Archivy

OBJEKTY PRO SPRÁVU DAT

Příručka uživatele a programátora

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

Verze dokumentu 1.14

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

Datum vydání: 19.08.2003
Datum posledního uložení dokumentu: 19.08.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.00	Hv	08.11.2002	První vydání.
1.10	1.XX	Hv	05.02.2003	Úprava dokumentu dle ISO9000. Přesnější popisy. Popis nových chyb v konstruktoru (ae_IncompParams, ae_AStructCrc, ae_BaseAddr). Popis nového flagu af_InitArchive. Oprava vývojového diagramu. Zavedení jednotné verze programového balíku v jednotce ATypes pod identifikátorem cVer (smazány dosavadní cName a cVer).
1.11	1.XX	Hv	11.02.2003	Oprava vývojového diagramu popisujícího konstruktor archivů – kontrola parametrů.
1.12	1.XX	Hv	04.04.2003	Oprava hlaviček funkcí a jména jednotky.
1.13	1.XX	Tu	20.5.2003	Úprava dokumentu dle ISO9000.
1.14	1.XX	Hv	19.08.2003	Úprava interface objektu z důvodu přechodu na rozšířenou standardní knihovnu Crc16. Dřívější verze používaly ExCrc16.

1.2. Účel dokumentu

Tento dokument slouží jako popis programového balíku knihoven Archive používaného pro ukládání a zpracování dat.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu není potřeba číst žádný další manuál, ale je potřeba orientovat se v používání programového vybavení SofCon.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

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

3. Úvod

Balík knihoven archivů je určen pro řídicí systémy KIT a slouží pro ukládání a zpracování dat. Data jsou ukládána v zadané struktuře a mohou být ukládána průběžně v pevně daném intervalu nebo v důsledku vzniklých událostí. Při jejich zápisu se s těmito daty ukládá datum a čas, který je v současné verzi ve formátu DOS o velikosti 4B.

Knihovny vznikly na základě dlouholetých zkušeností a při jejich vývoji byla zohledněna možnost snadného ladění na počítači PC.

Knihovny se snaží o max. zjednodušení rutinních prací při ukládání, zpracování a čtení dat, takže v některých případech se práce s daty omezí na deklarace struktur záznamů, umístění archivů v paměti a předefinování funkcí pro zápis dat do archivu.

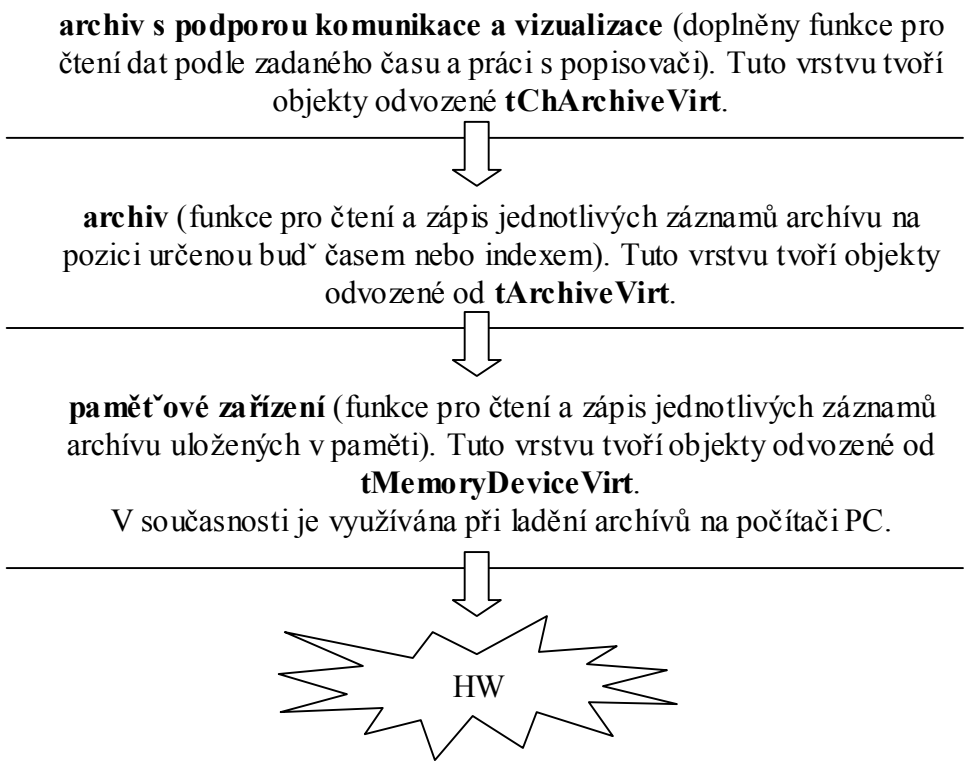
Protože vizualizace dat je v převážné většině případů prováděna pomocí programu TheKing existuje nad těmito knihovnami nadstavba, která implementuje vlastní komunikaci, tj. komunikační automaty. Tato nadstavba je popsána v samostatném manuálu.

Program TheKing lze spustit na počítači s operačním systémem Windows 98 a novější (Windows XP). Popis tohoto programu najdete v samostatném manuálu.

Pozn. U obrázků používaných pro popis odvozování objektů je v dolní části napravo uveden název knihovny, ve které je umístěn typ objektu nad tímto názvem knihovny.

4. Struktura knihoven pro správu a zpracování dat

Knihovny archivů lze rozdělit do několika vrstev. Tyto vrstvy jsou zachyceny na následujícím obrázku.



4.1. Vrstva HW

Tato vrstva představuje vlastní hardware (paměť), pod kterým si můžete představit statickou paměť, paměť FLASH případně paměť nad dostupnou 1MB pomocí knihoven DiskIO.

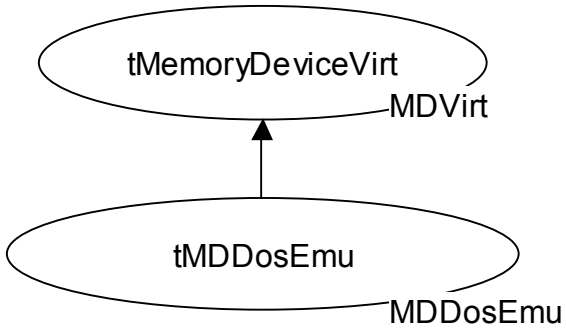
4.2. Vrstva paměťového zařízení

Tato vrstva slouží pro odstínění přístupů k paměti od objektu tArchiveVirt spravujícího záznamy.

Objekty této vrstvy vytváří jednotné rozhraní pro přístup k různým typům paměti - např.paměti nad 1MB a emulace paměti na počítači PC. (Pozn. na počítači PC se je paměť emulována, protože Borland Pascal nedovolí v reálném módu přistupovat k bloku paměti většímu než 64kB.)

V rámci zrychlení přístupů k záznamům archivu můžou objekty odvozené od tArchiveVirt tuto vrstvu obcházet a přistupovat rovnou na vrstvu HW. Způsob přístupu k paměti je určen pomocí příznaku při inicializaci objektu odvozeného od tArchiveVirt.

V této vrstvě jsou implementovány následující objekty:



- Základní objekt tMemoryDeviceVirt vytváří rozhraní pro objekty odvozené od tArchiveVirt. Mezi tyto funkce patří čtení a zápis záznamů, práce s CRC a zjištění poslední chyby.
- Objekt tMDDosEmu slouží pro emulaci souvislé paměti přesahující 64KB na počítači PC. Zároveň tento objekt řeší problém čtení a ukládání záznamů archivu, protože tato paměť je vytvořena pomocí souborů.

  Pokud při ladění archivu nechcete tento objekt použít, musí se v aplikaci přeložené pro spuštění na PC (viz podmíněné překlady) vyřešit rezervace paměti archivu a poté ukládání a obnova dat při spuštění aplikace.

  Pozn. při použití paměti HEAP nemáte zaručeno zachování dat a dva alokované bloky nemusí na sebe navazovat. Max. velikost jednoho alokovaného bloku je 64kB.

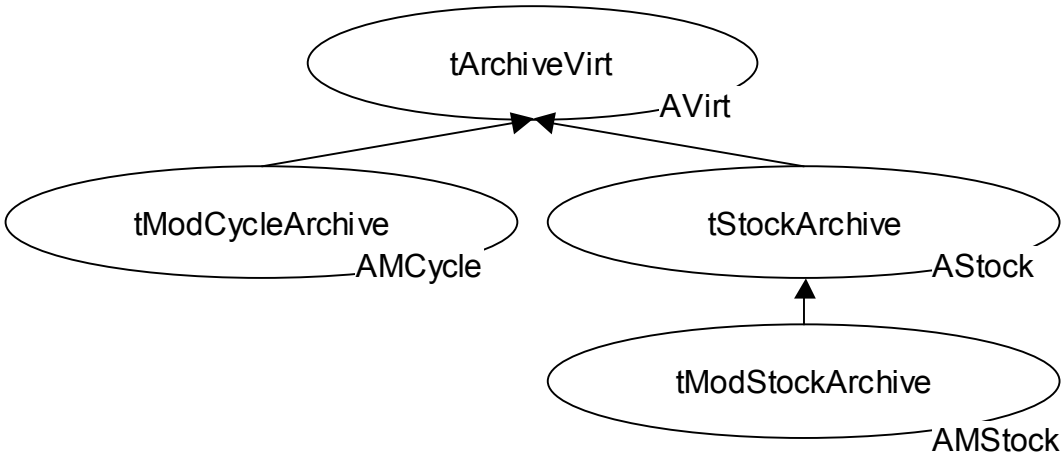
4.3. Vrstva archivy

V této vrstvě jsou implementovány objekty odvozené od tArchiveVirt. Tyto objekty vytváří jednotné rozhraní pro práci s daty aplikace, dále záznamy, které mají být zachovávány nezávisle na běhu aplikace tzn. i v případě nového spuštění aplikace.

Tyto záznamy mohou být ukládány postupně (v časové posloupnosti) nebo pomocí indexu do paměti uspořádané do kruhového bufferu nebo skladu.

Funkce rozhraní řeší synchronizační problémy pomocí zamykání jádra, což je dostačující při práci se záznamy v různých procesech. V případě zápisu záznamů v přerušovací rutině se synchronizace musí provést pomocí zákazu přerušení.

V této vrstvě jsou implementovány následující objekty:



- Základní objekt této vrstvy tArchiveVirt pracuje s pamětí uspořádanou do kruhového bufferu. Záznamy lze do paměti ukládat nebo číst prostřednictvím paměťového zařízení nebo přímo. Poskytované rozhraní umožňuje záznamy do paměti ukládat postupně nebo pomocí indexu.

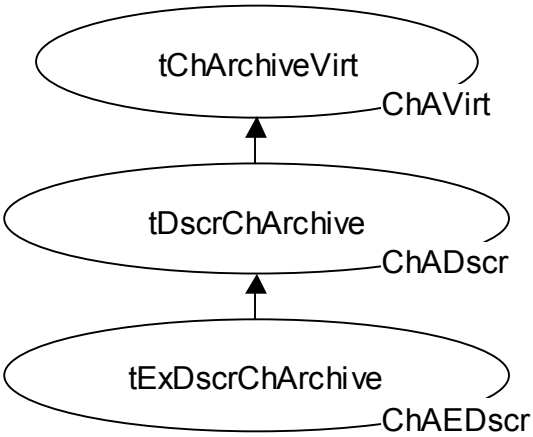
  Dále jsou poskytovány funkce pro inicializaci, kontrolu archivu, zjišťování posledního chybového stavu a konfiguračních hodnot archivu, např. velikost a počet záznamů.

- Objekt tModCycleArchive pracuje s pamětí uspořádanou do kruhového bufferu, ale po přečtení záznamu zavolá uživatelsky nastavitelnou funkci. Tato funkce umožňuje upravit záznam do požadovaného tvaru, např. převod číselné položky záznamu na textovou položku, která má být zobrazovaná v programu TheKing.

- Objekt tStockArchive pracuje s pamětí uspořádanou do skladu. Po zaplnění paměti skladu se vrací chybový kód.

- Objekt tModStockArchive pracuje s pamětí uspořádanou do skladu. Po zaplnění paměti skladu se vrací chybový kód. Na rozdíl od svého předka po přečtení záznamu zavolá uživatelsky nastavitelnou funkci. Tato funkce umožňuje upravit záznam do požadovaného tvaru, např. převod číselné

položky záznamu na textovou položku, která má být zobrazovaná v programu TheKing.



