WWW/DATA/2008/02. Počet zápisů při testování 16. Testovací program nejprve založil potřebné adresáře a poté zapsal 16 souborů s délkou 256 KB. V jednom cyklu PLC programu byl ukládán max. jeden sektor, tj. 512 bytů dat. Všechny časy uvedené v tabulce jsou v milisekundách.

Testované karty

Typy a velikosti testovaných karet byly vybrány s ohledem otestovat co nejširší škálu různých karet. Karty tedy měly různou kapacitu, karty s nižšími kapacitami byly zpravidla historicky starší.

Sledované parametry

Sledován byl maximální čas potřebný pro vytvoření adresářů, minimální a maximální čas potřebný pro uložení jednoho souboru a konečně maximální čas jednoho cyklu PLC. Testy byly provedeny na CP-7004 s firmwarem v3.0. Naměřené hodnoty platí i pro systém Foxtrot, neboť hardware procesorové jednotky je prakticky stejný.

Na rozdíl od parametrů udávaných výrobci, kteří udávají v zásadě průměrnou rychlost karty (průměrný datový tok), zajímá uživatele PLC špičková spotřeba času především pro operaci zápis do karty. O tento čas se totiž může zvýšit doba cyklu PLC, který zapisuje soubor do karty. S tímto navýšením doby cyklu PLC je třeba počítat při řízení technologie.

Cíle testování

Ověřit funkčnost ovladačů pro MMC/SD karty, ověřit funkčnost knihovny pro souborové operace, porovnat vlastnosti jednotlivých paměťových karet. Dalším cílem bylo vytvořit soubor údajů, se kterými lze porovnat parametry naměřené na novém, ještě netestovaném, typu karty a zjistit tak, jestli je nová karta vhodná pro použití v PLC či nikoliv.

Vyhodnocení testů

Testované karty vykazují značné rozdíly. Nevhodné formátování karty může výrazně ovlivnit použitelnost karty v PLC systému (viz výsledky pro kartu Corsair). Pomyslným vítězem klání se stala SD karta Cheetah 133x 1G výrobce Pretec, která vykazovala dobrý celkový čas zápisu, přičemž maximální časy přidané k době cyklu při zápisu byly na úrovni 65 milisekund. Typicky to pak bylo kolem 7 milisekund na jeden cykl, ve kterém probíhal zápis. Velmi dobře si vedla i karta MMC MyFlash 128 MB výrobce A-Data, nevýhodou je však kapacita karty.

Neprobarvené řádky tabulky byly naměřeny první verzí testovacího SW, takže mohou obsahovat mírně horší hodnoty, než by se naměřily s finální verzí SW.

KARTA	*VÝROBCE*	*KAPA-CITA*	*SAVING TIME MIN*	*SAVING TIME MAX*	*CYCLE MIN*	*CYCLE MAX*	*CREATE DIR TIME*
SD Ultra II.	SanDisk	2 GB	1793	1823	1	41	

KARTA	*VÝROBCE*	*KAPA-CITA*	*SAVING TIME MIN*	*SAVING TIME MAX*	*CYCLE MIN*	*CYCLE MAX*	*CREATE DIR TIME*
SD	SanDisk	2 GB	1635	1745	1	79	310
SD	Kingston	2 GB	2146	13094	1	331	192
SD	Kingston	1 GB	2233	2917	1	123	191
SD ElitePro 50x	Kingston	1 GB	2246	2284	1	49	93
SD HighSpeed 60x	takeMS	1 GB	1902	2258	1	171	540
SD	EagleTec	1 GB	1993	2039	1	139	3540
SD Speedy Duo	A-data	1 GB	3160	4796	1	409	409
SD Cheetah 133x	Pretec	1 GB	1648	1992	1	65	65
SD 133x (FAT16, cluster 16 KB)	Corsair	1 GB	1341	1410	1	61	85
SD 133x (FAT16, cluster 32 KB)	Corsair	1 GB	1295	1375	1	80	1710
SD 133x (FAT32, cluster 4 KB)	Corsair	1 GB	1550	2312	1	232	859
SD	Kingston	512 MB	2780	4118	1	381	381
SD Cheetah 133x	Pretec	256 MB	1554	1610	1	28	195
SD	SanDisk	128 MB	2346	2606	1	221	1860
SD	SanDisk	32 MB	1805	1871	1	49	782
SD	Panasonic	32 MB	7929	8093	1	95	465
SD	Panasonic	16 MB	6930	8356	1	81	58
SD	Toshiba	16 MB	12436	12589	1	134	110
Micro SD + adapter	Kingston	1 GB	1850	2672	1	221	221
Micro SD + adapter	Kingston	2 GB	1826	5823	1	188	188
MMC Mobile	Kingston	2 GB	2768	3032	1	155	296
MMC Mobile	Pretec	1 GB	5271	6798	1	215	229
MMC Mobile	Kingston	512 MB	1819	2748	1	439	267
MMC MyFlash	A-data	128 MB	1486	2270	1	42	56

Obr.1Testované karty