4.4. Vrstva archivů s podporou komunikace a vizualizace

V této vrstvě jsou implementovány objekty odvozené od tChArchiveVirt. Tyto objekty vytváří jednotné rozhraní pro další vrstvu komunikující s programem TheKing. Mezi základní funkce tohoto rozhraní patří metody pro práci s popisovači zobrazení záznamů pomocí objektů tDscr a tExDscr.

Tato vrstva vždy pracuje nad vytvořenými objekty odvozenými od tArchiveVirt. Zatímco objekty spodní vrstvy (odvozené od tArchiveVirt) umí ukládat a číst jednotlivé záznamy podle absolutně zadaného indexu (vždy od začátku zadané oblasti) případně ukládat záznamy postupně za sebou, umí objekty této vrstvy číst více záznamů na jednou v zadaném časovém intervalu.

V této vrstvě jsou implementovány následující objekty:

Základní objekt této vrstvy tChArchiveVirt slouží pro vytvoření rozhraní používaného knihovnami KitKing. Metody tohoto rozhraní neumí pracovat s popisovači zobrazení dat a musí být vždy přetíženy.

Potomek případně instance tohoto objektu je vhodná pro zobrazování a hledání dat v záznamech archivu přímo na terminálu.

- Objekt tDscrChArchive implementuje funkce pracující s popisovačem tabulek.

  Potomek případně instance tohoto objektu je vhodná pro zobrazení dat v tabulce programu TheKing.

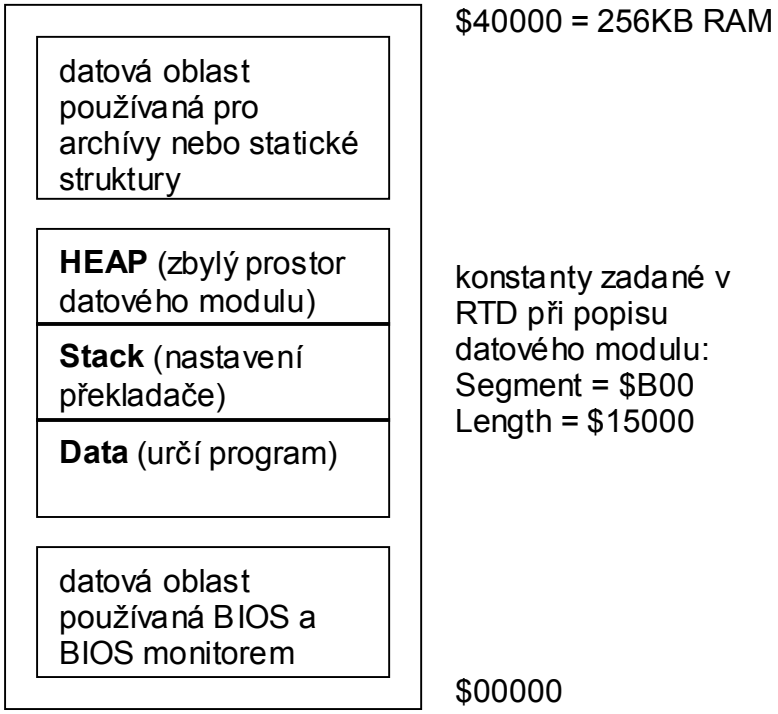
- Objekt tExDscrChArchive implementuje funkce pracující s popisovačem grafu.

  Potomek případně instance tohoto objektu je vhodná pro zobrazení dat v tabulce programu TheKing.

5. Paměťový prostor řídicího systému KIT

V této kapitole budou zopakovány některé základní informace týkající se práce s pamětí v řídicím systému KIT. Podrobnější informace najdete v manuálech týkajících se programového vybavení řídicích systémů KIT, např. BP a ReTOS jak na to, RTD, ReTOS apod.

V dále popsaném příkladu se vychází z následujících předpokladů. Řídicí systém KIT má statickou paměť o velikosti 256kB. V této paměti je vytvořen datový modul o velikosti $15000B (tj. 86016B) začínající na lineární adrese $B00 (tj. 2816). Popis tohoto modulu se zadává v RTD a musí být shodně nadefinován v souboru AppDesc. Popis položek, které musíte editovat a kde získáte jeho šablonu je uveden v dokumentu Začínáme vizualizovat.



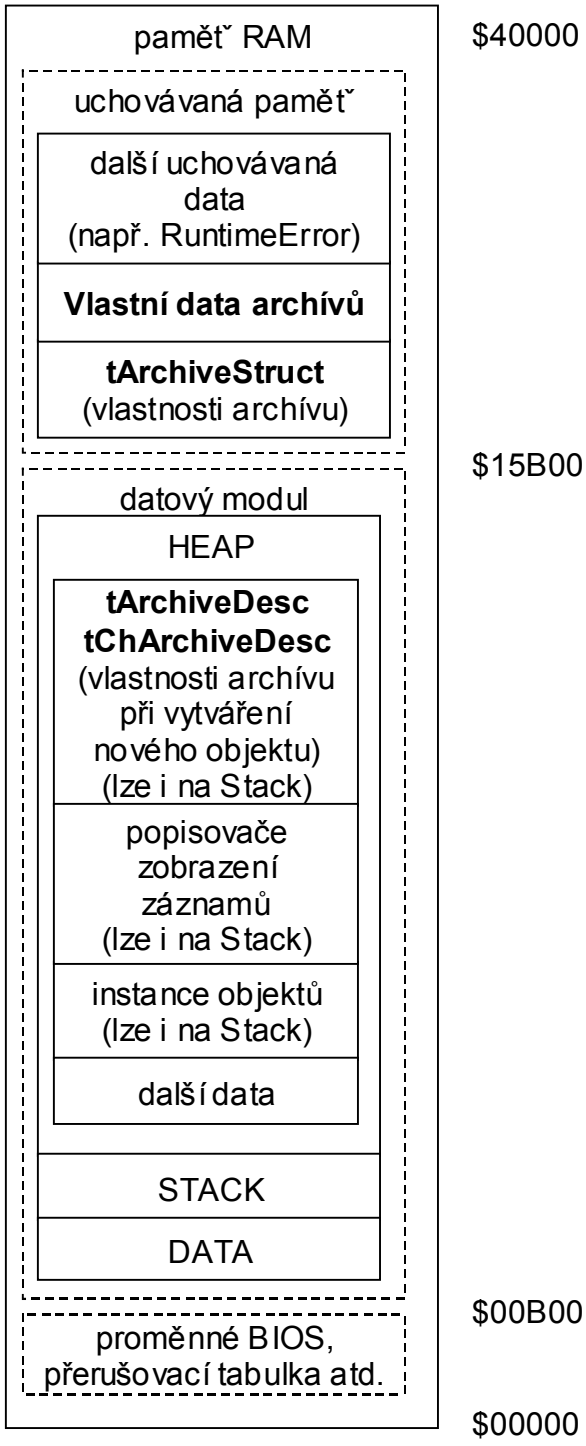
Pozn. Je doporučováno začít aplikaci vytvořením adresářové struktury projektu a poté upravit tento soubor.

Při určení paměťového prostoru pro archiv je důležité si uvědomit, že neměnná (stálá) data nelze vytvářet ani uchovávat na HEAP. Důvodem proč to nelze, je že jen stěží zaručíte pokaždé stejný paměťový prostor a souvislou oblast větší než 64kB. Proto se tento paměťový prostor rezervuje mimo datový modul. V této oblasti totiž používání paměti řídíte sami.

Pokud byste chtěli s touto pamětí pracovat sami, museli byste realizovat veškeré operace pro čtení a zápisy záznamů pomocí ukazatelů. Pokud, ale použijete tento balík knihoven budete ve většině případů používat metody dodaných objektů. Pouze ve speciálních případech budete muset vytvářet potomky dodaných objektů, ale téměř v nulovém procentu případů budete muset pracovat s ukazateli.

Jak vyplývá z předchozího obrázku, velikost HEAP není určena překladačem, ale je určena zbývajícím prostorem v datovém modulu. Tato velikost je dána velikostí jednotky Data a Stack, kterou nelze ovlivnit bez změny programu.

Jak už bylo uvedeno výše paměťový prostor pro Archivy se určí v oblasti nad datovým modulem. V této paměti se také musí alokovat struktury popisující vlastnosti archivů. Při inicializaci objektů a implementací popisovačů zobrazení se používají další struktury. Následující obrázek zachycuje rozmístění těchto struktur v paměti řídicího systému KIT.



Paměť RAM lze rozdělit do tří oblastí, jejichž velikost je buď určena nebo lze nastavit pomocí RTD případně osazenou pamětí RAM v řídicím systému KIT. Výše uvedený popis rozložení paměti platí pro osazení paměti RAM 256kB a nastavenou velikostí datového modulu na $15000.

- První oblast <$00000, $00B00)

  V této oblasti je uložena přerušovací tabulka, proměnné BIOS a oblast, která se používá při komunikaci pomocí BIOS Monitoru.

- Druhá oblast <$00B00, $15B00)

  Velikost této oblasti je určena v programu RTD a v tohoto modulu se umisťují jednotky Data, Stack a HEAP.

  Velikost jednotky Data je dána programem (inicializované konstanty, globální proměnné apod.)

  Velikost jednotky Stack je dána nastavením překladače případně direktivou programu.

  Velikost HEAP je dána zbylým prostorem v datovém modulu.

- Třetí oblast <$15B00, $40000)

  Velikost této oblasti je dána velikostí osazené paměti RAM, velikostí a umístěním datového modulu.

  Do této paměti se umisťují Archivy a stálá data dle Vašich požadavků.

6. Paměťová zařízení

6.1. Knihovna MDTypes

V knihovně jsou definovány chybové kódy, identifikační kódy paměťových zařízení a inicializační struktura.

6.1.1. Konstanty

Chybové kódy

`emd_NoError = 0;`
: Při práci s paměťovým zařízením nedošlo k žádné chybě.

`emd_InvalidParams = 1;`
: Při inicializaci byly paměťovému zařízení předány neplatné parametry.

`emd_NotInit = 2;`
: Práce s neinicializovaným paměťovým zařízením.

`emd_DamagedMd = 3;`
: Při čtení nebo zápisu hodnoty z paměťového zařízení došlo k chybě. Nejspíše je poškozené paměťové zařízení.

`emd_AllocError = 4;`
: Při vytváření paměťového zařízení došlo k chybě způsobené nedostatkem HEAP.

Identifikační kódy objektů paměťových tříd

`md_Virt = 1`
: Identifikační kód virtuálního předka paměťové třídy.

```
md_DosEmulation = 2
```

Identifikační kód paměťové třídy emulující Archivy na počítači PC.

6.1.2. Typy

```
PMemoryDeviceDesc = ^TMemoryDeviceDesc;
TMemoryDeviceDesc = record
 wMask:Word;                  { rezervovano }
 strFileName:FNameStr;
 wBuffSize:Word;
end;
```

Struktura se používá při inicializaci paměťových zařízení.

6.2. Knihovna MDVirt

V knihovně je definován předek všech paměťových zařízení. Tento předek vytváří rozhraní s objekty odvozenými od objektu tArchiveVirt.

6.2.1. Objekt tMemoryDeviceVirt

Objekt je používán jako předek všech paměťových zařízení vytvářejících rozhraní s objekty odvozenými od objektu tArchiveVirt.

6.2.1.1. Položky

```
m_wLastError                : Word;
```

Položka obsahuje kód poslední chyby při práci s paměťovým zařízením.

6.2.1.2. Metody

6.2.1.2.1. Konstruktor Init

```
constructor Init(var MdDesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek. Položky objektu jsou inicializovány následovně.

Položka m_wLastError je nastavena na hodnotu emd_NoError.

6.2.1.2.2. Destruktor Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu.

6.2.1.2.3. Get

```
function Get(laBase, Ix:LongInt; pBuff:Pointer):Boolean; virtual
```

Funkce přečte do datového bufferu pBuff hodnotu, která je určena lineární adresou laBase a indexem Ix.

Funkce vrací TRUE, pokud požadovaná hodnota byla zapsána do bufferu. V opačném případě vrací FALSE.

Funkce vždy vrací FALSE a položka m_wLastError je nastavena na hodnotu emd_NotInit.

6.2.1.2.4. Put

```
function Put(laBase, Ix:LongInt; pBuff:Pointer):Boolean; virtual;
```

Funkce zapíše hodnotu v datovém bufferu pBuff do paměti určené lineární adresou laBase a indexem Ix.

Funkce vrací TRUE, pokud požadovaná hodnota byla zapsána do paměti. V opačném případě vrací FALSE.

Funkce vždy vrací FALSE a položka m_wLastError je nastavena na hodnotu emd_NotInit.

6.2.1.2.5. Create

```
function Create:Boolean; virtual;
```

Funkce dokončí inicializaci paměťového zařízení a uvede ho do provozu.

Funkce vrací TRUE, pokud inicializace proběhla bez chyby. V opačném případě se vrací FALSE.

Funkce vždy vrací TRUE.

6.2.1.2.6. CheckCrc

```
function CheckCrc(wResidum, wCrc:Word):Boolean; virtual;
```

Funkce zkontroluje zda hodnoty uložené v paměťovém zařízení mají požadované CRC wCrc při zadané počáteční hodnotě wResidum.

Funkce vrací TRUE, pokud požadované CRC wCrc je stejné jako CRC všech hodnot v paměťovém zařízení.

Funkce vždy vrací FALSE a položka m_wLastError je nastavena na hodnotu emd_NotInit.

6.2.1.2.7. CalculateCrc

```
function CalculateCrc(wResidum:Word):Word; virtual;
```

Funkce vypočte CRC pro hodnoty uložené v paměťovém zařízení při zadané počáteční hodnotě wResidum.

6.2.1.2.8. SetLastError

```
procedure SetLastError(wCode:Word); virtual;
```

Procedura nastaví kód chyby v paměťovém zařízení.

6.2.1.2.9. GetLastError

```
function  GetLastError:Word; virtual;
```

Funkce vrací kód poslední chyby v paměťovém zařízení.

6.2.1.2.10. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci typu paměťového zařízení.

Funkce vždy vrací md_Virt.

6.3. Knihovna MDDosEmu

V knihovně je definováno paměťové zařízení, které na počítači PC umožní vytvořit Archivy s pamětí větší než 64kB. Tyto Archivy jsou vytvořeny pomocí souborů spravovaných objektem tBufStream.

Pokud se při ladění archivů na počítači PC použije tento objekt nemusí se dále řešit problém s čtením a ukládáním dat při spuštění a ukončení aplikace.

6.3.1. Objekt tMDDosEmu

Objekt vytváří paměťové zařízení s pamětí větší jak 64kB a zajišťuje čtení a ukládání dat při spuštění a ukončení aplikace.

6.3.1.1. Položky

`m_strFileName : FNameStr;`

Jméno souboru, které se bude používat při emulaci paměti.

`m_wBuffSize : Word;`

Velikost bufferu, který se používá při zápisu a čtení hodnot ze souboru.

`m_BufStream : TBufStream;`

V položce je instance objektu typu Stream pracující se souborem, ve kterém se emuluje paměť.

6.3.1.2. Metody

6.3.1.2.1. Konstruktor Init

`constructor Init(var MdDesc);`

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek. Položky objektu jsou inicializovány následovně.

Při inicializaci se volá konstruktor předka, který nastaví položku m_wLastError na hodnotu emd_NoError. Dále se nastaví položky m_strFileName a m_wBuffSize na hodnoty dle struktury MdDesc. Pokud jsou předány platné hodnoty, dokončí se inicializace paměťového zařízení voláním metody Create. V těle této metody se dokončí inicializace paměťového zařízení.

Hodnota m_wBuffSize by měla odpovídat celočíselnému násobku velikosti záznamů, které se budou do souboru ukládat.

Pokud jsou předané položky ve struktuře MdDesc neplatné, nastaví se proměnná m_wLastError na emd_InvalidParams a metoda Create se nevolá.

6.3.1.2.2. Get

```
function Get(laBase, Ix:LongInt; pBuff:Pointer):Boolean; virtual
```

Funkce přečte do datového bufferu pBuff hodnotu, která je určena lineární adresou laBase a indexem Ix.

Při čtení hodnoty do bufferu pBuff se nejdříve otevře soubor definovaný při inicializaci objektu. Pokud je soubor bez chyby otevřen, přečte se záznam určený indexem Ix do bufferu pBuff. (Pozn. Lineární adresa laBase se používá pouze při práci s pamětí.)

Pokud při čtení hodnoty nebo otevírání souboru dojde k chybě, vrací se FALSE a do proměnné m_wLastError se nastaví emd_DamagedMd. V opačném případě se vrací TRUE.

6.3.1.2.3. Put

```
function Put(laBase, Ix:LongInt; pBuff:Pointer):Boolean; virtual;
```

Funkce zapíše hodnotu v datovém bufferu pBuff do paměťového zařízené na pozici určenou lineární adresou laBase a indexem Ix.

Při zápisu hodnoty se nejdříve otevře soubor definovaný při inicializaci objektu. Pokud je soubor bez chyby otevřen, zapíše se hodnota v bufferu pBuff na pozici určenou indexem Ix. (Pozn. Lineární adresa laBase se používá pouze při práci s pamětí.)

Pokud při zápisu hodnoty nebo otevírání souboru dojde k chybě, vrací se FALSE a do proměnné m_wLastError se nastaví emd_DamagedMd. V opačném případě se vrací TRUE.

6.3.1.2.4. Create

```
function Create:Boolean; virtual;
```

Funkce dokončí inicializaci paměťového zařízení a zařízení uvede do provozu. V těle funkce se ověří zda existuje soubor, jehož jméno bylo objektu předané při inicializaci. Pokud tento soubor neexistuje, je vytvořen.

Pokud při otevírání nebo vytváření souboru dojde k chybě, vrací se FALSE a do proměnné m_wLastError se nastaví emd_DamagedMd. V opačném případě se vrací TRUE.

6.3.1.2.5. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci typu paměťového zařízení.

Funkce vždy vrací md_DosEmu.

7. Vrstva archivů

V této vrstvě jsou implementovány objekty odvozené od tArchiveVirt. Tyto objekty vytváří jednotné rozhraní pro práci s daty aplikace, dále záznamy, které mají být zachovávány nezávisle na běhu aplikace tzn. i v případě nového spuštění aplikace. Struktura těchto záznamů je pevná a při její změně se musí provést konverze dat. Pokud je potřeba ukládat záznamy s různou strukturou musí se vytvořit několik různých archivů.

Tyto záznamy mohou být ukládány postupně (v časové posloupnosti) nebo pomocí indexu do paměti uspořádané do kruhového bufferu nebo skladu.

Funkce rozhraní řeší synchronizační problémy pomocí zamykání jádra, což je dostačující při práci se záznamy v různých procesech. V případě zápisu záznamů v přerušovací rutině se synchronizace musí provést pomocí zákazu přerušení.

Pozn.: Deklarace záznamu archivu musí začínat položkou obsahující čas ve formátu DOS (LongInt). Z tohoto formátu vyplývá omezení max. roku a min. časové rozlišení 2 sec. Při dodržení této podmínky lze záznamy archivů zobrazovat v programu TheKing.

7.1. Knihovna ATypes

V knihovně jsou definovány chybové kódy, identifikační kódy archivů, inicializační struktura a struktura popisující stav archivu. V této struktuře jsou uloženy informace o označení banku, velikosti a počtu prvků, bázové adrese archivu a dva ukazatele na strukturu popisující stav archivu.

7.1.1. Konstanty

```
cVer = 'v1.05, 3.2.2003'
```

konstanta udává celkovou verzi a poslední změnu knihoven archivů, tzn. všech jednotek tvořící programový balík knihoven archivů.

```
cMinValidTime = $00210000;
```

konstanta určuje hodnotu min. platného času ve formátu DOS, který může být uložen v záznamu archivu. Pokud má záznam menší čas, je to detekováno jako chyba.

Tato hodnota vyjadřuje datum 1.1.1980 0:0:0.

```
cMaxValidTime = $C79FBF7D;
```

konstanta určuje hodnotu max. platného času ve formátu DOS, který může být uložen v záznamu archivu. Pokud má záznam větší čas, je to detekováno jako chyba.

Tato hodnota vyjadřuje datum 31.12.2079 23:59:59.

```
cInvalidIx = $FFFFFFFF;
```

hodnota neplatného indexu signalizujícího zaplnění skladu.

```
setFixArchiveFlags : tSetArchiveFlags =
  [af_InitArchive, af_SetFull, af_DisableInc, af_StockArchive,
                   af_UseMemoryDevice];
```

Množina vlastností archivu, které jsou nastaveny při inicializaci archivu a poté už nejsou měněny.

```
cAStateCrcSize = sizeof(tArchiveState)-sizeof(WORD){CRC};
```

Velikost struktury tArchiveState bez jejího CRC.

```
cArchiveResidum = $2347;
```

Inicializační hodnota při výpočtu CRC u zabezpečených struktur archivu. Především se jedná o struktury popisující stav archivu.

Chybové kódy

`ae_NoError = 0;`
: Při práci s archivem nedošlo k žádné chybě.

`ae_InvalidParams = 1;`
: Při inicializaci archivu byly předány neplatné parametry.

`ae_InvalidMd = 2;`
: Při inicializaci archivu bylo předáno neplatné paměťové zařízení případně nebylo zadáno. Při použití příznaku af_UseMemoryDevice musí být toto zařízení zadáno, tj. nesmí být NIL.

`ae_WriteError = 3;`
: Do paměti paměťového zařízení nelze zapsat novou hodnotu.

`ae_IncompParams = 4;`
: Při inicializaci archivu bylo zjištěno, že popisovač archivu je platný tj. bez chyby, ale předané parametry konstruktoru jsou nekompatibilní s parametry popisovače archivu.

`ae_BaseAddr = 5;`
: Při inicializace archivu bylo zjištěno, že popisovač archivu je platný tj. bez chyby a pouze předaná bázová adresa dat archivu se liší od uložené bázové adresy v popisovači archivu. V případě chyby ae_IncompParams se liší ještě alespoň jeden parametr archivu uložený v popisovači archivu.
Pozn. Při ladění programu na PC může dojít k posunu bázové adresy dat oproti uložené hodnotě v popisovači archivu. V případě překladu programu pro chráněný režim (protect mode) není vůbec zajištěna alokace stejného prostoru na HEAP. Proto je v případě ladění na PC potřeba tuto chybu ošetřovat jinak než pro řídicí jednotku KIT.

`ae_AStructCrc = 6;`
: Při inicializaci archivu bylo zjištěno, že popisovač archivu je poškozený.

Identifikační kódy objektů archivů

`at_CycleArchive = 0;`
: Identifikační kód objektu, který umí pracovat s pamětí uspořádanou do kruhového bufferu.

`at_ModCycleArchive = 1;`
: Identifikační kód objektu, který umí pracovat s paměti uspořádanou do kruhového bufferu. Tento objekt při čtení záznamů volá uživatelsky nastavitelnou funkci pro úpravu položek.

`at_StockArchive = 2;`
: Identifikační kód objektu, který umí pracovat s pamětí uspořádanou jako skladu.

`at_ModStockArchive = 3;`
: Identifikační kód objektu, který umí pracovat s pamětí uspořádanou jako sklad. Tento objekt při čtení záznamů volá uživatelsky nastavitelnou funkci pro úpravu položek.

7.1.2. Typy

```
tArchiveFlags = (
{ vlastnosti, ktere se nastavuji pri behu aplikace }
 af_ArchiveEmpty,            { archiv je prazdny - byl inicializovan
                           a jeste se do nej nezapsalo }
 af_ArchiveFull,                                      { priznak,
                           ze archiv je plny tj. pri zapisu doslo
                           alespon k jednomu preteceni - archiv
                           obsahuje max. pocet zaznamu }
{ vlastnosti, ktere jsou nastavovany pri vytvareni archivu }
 af_InitArchive,           { pri zjisteni poskozeneho archivu s
                           automaticky provede jeho inicializace
                           podle priznaku af_SetFull
 af_SetFull,                 { priznak, ktery urcuje zda v pripade
                           inicializace archivu funkci InitArchive
                           nastavit popisovace, aby se archiv
                           tvaril jako plny.
                           ++ defaultne vypnuto. Po inicializaci
                           popisovacu se archiv tvari jako prazdny.
                           }
 af_DisableInc,              { zakaze pouzivani inkrementalniho
                           cteni z archivu, tj. MASTER muze vzdy
                           cist pouze od zacatku archivu, tj. nikdy
                           nepouzije index na posledni cteny zaznam
                           }
 af_StockArchive,            { sklad (pole), do ktereho se po
                           zaplneni prestane zapisovat.
                           ++defaultne vypnuto. Pouziva se kruhovy
                           buffer. }
 af_UseMemoryDevice,       { pri pristupu k pameti se volaji funkce
                           zadaneho pametoveho zarizeni.
                           ++defaultne vypnuto. Pouziva se primy
                           pristup do pameti }
 af_NotCheckAStruct,       { pri inicialiazaci nejsou kontrolovany
                           popisovace archivu a jsou primo
                           nastaveny vlastnosti archivu dle hodnot
                           v ADesc. Při praci s timto priznakem
                           muze dojit ke ztrate dat! }
 af_Res9, af_Res10, af_Res11, af_Res12, af_Res13, af_Res14, af_Res15,
                           af_Res16
);
```

Výčtový typ vlastností archivu. Některé vlastnosti archivu jsou předdefinovány pro vyšší vrstvu.

```
tSetArchiveFlags = set of tArchiveFlags;
```

Množina vlastností archivu. Velikost je optimalizována na Word – atomické operace.

```
tArchiveState = record
 SetArchiveFlags   : tSetArchiveFlags;
                           { pole stavu a atributu archivu }
 IxWriteAt         : LongInt; { index pro zapis noveho zaznamu }
 IxOldestRecord    : LongInt; { index nejstarsiho zaznamu }
 EarlestRecordTime : LongInt; { cas nejmladsiho zaznamu v archivu }
 OldestRecordTime  : LongInt; { cas nejstarsiho zaznamu v archivu }
 wStateCrc         : Word;   { CRC pro zabezpeceni stavu archivu }
end;
```

Struktura obsahující indexy a stavové proměnné archivu.

Pozn. Tato struktura je u každého archivu dvakrát, jednou v pracovní verzi a podruhé v záložní verzi. Pokud by při práci s pracovní verzí např. došlo k výpadku napájení, je po novém startu aplikace zjištěno její poškození a její stav je obnoven ze záložní verze před nedokončenou operací.

```
pArchiveStruct = ^tArchiveStruct;
tArchiveStruct = record
{ priznak, ktery oznacuje aktualne platnou strukturu se stavem
                              archivu
  tj. FALSE - struktura WorkState je platna
      TRUE  - struktura WorkState je neplatna, protoze pri praci s
                              archivem doslo k chybe (posledni operace
                              nebyla dokoncena).
                              V tomto pripade by mela byt platna
                              zalozni struktura se stavem archivu }
 FBackupValid                 : Boolean;
{ pracovni strutura se stavem archivu }
 WorkState                    : tArchiveState;
{ zalozni struktura se stavem archivu }
 BackupState                  : tArchiveState;
 m_wArchiveNo                 : Word;       { cislo archivu }
 m_laBaseAddr                 : LongInt;          { linearni adresa
                              zacatku zalohovane pameti, v ktere je
                              archiv ulozen }
 m_wRecordSize                : Word; { velikost jednoho zaznamu v
                              bytech }
 m_dwMaxRecordCnt             : LongInt;    { max. pocet zaznamu v
                              archivu }
end;
```

Struktura popisuje parametry a stav archivu, dále bude uváděna jako popisovač archivu.

Pozn. Struktura musí být uložena v paměti nad datovým modulem a musí být uložena v zálohované paměti.

```
tfnModification = procedure(pRecord:Pointer; pExRecord:Pointer);
```

Deklarace uživatelské funkce umožňující úpravu záznamů při jejich čtení z archivu.

```
pArchiveDesc = ^tArchiveDesc;
tArchiveDesc = record
 wMask                        : Word; { rezervovano }
 wArchiveNo                   : Word; { cislo banku }
 laBaseAddr                   : LongInt;    { linearni adresa zacatku
                              zalohovane pameti archivu }
 wRecordSize                  : Word; { velikost jednoho zaznamu
                              archivu }
 dwMaxRecordCnt               : LongInt;    { max. pocet zaznamu v
                              archivu }
 SetAFlags                    : tSetArchiveFlags;
                              { priznaky archivu }
 pAStruct                     : pArchiveStruct;
                              { ukazatel na stav archivu - struktura
                              je ulozena v zalohovane pameti }
{ pouze u objektu, ktere modifikuji prvky archivu }
 wExRecordSize                : Word; { velikost upraveneho zaznamu }
 fnModification               : tfnModification;
                              { modifikacni funkce }
end;
```

Struktura, která se předává konstruktoru při inicializaci archivu.

7.1.3. Funkce

Dále bude uvedena deklarace několika pomocných funkcí, které jsou používány objekty na této nebo další vrstvě.

`function BeforeAndSameTime(X,Y:Longint):Boolean;`

Funkce porovnává dva časy ve formátu DOS a vrací TRUE, pokud čas X je menší nebo roven času Y. Pokud X je větší jak Y, vrací FALSE.
Funkce není reentrantní.
Porovnání času ve formátu DOS se provádí jako porovnání dvou DWord čísel.

`function BeforeTime(X,Y:Longint):Boolean;`

Funkce porovnává dva časy ve formátu DOS a vrací TRUE, pokud čas X je menší než čas Y. Pokud je X větší nebo roven Y vrací FALSE.
Funkce není reentrantní.
Porovnání času ve formátu DOS se provádí jako porovnání dvou DWord čísel.

`function` `ExLockKernel`*`:Boolean;`*

Protože během práce s Archivy je potřeba zajišťovat synchronizace, je v této knihovně obsažena funkce ExLockKernel. Tato funkce řeší hazardní stav při běžném používání funkce LockKernel. Funkce vrací TRUE, pokud jádro operačního systému už bylo uzamčeno. V případě, že jádro ještě nebylo uzamčeno, provede se jeho uzamknutí a funkce vrátí FALSE.
Tuto funkci nelze použít při synchronizaci práce s proměnnými, do kterých se zapisuje v přerušení. V případě těchto proměnných se musí provádět synchronizace pomocí příkazů assembleru PUSHF, CLI a POPF.

7.2. Knihovna AVirt

V knihovně je implementován objekt tArchiveVirt pracující s pamětí uspořádanou do kruhového bufferu. Do této paměti se ukládají záznamy konstantní délky a s pevně daným počtem prvků. Při ukládání nebo čtení záznamů se může do paměti přistupovat pomocí paměťového zařízení (nastavení příznaku af_UseMemoryDevice) nebo přímo.

Objekt implementuje rozhraní pro vyšší vrstvu, které umí čtení a zápis jednotlivých záznamu.

7.2.1. Objekt tArchiveVirt

Objekt je používán jako předek všech archivů a definuje rozhraní pro objekty vyšší vrstvy.

Pokud je nastaven příznak af_UseMemoryDevice používá se paměťové zařízení, v opačném případě se přistupuje přímo na paměť. Metody objektu provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

Pozn.: Deklarace záznamu archivu musí začínat položkou obsahující čas ve formátu DOS (LongInt). Z tohoto formátu vyplývá omezení max. roku a min. časové rozlišení 2 sec. Při dodržení této podmínky lze záznamy archivů zobrazovat v programu TheKing.

7.2.1.1. Položky

`m_wLastError : Word;`

Položka obsahuje kód poslední chyby při práci s Archivy.

`m_pArchiveStruct : pArchiveStruct;`

Ukazatel na popisovač archivu uložený v zálohované paměti.

`m_pMemoryDevice : pMemoryDeviceVirt;`

Ukazatel na paměťové zařízení.

`m_pTempRecord : Pointer;`

Pomocná paměť pro čtení záznamu z paměťového zařízení.

`m_theCrc : tCrc16;`

Statický objekt pro výpočet CRC zabezpečených struktur.

7.2.1.2. Metody

7.2.1.2.1. Konstruktor Init

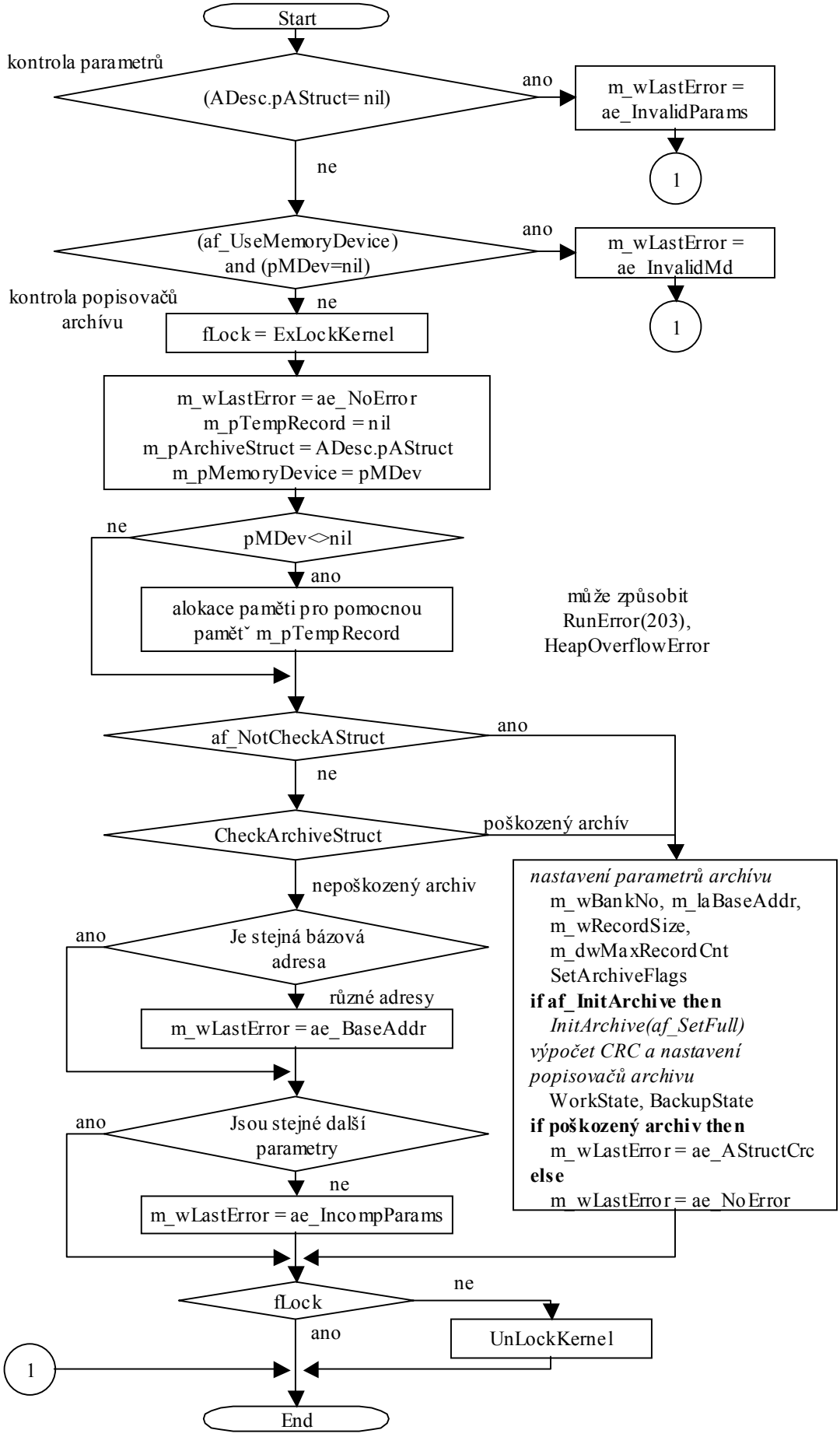
`constructor Init(pMDev:pMemoryDeviceVirt; var ADesc);`

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

Při spuštění konstruktoru se provede kód, který je zachycen ve vývojovém diagramu, viz obrázek 1: Výkonný kód konstruktoru.

Přestože funkce CheckArchiveStruct bude podrobně popsána v další části, je důležité pro pochopení činnosti konstruktoru vědět, že kontroluje především popisovače archivů. Pokud jsou popisovače nepoškozeny, zkontrolují se parametry archivu předané ve struktuře ADesc s parametry uloženými v zálohované paměti. Snadno se totiž může stát, že se tyto parametry mohou lišit buď v důsledku programové změny, která má vliv na strukturu archivu nebo poškození popisovačů. Pokud se liší bázová adresa uložení dat archivu, vrací se ae_BaseAddr v položce m_wLastError. Jestliže se liší více parametrů vrací se ae_IncompParams. Jsou-li ale poškozeny popisovače, vrací se ae_AStructCrc. Hodnota ae_InvalidMd se vrací, pokud je nenastavený pMDev.

Pokud jsou popisovače poškozeny, tzn. CheckArchiveStruct vrací FALSE, nastaví se popisovače archivu na předané hodnoty ve struktuře ADesc a je-li nastaven příznak af_InitArchive nastaví se stav archivu dle příznaku af_SetFull. Pokud příznak af_InitArchive není nastaven, musí se po ukončení konstruktoru provést inicializace dat archivu pomocí funkce InitArchive.


Start
kontrola parametrů
(ADesc.pAStruct= nil)
ano
m_wLastError = ae_InvalidParams
1
ne
(af_UseMemoryDevice) and (pMDev=nil)
ano
m_wLastError = ae_InvalidMd
1
kontrola popisovačů archívu
ne
fLock = ExLockKernel
m_wLastError = ae_NoError
m_pTempRecord = nil
m_pArchiveStruct = ADesc.pAStruct
m_pMemoryDevice = pMDev
ne
pMDev<>nil
ano
alokace paměti pro pomocnou pamět˘ m_pTempRecord
může způsobit RunError(203), HeapOverflowError
af_NotCheckAStruct
ano
ne
CheckArchiveStruct
poškozený archív
nepoškozený archiv
nastavení parametrů archívu
m_wBankNo, m_laBaseAddr,
m_wRecordSize,
m_dwMaxRecordCnt
SetArchiveFlags
if af_InitArchive then
InitArchive(af_SetFull)
výpočet CRC a nastavení popisovačů archivu
WorkState, BackupState
if poškozený archiv then
m_wLastError = ae_AStructCrc
else
m_wLastError = ae_NoError
ano
Je stejná bázová adresa
různé adresy
m_wLastError = ae_BaseAddr
ano
Jsou stejné další parametry
ne
m_wLastError = ae_IncompParams
fLock
ne
UnLockKernel
ano
1
End

obrázek 1: Výkonný kód konstruktoru

7.2.1.2.2. Destruktor Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu a uvolnění pomocné paměti pro práci s paměťovým zařízením. Na tuto paměť ukazuje položka m_pTempRecord. Destruktor objektu předchozí vrstvy se nevolá.

Funkce provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

7.2.1.2.3. IncIx

```
function IncIx(var Ix:LongInt):boolean; virtual;
```

Funkce nastaví index na další záznam podle nastavených parametrů objektu archivu, tzn. kruhový buffer v. sklad a dle max. počtu záznamů.

Protože objekt implementuje kruhový buffer, vrací funkce TRUE, pokud došlo k přetečení. V opačném případě vrací FALSE.

Funkce nemá ošetřenu synchronizaci, protože je zamýšlena jako interní funkce objektu.

7.2.1.2.4. DecIx

```
function DecIx(var Ix:LongInt):boolean; virtual;
```

Funkce nastaví index na předchozí záznam podle nastavených parametrů objektu archivu, tzn. kruhový buffer v. sklad a dle max. počtu záznamů.

Protože objekt implementuje kruhový buffer, vrací funkce TRUE, pokud došlo k podtečení. V opačném případě vrací FALSE.

Funkce nemá ošetřenu synchronizaci, protože je zamýšlena jako interní funkce objektu.

7.2.1.2.5. GetTimeBuff

```
function GetTimeBuff(pBuff:Pointer):LongInt; virtual;
```

Funkce vrací čas záznamu v datovém bufferu pBuff.

Pokud data archivu mají být vizualizovány pomocí programu TheKing, musí být čas záznamu umístěn u první položky a musí být ve formátu DOS (LongInt).

7.2.1.2.6. GetTimeIx

```
function  GetTimeIx(Ix:LongInt):LongInt; virtual;
```

Funkce vrací čas záznamu na místě indexu Ix. Pokud je Ix větší než max. počet prvků v archivu, vrací se cMinValidTime.

Pokud data archivu mají být vizualizovány pomocí programu TheKing, musí být čas záznamu umístěn u první položky a musí být ve formátu DOS (LongInt).

7.2.1.2.7. SetTimeIx

```
function  SetTimeIx(Ix:LongInt, Time:LongInt):Boolean; virtual;
```

Funkce nastaví čas záznamu na místě indexu Ix. Pokud je Ix větší než max. počet prvků v archivu, vrací se FALSE. V opačném případě se vrací TRUE.

7.2.1.2.8. GetLastError

```
function GetLastError:Word;
```

Funkce vrací kód poslední chyby.

7.2.1.2.9. SetLastError

```
procedure SetLastError(wCode:Word);
```

Funkce nastaví položku m_wLastError na hodnotu wCode.

7.2.1.2.10. InitArchive

```
procedure InitArchive(fFull:Boolean); virtual;
```

Funkce nastaví popisovače archivu do požadovaného stavu podle argumentu fFull:

- TRUE - archiv se tváří jako plný, čas nejmladšího záznamu EarlestRecordTime je nastaven na hodnotu cMaxValidTime a čas nejstaršího záznamu OldestRecordTime je nastaven na hodnotu cMinValidTime. Příznaky objektu jsou smazány a poté nastaveny na af_ArchiveFull. Vlastní záznamy zůstanou zachovány.
- FALSE - archiv se tváří jako prázdný, čas nejmladšího záznamu EarlestRecordTime a nejstaršího záznamu OldestRecordTime se nastaví na hodnotu cMinValidTime. Příznaky objektu jsou smazány a poté nastaveny na af_ArchiveEmpty. Vlastní záznamy zůstanou zachovány.

Po nastavení pracovní verze popisovače archivu se nastaví záložní verze.

7.2.1.2.11. CheckArchiveStruct

```
function  CheckArchiveStruct:Boolean; virtual;
```

Funkce zkontroluje obě verze popisovačů archivu a vrátí jejich stav. Pokud se zjistí, že některá verze popisovače archivu je neplatná, obnoví se její stav z druhé verze. Jsou-li obě verze popisovačů neplatné, vrací se FALSE. V opačném případě se vrací TRUE.

7.2.1.2.12. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací at_CycleArchive.

7.2.1.2.13. GetLen

```
function  GetLen : Longint; virtual;
```

Funkce vrací velikost použité paměti (v BYTE) vypočtené ze záznamů vracených funkcí At a počtu uložených záznamů. (Pozn.: Funkce může volat uživatelsky nastavitelnou funkcí pro úpravu záznamů.) Protože tento objekt úpravy záznamů pomocí uživatelské funkce nedovoluje, vrací se velikost použité paměti archivu.

Max. velikost použité paměti je vypočtena z GetRecordSize a z položky m_dwMaxRecordCnt nastavené při inicializaci archivu.

7.2.1.2.14. GetRecordSize

```
function  GetRecordSize:Word; virtual;
```

Funkce vrací velikost záznamu vraceného funkcí At. Protože tento objekt úpravu záznamů nedovoluje, vrací se přímo velikost záznamu v archivu.

Velikost záznamu – tj. m_wRecordSize nebo m_wExRecordSize se nastavuje při inicializaci archivu.

7.2.1.2.15. GetMaxRecordCnt

```
function  GetMaxRecordCnt:LongInt; virtual;
```

Funkce vrací max. počet záznamů v archivu.
Max. počet záznamů se nastavuje při inicializaci archivu.

7.2.1.2.16. GetArchiveNo

```
function  GetArchiveNo:Word;
```

Funkce vrací číslo označující archiv.
Zpravidla odpovídá indexu v poli objektu archivů.

7.2.1.2.17. GetOldestTime

```
function  GetOldestTime:LongInt;
```

Funkce vrací čas nejstaršího záznamu v archivu.

7.2.1.2.18. GetIxOldestTime

```
function  GetIxOldestTime:LongInt;
```

Funkce vrací index nejstaršího záznamu.

7.2.1.2.19. GetEarlestTime

```
function  GetEarlestTime:LongInt;
```

Funkce vrací čas nejmladšího záznamu v archivu.

7.2.1.2.20. GetIxForNewWrite

```
function  GetIxForNewWrite:LongInt;
```

Funkce vrací index, na který se bude zapisovat nový záznam.

7.2.1.2.21. IsSetFlag

```
function  IsSetFlag(fFlag:tArchiveFlags):Boolean;
```

Funkce vrací TRUE, pokud je zadaný příznak fFlag v archivu nastaven. V opačném případě vrací FALSE.

7.2.1.2.22. At

```
function  At(Ix:Longint;pRecord:Pointer):Boolean; virtual;
```

Funkce přečte hodnotu záznamu do datového bufferu pRecord z archivu podle pozice určené indexem Ix. Funkce vrací TRUE, pokud byla hodnota záznamu přečtena. V opačném případě vrací FALSE, např. index záznamu je větší než max. počet záznamů.

7.2.1.2.23. AtPut

```
function  AtPut(Ix:Longint;pRecord:Pointer):Boolean; virtual;
```

Funkce zapíše hodnotu záznamu v bufferu pRecord do archivu na pozici určenou indexem Ix. Funkce vrací TRUE, pokud byla hodnota záznamu zapsána. V opačném případě vrací FALSE, např. index záznamu je větší než max. počet záznamů.
Při nastavování hodnoty se nastaví čas nejmladšího a nejstaršího záznamu dle stavu archivu.
Funkce je zamýšlena pro volání, pokud se archiv používá jako pole.

7.2.1.2.24. Write

```
function Write(pRecord:Pointer):Boolean; virtual;
```

Funkce zapíše hodnotu záznamu v bufferu pRecord za naposledy zapsaný záznam. Při zápisu nového záznamu může dojít k přepisu starého záznamu, protože objekt používá kruhový buffer.

Při zápisu záznamu se nastaví čas nejmladšího a nejstaršího záznamu dle stavu archivu, ale neprovedou se žádné kontroly času. Do archivu lze proto zapsat záznam s neplatným časem, případně časem starším než je čas nejstaršího záznamu.

7.2.1.2.25. CheckWrite

```
function  CheckWrite(pRecord:Pointer):Boolean; virtual;
```

Funkce nejdříve zkontroluje čas nového záznamu, pokud je čas mladší nebo stejný volá se funkce Write. Při návratu se pak vrací její návratová hodnota. V opačném případě se funkce ukončí a vrací se FALSE.

7.3. Knihovna AMCycle

V knihovně je definován objekt archivu odvozený od objektu tArchiveVirt. Tento objekt na rozdíl od svého předka při čtení záznamu volá uživatelsky nastavitelnou funkci pro úpravu záznamů.

7.3.1. Objekt tModCycleArchive

Objekt implementuje archiv s pamětí uspořádanou do kruhového bufferu jako jeho předek, jenomže při čtení položek volá uživatelsky nastavitelnou funkci pro úpravu položek.

Pokud je nastaven příznak af_UseMemoryDevice používá se paměťové zařízení, v opačném případě se přistupuje přímo na paměť. Metody objektu provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

Pozn.: Deklarace záznamu archivu musí začínat položkou obsahující čas ve formátu DOS (LongInt). Z tohoto formátu vyplývá omezení max. roku a min. časové rozlišení 2 sec. Při dodržení této podmínky lze záznamy archivů zobrazovat v programu TheKing.

7.3.1.1. Položky

```
m_pOriginRecord             : Pointer;
```

Ukazatel na pomocnou paměť, do které se přečte záznam z archivu před jeho úpravou funkcí m_fnModification.

```
m_wExRecordSize             : Word;
```

Položka obsahuje velikost upraveného záznamu, který se vrací při čtení záznamu pomocí funkce At.

```
m_fnModification            : tfnModification;
```

Položka obsahuje uživatelsky nastavitelnou funkci, která provádí úpravu záznamů. Funkce se volá v těle At a provádí úpravu záznamu určeného proměnnou pRecord upravený záznam zapíše do na místo určené proměnnou pExRecord.

7.3.1.2. Metody

7.3.1.2.1. Konstruktor Init

```
constructor Init(pMDev:pMemoryDeviceVirt; var ADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

Po spuštění konstruktoru se zavolá konstruktor předka, jehož výkonný kód je zachycen ve vývojovém diagramu, viz obrázek 1: Výkonný kód konstruktoru. Po jeho vykonání se do položky m_fnModification nastaví uživatelsky nastavitelná funkce a provede se alokace pomocné paměti pro úpravu záznamů. Ukazatel na tuto paměť je uložen do položky m_pOriginRecord. Při této alokaci může vzniknout chyba RunError(203), HeapOverflowError.

Pokud funkce fnModification není nastavena nebo velikost upravených položek je 0, nastaví se položka m_wLastError na hodnotu ae_InvalidParams.

7.3.1.2.2. Destruktor Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu a uvolnění pomocné paměti pro úpravu záznamů. Na tuto paměť ukazuje m_pOriginRecord. Destruktor objektu předchozí vrstvy se nevolá.

7.3.1.2.3. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací at_ModCycleArchive.

7.3.1.2.4. GetRecordSize

```
function  GetRecordSize:Word; virtual;
```

Funkce vrací velikost záznamu vraceného funkcí At. Protože tento objekt dovoluje úpravu záznamů, vrací se velikost m_wExRecordSize nastavená při inicializaci archivu.

7.3.1.2.5. At

```
function  At(Ix:Longint;pRecord:Pointer):Boolean; virtual;
```

Funkce zavolá metodu předka At, která přečte hodnotu záznamu z archivu určenou indexem Ix do pomocné paměti určené m_pOriginRecord. Pokud předek vrátil TRUE, volá se uživatelsky nastavitelná funkce pro úpravu záznamů. Tato funkce nastaví hodnotu upraveného záznamu do paměti určené pRecord. Její návratová hodnota se poté vrací jako výsledný kód funkce. V případě, kdy předek vrátil FALSE, je funkce ukončena s návratovou hodnotou FALSE.

7.4. Knihovna AStock

V knihovně je definován objekt archivu odvozený od objektu tArchiveVirt. Tento objekt na rozdíl od svého předka implementuje sklad. Do prostoru skladu se ukládají jednotlivé záznamy tak dlouho, dokud není zaplněn. Po jeho zaplnění se záznamy přestanou ukládat a funkce pro zápis vrací chybu.

7.4.1. Objekt tStockArchive

Objekt implementuje archiv s pamětí uspořádanou jako sklad. Do této paměti se jednotlivé záznamy ukládají tak dlouho, dokud paměť není zaplněna. Po jejím zaplnění se záznamy přestanou ukládat a funkce pro zápis vrací chybu.

Pokud je nastaven příznak af_UseMemoryDevice používá se paměťové zařízení, v opačném případě se přistupuje přímo na paměť. Metody objektu provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

Pozn.: Deklarace záznamu archivu musí začínat položkou obsahující čas ve formátu DOS (LongInt). Z tohoto formátu vyplývá omezení max. roku a min. časové rozlišení 2 sec. Při dodržení této podmínky lze záznamy archivů zobrazovat v programu TheKing.

7.4.1.1. Metody

7.4.1.1.1. IncIx

```
function IncIx(var Ix:LongInt):boolean; virtual;
```

Funkce nastaví index na další záznam podle nastavených parametrů objektu archivu, tzn. kruhový buffer v. sklad a dle max. počtu záznamů.
Protože objekt implementuje sklad, vrací funkce TRUE, pokud je archiv zaplněn. V opačném případě vrací FALSE.
Funkce nemá ošetřenu synchronizaci, protože je zamýšlena jako interní funkce objektu.

7.4.1.1.2. DecIx

```
function DecIx(var Ix:LongInt):boolean; virtual;
```

Funkce nastaví index na předchozí záznam podle nastavených parametrů objektu archivu, tzn. kruhový buffer v. sklad a dle max. počtu záznamů.
Protože objekt implementuje sklad, vrací funkce TRUE, pokud je archiv na začátku. V opačném případě vrací FALSE.
Funkce nemá ošetřenu synchronizaci, protože je zamýšlena jako interní funkce objektu.

7.4.1.1.3. Konstruktor Init

```
constructor Init(pMDev:pMemoryDeviceVirt; var ADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.
Po spuštění konstruktoru se zavolá konstruktor předka, jehož výkonný kód je zachycen ve vývojovém diagramu, viz obrázek 1: Výkonný kód konstruktoru.

7.4.1.1.4. InitArchive

```
procedure InitArchive(fFull:Boolean); virtual;
```

Funkce volá metodu předka a po jejím dokončení přestaví položku v položce IxWriteAt na neplatný index, pokud je archiv zaplněn (fFull = TRUE). Pokud mají být popisovače nastaveny na popis prázdného archivu (fFull = FALSE), ponechá se nastavení provedené předkem.

7.4.1.1.5. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací at_StoreArchive.

7.4.1.1.6. Write

```
function Write(pItem:Pointer):Boolean; virtual;
```

Před zavoláním metody předka Write se zkontroluje zda archiv není zaplněn, tj. položka IxWriteAt není nastavena na neplatný index. Pokud je archiv zaplněn vrací se FALSE. V opačném případě se volá metoda předka. Její návratová hodnota se pak vrací jako výsledný kód funkce.

7.5. Knihovna AMStock

V knihovně je definován objekt archivu odvozený od objektu tStockArchive. Tento objekt na rozdíl od svého předka při čtení záznamů volá uživatelsky nastavitelnou funkci pro úpravu záznamů.

7.5.1. Objekt tModCycleArchive

Objekt implementuje archiv s pamětí uspořádanou jako sklad. Na rozdíl od svého předka při čtení záznamů volá uživatelsky nastavitelnou funkci pro úpravu záznamů.

Pokud je nastaven příznak af_UseMemoryDevice používá se paměťové zařízení, v opačném případě se přistupuje přímo na paměť. Metody objektu provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

Pozn.: Deklarace záznamu archivu musí začínat položkou obsahující čas ve formátu DOS (LongInt). Z tohoto formátu vyplývá omezení max. roku a min. časové rozlišení 2 sec. Při dodržení této podmínky lze záznamy archivů zobrazovat v programu TheKing.

7.5.1.1. Položky

```
m_pOriginRecord             : Pointer;
```

Ukazatel na pomocnou paměť, do které se přečte záznam z archivu před jeho úpravou funkcí m_fnModification.

```
m_wExRecordSize             : Word;
```

Položka obsahuje velikost upraveného záznamu, který se vrací při čtení záznamu pomocí funkce At.

```
m_fnModification            : tfnModification;
```

Položka obsahuje uživatelsky nastavitelnou funkci, která provádí úpravu záznamů. Funkce se volá v těle At a provádí úpravu záznamu určeného proměnnou pRecord upravený záznam zapíše do na místo určené proměnnou pExRecord.

7.5.1.2. Metody

7.5.1.2.1. Konstruktor Init

```
constructor Init(pMDev:pMemoryDeviceVirt; var ADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

Po spuštění konstruktoru se zavolá konstruktor předka, který dále volá konstruktor svého předka, jehož výkonný kód je zachycen ve vývojovém diagramu, viz obrázek 1: Výkonný kód konstruktoru. Po jeho vykonání se nastaví do položky m_fnModification uživatelsky nastavitelná funkce a provede se alokace pomocné paměti pro úpravu záznamů. Ukazatel na tuto paměť je uložen do položky m_pOriginRecord. Při této alokaci může vzniknout chyba RunError(203), HeapOverflowError.

Pokud funkce fnModification není nastavena nebo velikost upravených položek je 0, nastaví se položka m_wLastError na hodnotu ae_InvalidParams.

7.5.1.2.2. Destruktor Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu a uvolnění pomocné paměti pro úpravu záznamů. Na tuto paměť ukazuje m_pOriginRecord. Destruktor objektu předchozí vrstvy se nevolá.

7.5.1.2.3. GetId

```
function  GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací at_ModStockArchive.

7.5.1.2.4. GetRecordSize

```
function  GetRecordSize:Word; virtual;
```

Funkce vrací velikost záznamu vraceného funkcí At. Protože tento objekt dovoluje úpravu záznamů, vrací se velikost m_wExRecordSize nastavená při inicializaci archivu.

7.5.1.2.5. At

```
function  At(Ix:Longint;pRecord:Pointer):Boolean; virtual;
```

Funkce zavolá metodu předka At, která přečte hodnotu záznamu z archivu určenou indexem Ix do pomocné paměti určené m_pOriginRecord. Pokud předek vrátil TRUE, volá se uživatelsky nastavitelná funkce pro úpravu záznamů. Tato funkce nastaví hodnotu upraveného záznamu do paměti určené pRecord. Její návratová hodnota se poté vrací jako výsledný kód funkce. V případě, kdy předek vrátil FALSE, je funkce ukončena s návratovou hodnotou FALSE.

8. Vrstva archivů s podporou komunikace a vizualizace

V této vrstvě jsou implementovány objekty odvozené od tChArchiveVirt. Tyto objekty vytváří jednotné rozhraní pro další vrstvu komunikující s programem TheKing. Mezi základní funkce tohoto rozhraní patří metody pro práci s popisovači zobrazení záznamů pomocí objektů tDscr a tExDscr.

Tato vrstva vždy pracuje nad objekty odvozenými od tArchiveVirt. Zatímco objekty spodní vrstvy (odvozené od tArchiveVirt) umí ukládat a číst jednotlivé záznamy podle zadaného indexu případně ukládat záznamy postupně za sebou, umí objekty této vrstvy číst více záznamů najednou ze zadaného časového intervalu.

8.1. Knihovna ChATypes

V knihovně jsou definovány chybové kódy, identifikační kódy archivů, inicializační struktura a struktury (dále popisovače tabulky nebo grafu) popisující zobrazení v programu TheKing.

8.1.1. Konstanty

`cMinSecDiff = 2;`

Hodnota se zadává v sekundách a určuje čas posledního přečteného záznamu z řídicího systému. Tento čas je vypočten jako (aktuální čas – cMinSecDiff). Tímto zpožděním je zajištěno, že v programu TheKing se nebude stejný záznam objevovat vícekrát. Tento problém nastává, pokud se v jednom časovém okamžiku do archivu ukládají dva záznamy se stejným časem a mezi jejich uložením se budou číst záznamy.

Identifikační kódy objektů archivů podporujících komunikaci a vizualizaci

`chat_ChArchiveVirt = 0;`

Identifikační kód archivu, který podporuje čtení záznamů pomocí zadaného časového intervalu. Dále tento objekt vytváří rozhraní pro objekty komunikující s programem TheKing.

`chat_DscrChArchive = 1;`

Identifikační kód archivu, který implementuje funkce pracující s popisovači tabulek. Proto při použití tohoto objektu jsou záznamy v programu TheKing zobrazeny tabulkou.

`chat_ExDscrChArchive = 2;`

Identifikační kód archivu, který implementuje funkce pracující s popisovači grafu. Proto při použití tohoto objektu mohu být záznamy v programu TheKing zobrazeny tabulkou nebo grafem.

```
cTypeMaskDscr : array [1..XtKing_1.DscrMsk_TypPolMax] of
                              TTypeMaskDscr = (
  (m_bTypeMask:DscrMsk_Byte;         m_wLen:1),
  (m_bTypeMask:DscrMsk_Word;         m_wLen:2),
  (m_bTypeMask:DscrMsk_SmallInt;     m_wLen:2),
  (m_bTypeMask:DscrMsk_Integer;      m_wLen:4),
  (m_bTypeMask:DscrMsk_LongInt;      m_wLen:4),
  (m_bTypeMask:DscrMsk_Dword;        m_wLen:4),
  (m_bTypeMask:DscrMsk_String;       m_wLen:0),
  (m_bTypeMask:DscrMsk_RealP;        m_wLen:6),
  (m_bTypeMask:DscrMsk_DosDaTi;      m_wLen:4),
  (m_bTypeMask:DscrMsk_ByteBits;     m_wLen:1),
  (m_bTypeMask:DscrMsk_undef;        m_wLen:0),
  (m_bTypeMask:DscrMsk_undef;        m_wLen:0),
  (m_bTypeMask:DscrMsk_undef;        m_wLen:0),
  (m_bTypeMask:DscrMsk_undef;        m_wLen:0),
  (m_bTypeMask:DscrMsk_undef;        m_wLen:0),
  (m_bTypeMask:DscrMsk_SWdiv10;      m_wLen:2),
  (m_bTypeMask:DscrMsk_SWdiv100;     m_wLen:2),
  (m_bTypeMask:DscrMsk_SWdiv1000;    m_wLen:2),
  (m_bTypeMask:DscrMsk_SWdiv10000;   m_wLen:2),
  (m_bTypeMask:DscrMsk_SIdiv10;      m_wLen:2),
  (m_bTypeMask:DscrMsk_SIdiv100;     m_wLen:2),
  (m_bTypeMask:DscrMsk_SIdiv1000;    m_wLen:2),
  (m_bTypeMask:DscrMsk_SIdiv10000;   m_wLen:2),
  (m_bTypeMask:DscrMsk_SLdiv10;      m_wLen:4),
  (m_bTypeMask:DscrMsk_SLdiv100;     m_wLen:4),
  (m_bTypeMask:DscrMsk_SLdiv1000;    m_wLen:4),
  (m_bTypeMask:DscrMsk_SLdiv10000;   m_wLen:4));
```

Tabulka typů a jejich velikostí. Používá se při sestavení popisovače tabulky pro program TheKing. Pořadí a velikost typů musí odpovídat deklaraci v jednotce XtKing_1.pas

Popisované konstanty jsou definovány v jednotce Xtking_1.pas. V této jednotce jsou definovány konstanty a typy, které se sdílí mezi aplikací v řídicím systému a programem TheKing.

Popis konstant je uveden pro úplnost deklarace konstant.

Typ	Velikost (Byte)	Rozsah
DscrMsk_Undef	0	Typ se nesmí použít
DscrMsk_Byte	1	0 - 255
DscrMsk_Word	2	0 - 65535
DscrMsk_SmallInt	2	-32768 – 32767
DscrMsk_Integer	4	- 2147483648 - 2147483647
DscrMsk_LongInt	4	- 2147483648 - 2147483647
DscrMsk_Dword	4	0 – 4294967296
DscrMsk_String	dáno vstupem, řetězec typu PASCAL, 0BYTE definuje délku řetězce	
DscrMsk_Realp	6	$\pm 2.9e^{-39}$ - $1.7e^{38}$
DscrMsk_DosDaTi	4	1.1.1980 0:0:0 – 31.12.2079 23:59:59
DscrMsk_ByteBits	1	0-255
DscrMsk_SWdiv10	2	rozlišení 1/10, -0 – 6553,5
DscrMsk_SWdiv100	2	rozlišení 1/100, -0 – 655,35
DscrMsk_SWdiv1000	2	rozlišení 1/1000, -0 – 65,535
DscrMsk_SWdiv10000	2	rozlišení 1/10000, -0 – 6,5535
DscrMsk_SIdiv10	2	rozlišení 1/10, -3276,8 – 3276,7
DscrMsk_SIdiv100	2	rozlišení 1/100, -327,68 – 327,67
DscrMsk_SIdiv1000	2	rozlišení 1/1000, -32,768 – 32,767
DscrMsk_SIdiv10000	2	rozlišení 1/10000, -3,2768 – 3,2767
DscrMsk_SLdiv10	4	rozlišení 1/10, - 214748364,8 – 214748364,7
DscrMsk_SLdiv100	4	rozlišení 1/100, - 21474836,48 – 21474836,47
DscrMsk_SLdiv1000	4	rozlišení 1/1000, - 2147483,648 – 2147483,647
DscrMsk_SLdiv10000	4	rozlišení 1/10000, - 214748,3648 – 214748,3647

Tabulka popisuje velikosti (byte) a rozsahy typů definovaný v souboru XtKing_1.pas.

Popisované konstanty jsou definovány v jednotce Xtking_2.pas. V této jednotce jsou definovány konstanty a typy, které se sdílí mezi aplikací v řídicím systému a programem TheKing.

Popis konstant je uveden pro úplnost deklarace konstant.

Konstanty popisovačů zobrazení grafu

V dnešní době program TheKing podporuje následující konstanty:

`exfat_TExFieldAttrChartSeries = 1;`

Konstanta definuje typ popisovače, který bude popisovat na jakou souřadnou osu se vybraná položka záznamu bude vykreslovat.

`exfat_TExFieldAttrChartPhysicalUnits = 2;`

Konstanta definuje typ popisovače, který bude popisovat fyzikální jednotky jednotlivých položek záznamu.

`exfat_TExFieldAttrChartBitNames = 3;`

Konstanta definuje typ popisovače, který bude popisovat jednotlivé bity vybrané položky záznamu. Tato položka musí být typu bitové pole.

Konstanty popisující propojování bodů grafu

V dnešní době program TheKing podporuje následující konstanty:

`teeseries_msk_SeriesType_None = $00;`

Při zadání této konstanty se nebude vybraná položka záznamu vykreslovat v grafu, protože nemá definovaný způsob vykreslování bodů a jejich propojování. Tuto konstantu s výhodou použijete, pokud tuto položku nebudete chtít vykreslovat v grafu.

`teeseries_msk_SeriesType_Line = $01;`

Jednotlivé body vybrané položky záznamu budou spojeny čarou.

Konstanta popisující vykreslování položek

`teeseries_msk_SeriesAxisX_DaTi = $20;`

U vybrané položky záznamu se určí, že čas na ose X má být zobrazován ve formátu datum a čas. Pokud tato konstanta není zadána, zobrazuje se reálné číslo.

`teeseries_msk_SeriesAxisY_Default = $00;`

Tato konstanta určí u vybrané položky záznamu, že její hodnoty mají být vztaženy vůči levé ose Y.

`teeseries_msk_SeriesAxisY_Left = $40;`

Tato konstanta určí u vybrané položky záznamu, že její hodnoty mají být vztaženy vůči levé ose Y.

`teeseries_msk_SeriesAxisY_Right = $80;`

Tato konstanta určí u vybrané položky záznamu, že její hodnoty mají být vztaženy vůči pravé ose Y.

8.1.2. Typy

```
PTypeMaskDscr = ^TTypeMaskDscr;
TTypeMaskDscr = record
 m_bTypeMask                : XtKing_1.TDscrMsk_Type;
 m_wLen     : Word;
 end;
```

Struktura definuje typ jedné položky záznamu. Používá se pro sestavení kódovací tabulky typů.

```
tChArchiveFlags = (
 chaf_LostDataForRead,  { doslo ke ztrate dat urcenych pro cteni }
 chaf_IncReading,       { z archivu lze cist pouze nove polozky }
 chaf_Res3, chaf_Res4, chaf_Res5, chaf_Res6, chaf_Res7, chaf_Res8,
 chaf_Res9, chaf_Res10, chaf_Res11, chaf_Res12, chaf_Res13,
 chaf_Res14, chaf_Res15, chaf_Res16);
tSetChArchiveFlags=set of tChArchiveFlags;
```

Výčtový typ a množina vlastností archivů podporujících komunikaci a vizualizaci.

```
pChArchiveDesc = ^tChArchiveDesc;
tChArchiveDesc = record
 wMask     : Word;          { rezervovano }
 SecDiff   : LongInt;       { pri prenosu se prenasi pouze polozky,
                            ktere jsou starsi jak (aktualni cas -
                            tato konstanta) }
 { DscrChArchive }
 pDscrObj  :pDscr;          { ukazatel na objekt popisovace}
 strName   :tArchName;      { jmeno archivu }
 { ExDescrArchive }
 wExDscrCnt : Word;         { max. pocet popisovacu grafu, ktere lze
                            u objektu zadat }
end;
```

Inicializační struktura archivů s podporou komunikace a vizualizace.

```
pExDscrData = ^tExDscrData;
tExDscrData = record
 case Byte of
 0 : (ChartSeries:XtKing_2.TExFieldAttrChartSeries;);
 1 : (ChartPhysicalUnits:XtKing_2.TExFieldAttrChartPhysicalUnits;);
 2 : (ChartBitNames:XtKing_2.TExFieldAttrChartBitNames;);
end;
```

Variantní typ struktury používané při inicializaci popisovačů zobrazení v grafu.

Popisované typy jsou definovány v jednotce Xtking_1.pas. V této jednotce jsou definovány konstanty a typy, které se sdílí mezi aplikací v řídicím systému a programem TheKing.

Popis typů je uveden pro úplnost deklarace jednotlivých typů.

```
tArchName = string[20];
```

Typ definuje řetězec o max. počtu znaků, kterým lze pojmenovat archiv. Tento řetězec se bude zobrazovat v programu TheKing.

```
TDscrMsk_Type = byte;
```

Typ pro popis typů položek záznamu archivu.

```
TArchiveRecHeader = string[254];
```

Typ pro popis názvů jednotlivých sloupců tabulky zobrazující záznamy archivu. Název jednoho sloupce se doporučuje max. 20 znaků vzhledem ke struktuře pro nastavení popisu jednotlivých bitů.

```
TArchiveFmtHeader = string[254];
```

Typ popisující zobrazování dat v tabulce. Řetězec popisující zobrazování položky záznamu se sestaví stejně jako formátovací řetězec u příkazu FormatStr v jednotce Drivers.

```
TArchiveRecDscr = record
 RecLen     : word;          { udava delku nasledujici Masky/Zaznamu
                             v archivu}
 ResDscrMask                 : array[0..MaxArchiveRecSize-1] of byte;
                             { maska popisujici typ dat DscrMsk_xxx }
end;
```

Struktura popisující typy položek záznamů zobrazovaných v tabulce. Proměnná tohoto typu je sestavena pomocí objektu tDscr a předává se přímo komunikačním objektům.

```
TArchDataHeader = record     { slouzi k pretypovani bufferu promenne
                             delky }
 NextHeaderFl                : Boolean;    { priznak ne-posledniho
                             prenaseneho header }
 ExHeaderCount               : byte; { pocet naslednych
                             TArchDataExHeader }
 ArchiveID  : byte;          { identifikace typu archivu archid_xxx }
 ArchBankNum                 : byte; { cislo banky archivu }
 ArchName   : TArchName;     { nazev archivu, pripadne jmeno souboru
                             }
 ArchAttr   : byte;          { attributy banky archivu archattr_xxx }
 ArchState  : TRecArchiveState;
                             { stav archivu v bance }
 ArchRecHeader               : TArchiveRecHeader;
                             { popisky/zahlavi polozek zaznamu }
 ArchFmtHeader               : TArchiveFmtHeader;
                             { format % strings polozek zaznamu like
                             CSV }
 ArchRecDscr                 : TArchiveRecDscr;
                             { delka ovlivnena ArchivRecDscr.RecLen
                             !!!}
end;
```

Struktura popisující zobrazování záznamů do tabulky. Proměnná tohoto typu je sestavena pomocí potomků tChArchiveVirt a předává se přímo komunikačním objektům. Při sestavování této struktury se volají funkce objektu tDscr.

```
TRecArchiveState = record    { Zaznam statusu banky archivu }
 UpDateFlg  : Boolean;       { priznak noveho zaznamu od posledniho
                             vycteni }
 DataLost   : Boolean;       { Priznak ztraty dat v dusledku
                             preteceni bufferu }
 TimeLastClear               : Longint;    { (MS-DOS PackTime) cas
                             nejstarsiho zaznamu }
 TimeLastUpDate              : Longint;    { (MS-DOS PackTime) cas
                             nejmladsiho zaznamu }
 ArchiveLength               : Longint;    { delka zaplneneho bufferu
                             banky archivu }
end;
```

Struktura popisující stav archivu. Proměnná tohoto typu je sestavena pomocí potomků tChArchiveVirt a předává se přímo komunikačním objektům.

```
TArchExRecDscr = record      { Extended descriptor - dalsi attributy
                             polozek  }
 ExFieldAttrType             : byte; { identifikace typu ~
                             TExFieldAttrRec ~ exfat_xx }
 ExFieldSize                 : word; { delka Attr polozky ~
                             SizeOf(TExFieldAttrRec)   }
 ExFieldsCount               : word; { skutecny pocet polozek v
                             ExFieldsAttrRec       }
 ExFieldsAttrRec             : array[0..MaxExFieldsAttrRecSize-1] of
                             byte;{ pozor[index] po bytech }
 { ^toto pole je pouzivano jako array[..] of TExFieldAttrRec !!!}
end;
```

Struktura popisující zobrazování záznamů do grafu. Proměnná tohoto typu je sestavena pomocí objektu tExDscr a předává se přímo komunikačním objektům.

```
TArchDataExHeader = record   { slouzi k pretypovani bufferu promenne
                             delky }
 NextExHeaderFl              : Boolean;    { priznak ne-posledniho
                             prenaseneho ExHeader }
 ArchExRecDscr               : TArchExRecDscr;
                             { delka ovlivnena skutecnou delkou !!!}
end;
```

Hlavní struktura popisující zobrazování záznamů do grafu. Proměnná tohoto typu je sestavena pomocí potomků tChArchiveVirt a předává se přímo komunikačním objektům. Při sestavování této struktury se volají funkce objektu tExDscr.

Popisované typy jsou definovány v jednotce Xtking_2.pas. V této jednotce jsou definovány konstanty a typy, které se sdílí mezi aplikací v řídicím systému a programem TheKing.

Popis typů je uveden pro úplnost deklarace jednotlivých typů.

```
type
PExFieldAttrChartSeries = ^TExFieldAttrChartSeries;
TExFieldAttrChartSeries = record
 TeeSeriesAttr               :byte;
 ScanSemAttr                 :byte;
end;
```

Deklarace struktury popisovače grafu, která umožňuje zadávat popis vykreslování jednotlivých položek záznamu v grafu. Do položky TeeSeriesAttr se zadávají konstanty s prefixem teeseries_msk_. Položka SceSemAttr je rezervována.

Do objektu tExDscr může být vloženo více popisovačů tohoto typu.

```
TAttrChartAxisLabel = string[ 63];
```

Typ pro textový popis jednotlivých os grafu.

```
TAttrChartPhysicalUnitStrings = string[254];
```

Typ pro textový popis fyzikálních jednotek jednotlivých položek záznamu. Jednotlivé položky jsou od sebe odděleny znakem '|'.

```
PExFieldAttrChartPhysicalUnits = ^TExFieldAttrChartPhysicalUnits;
TExFieldAttrChartPhysicalUnits = record
 LeftAxisLabel              :TAttrChartAxisLabel; { popiska leve osy
                            Y }
 RightAxisLabel             :TAttrChartAxisLabel; { popiska prave
                            osy Y }
 PhysicalUnitStrings : TAttrChartPhysicalUnitStrings; { fyzikalni
                            jednotky pro polozky vynasene do grafu }
end;
```

Typ pro popis popisek os a fyzikálních jednotek jednotlivých položek záznamu.

Do objektu tExDscr může být vložena pouze jedna položka tohoto typu.

```
TAttrCharBitsFieldName = string[20];
```

Typ pro jméno položky ze seznamu položek vynášených do grafu. Toto jméno musí odpovídat označení sloupce uvedeném v řetězci Key při volání konstruktoru tDscr.

```
TAttrCharBitNameStrings = string[80];
```

Typ pro jména bitů pro vybranou položku záznamu. Jména jednotlivých bitů jsou od sebe oddělena znakem '|' a zadávají se od nejnižšího bitu B0 až po nejvyšší bit B7. Pokud typ položky není DscrMsk_ByteBits, je tento popisovač ignorován.

```
PExFieldAttrChartBitNames = ^TExFieldAttrChartBitNames;
TExFieldAttrChartBitNames = record
 FieldName  :TAttrCharBitsFieldName
 BitNameStrings             :TAttrCharBitNameStrings;
end;
```

Typ pro popis jednotlivých bitů jedné položky záznamu typu DscrMsk_ByteBits.

Do objektu tExDscr může být vloženo více popisovačů tohoto typu.

8.1.3. Objekt tDscr

Objekt se používá pro sestavení popisovače zobrazení tabulky a je volán potomky tChArchiveVirt při sestavování struktury pro objekty komunikující s programem TheKing.

8.1.3.1. Položky

```
m_Key       : XtKing_1.TArchiveRecHeader;
```

Položka obsahuje názvy jednotlivých sloupců v tabulce zobrazující data archivu. Popis se zadává v textovém formátu a jednotlivé sloupce jsou od sebe odděleny znakem |, např. 'cas|kod'.

```
m_Fmt       : XtKing_1.TArchiveFmtHeader;
```

Položka popisuje zobrazení dat v tabulce. Popis se zadává v textovém formátu pomocí stejné syntaxe jako ve formátovacím řetězci při použití příkazu FormatStr v jednotce Drivers, např. %3.2f|%4.1f|%4.4x. Sloupce se oddělují pomocí znaku |.

```
m_wTypeCnt  : Word;
```

Počet zadaných typů jednotlivých položek záznamů pomocí funkce AddType.

```
m_wMaxTypeCnt               : Word;
```

Počet položek v jednom záznamu, který se musí zadat. Tato hodnota se vypočte při inicializaci objektu z počtu znaků | v řetězci m_Key.

```
m_pTypeMaskArr              : Pointer;
```
Ukazatel na paměť popisující typy jednotlivých položek záznamů. Velikost této paměti je dána počtem položek v jednom záznamu m_wMaxTypeCnt.

```
m_wRecordSize               : Word;
```
Velikost (byte) všech zadaných typů jednotlivých položek v záznamu. Tato velikost se vypočítává při volání AddType.

```
m_wMaxRecordSize            : Word;
```
Velikost (byte) celého záznamu. Tato velikost se zadává při inicializaci objektu a používá se pro kontrolu správného zadání typů jednotlivých položek záznamu.

8.1.3.2. Metody

8.1.3.2.1. Init

```
constructor Init(const Key:XtKing_1.TArchiveRecHeader; const
                            Fmt:XtKing_1.TArchiveFmtHeader;
                            wRecordSize:Word);
```
Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

V těle se vypočte počet položek záznamu z proměnné Key, v které se spočítá počet znaků |. Tato proměnná se zapíše do položky m_Key. V této položce se zadávají jména sloupců odpovídající jednotlivým položkám záznamu. Počet znaků jména sloupce by neměl být vyšší než 20 znaků a jména sloupců se od sebe oddělují pomocí znaku |. Proměnná Fmt se zapíše do položky m_Fmt a její syntaxe je stejná jako u formátovacího řetězce u příkazu FormatStr v jednotce Drivers. Formátovací řetězce pro jednotlivé sloupce se oddělují pomocí znaku |. Hodnota wRecordSize musí odpovídat velikosti záznamu v archivu a je uložena do položky m_wMaxRecordSize.

Dále se nastaví položky m_wItemCnt, m_wRecordSize a m_wMaxItemSize na 0.

Nakonec se provede alokace paměti pro buffer m_pTypeMaskArr, do které se budou ukládat typy a velikosti jednotlivých položek záznamu.

8.1.3.2.2. Done

```
destructor Done; virtual;
```
Destruktor slouží ke zrušení objektu a uvolnění paměti pro popis typů jednotlivých položek záznamů. Na tuto paměť ukazuje m_pTypeMaskArr.

8.1.3.2.3. AddType

```
function AddType(TypeMask:XtKing_1.TDscrMsk_Type; wLen:Word):Boolean;
                            virtual;
```
Funkce určí další typ položky záznamu. Typy jednotlivých položek se musí přidávat ve stejném pořadí jako je deklarace záznamu, jinak zobrazení v tabulce nebude odpovídat hodnotám v záznamu. Parametr wLen se musí zadat pouze u typů položek, které to vyžadují, viz tabulka popisující velikosti a typy.

8.1.3.2.4. GetDscr

```
procedure GetDscr(var RecDscr:XtKing_1.TArchiveRecDscr); virtual;
```

Procedura sestaví strukturu TArchiveRecDscr používanou objekty komunikující s programem TheKing . Procedura je volána potomky objektu tChArchiveVirt.

8.1.3.2.5. GetDscrLen

```
function GetDscrMaskLen:Word; virtual;
```

Funkce vrací velikost paměti (byte) u položky ResDscrMask ve struktuře TArchiveRecDscr.

8.1.4. Objekt tExDscr

Objekt se používá pro sestavení jednoho typu popisovače zobrazení grafu a je volán potomky tChArchiveVirt při sestavování struktury pro objekty komunikující s programem TheKing.

Pokud to typ popisovače dovoluje, lze zadat více položek. Zadání těchto položek se provádí pomocí funkce AddExDscrItem.

Těchto objektů můžou mít potomci nebo instance objektu tExDscrChArchive několik.

8.1.4.1. Položky

```
m_bExDscrType          : Byte;
```

Položka obsahuje typ popisovače grafu, který přímo ovlivňuje hodnoty v položkách m_ExDscrItemSize a m_pItemArr.

```
m_wExDscrItemCnt       : Word;
```

Položka obsahuje počet zadaných položek vybraného popisovače grafu.

```
m_wMaxExDscrItemCnt    : Word;
```

Položka obsahuje maximální počet položek vybraného popisovačů grafu, který se smí zadat. Tento počet se zadává při inicializaci objektu.

```
m_wExDscrItemSize      : Word;
```

Velikost jedné položky popisovače grafu. Tato velikost se mění podle jeho typu a je nastavována automaticky při inicializaci objektu.

```
m_pItemArr  : Pointer;
```

Ukazatel na paměť pro jednotlivé položky vybraného popisovače grafu. Paměť se alokuje při inicializaci objektu.

8.1.4.2. Metody

8.1.4.2.1. Init

```
constructor Init(bExDscrType:Byte; wExDscrItemCnt:Word);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

V těle se nastaví položka m_bExDscrType na hodnotu proměnné bExDscrType. Dále se nastaví m_wExDscrItemCnt na 0 a m_wMaxExDscrItemCnt na hodnotu proměnné wExDscrItemCnt. Velikost popisovače m_wExDscrItemSize se nastaví na hodnotu dle jeho typu. Nakonec se provede alokace paměti, na kterou ukazuje m_pItemArr. Do této paměti se budou ukládat jednotlivé položky vybraného typu popisovače.

8.1.4.2.2. Done

```
destructor  Done; virtual;
```

Destruktor slouží ke zrušení objektu a uvolnění paměti pro položky vybraného typu popisovače grafu. Na tuto paměť ukazuje m_pItemArr.

8.1.4.2.3. AddExDscrItem

```
function  AddExDscrItem(var Data):Boolean; virtual;
```

Funkce přidá další položku vybraného typu popisovač grafu. Funkce vrací FALSE, pokud se má přidat více položek než kolik je povoleno, viz m_wMaxExDscrItemCnt.

Pokud by se přidával popisovač jiného typu tak funkce tuto chybu nezjistí.

8.1.4.2.4. GetExDscr

```
procedure GetExDscr(var ExRecDscr:XtKing_1.TArchExRecDscr); virtual;
```

Procedura sestaví strukturu TArchExRecDscr používanou objekty komunikující s programem TheKing. Procedura je volána potomky objektu tChArchiveVirt.

8.1.4.2.5. GetExDscrItemsLen

```
function GetExDscrLen:Word; virtual;
```

Funkce vrací velikost paměti (byte) u položky ExFieldsAttrRec ve struktuře TArchiveRecDscr.

8.2. Knihovna ChAVirt

V knihovně je implementován objekt tChArchiveVirt, který podporuje čtení více záznamů určených časovým intervalem.

8.2.1. Objekt tChArchiveVirt

V knihovně je implementován objekt tChArchiveVirt, který podporuje čtení více záznamů určených časovým intervalem.

Objekt implementuje rozhraní pro objekty komunikující s programem TheKing. Tento objekt implementuje několik metod, které musí být vždy přetíženy. Pokud tyto metody nebudou přetíženy, vrací se RunError(211), Call to abstract method.

Při čtení záznamů lze rozlišit dva čtecí módy – inkrementální a absolutní. Při inkrementálním čtení se přečtou všechny nové záznamy od posledního ukončeného čtení (CloseArchive). Pokud dojde ke ztrátě komunikace během operace čtení, vyšlou se v dalším čtecím cyklu stejná data. Při absolutním čtení se vrací všechny záznamy, které splňují zadaný časový interval.

Aby se odstranil problém zobrazování stejného záznamu víckrát v programu TheKing, je čas nejstarší záznamu omezen pomocí položky m_SecDiff, viz cMinSecDiff..

Protože práce s archivem je rozdělena mezi min. dva procesy, jeden zapisovací a jeden či více procesů čtecích, může se stát, že data označená pro přečtení budou přepsána novými daty. Proto se může stát, že v jednom čtecím cyklu existují data pro

přečtení a v dalším čtecím cyklu už ne. V tomto případě došlo ke ztrátě dat, což je signalizováno při zavírání archivu.

Pokud je archiv čten z více procesů, musí mít každý proces vlastní instanci tohoto objektu. Při použití více procesů vzniká problém při zápisu nových dat, protože proces zapisující nová data musí čekat až se dostane na řadu.

Funkce provádí synchronizaci dat pomocí zamykání Kernel funkcí ExLockKernel.

8.2.1.1. Položky

`m_wLastError : Word;`

Položka obsahuje kód poslední chyby při práci s objektem.

`m_SecDiff : LongInt;`

Hodnota se zadává v sekundách a určuje čas posledního přečteného záznamu z řídicího systému. Tento čas je vypočten jako (aktuální čas – m_SecDiff). Tímto zpožděním je zajištěno, že nebude v programu TheKing se nebude stejný záznam objevovat vícekrát. Tento problém nastává, pokud se v jednom časovém okamžiku do archivu ukládají dva záznamy se stejným časem a mezi jejich uložením se budou číst záznamy.

`m_ChAFlags : tSetChArchiveFlags;`

Množina příznaků popisující stav objektu při čtení dat.

`m_IxFinished : LongInt;`

Index záznamu, který byl naposledy přečten při poslední ukončené komunikaci. Od tohoto indexu se bude v inkrementálním módu pokračovat při čtení záznamů.

`m_FinishedTime : LongInt;`

Čas záznamu, který byl naposledy přečten při poslední ukončené komunikaci. Od tohoto času se bude v inkrementálním módu pokračovat při čtení záznamů.

`m_IxReadFrom : LongInt;`

Index obsahuje spodní mez při čtení archivu. Záznam s tímto indexem se bude vysílat.

`m_IxReadTo : LongInt;`

Index obsahuje horní mez při čtení archivu. Záznam s tímto indexem se už nebude vysílat.

`m_NextRecordTime : LongInt;`

Čas záznamu, který se má přečíst při dalším čtení. Pokud se čas bude lišit došlo k přepisu dat a musí se najít další záznam.

`m_LastRecordTime : LongInt;`

Čas záznamu, který se má číst jako poslední. S použitím tohoto času se určí, které záznamy lze ještě odvysílat a které už ne.

`m_pArchive : pArchiveVirt;`

Ukazatel na objekt předchozí vrstvy, tj. potomka tArchiveVirt.

8.2.1.2. Metody

8.2.1.2.1. Konstruktor Init

```
constructor Init(pArchive:pArchiveVirt; var ChADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

V těle konstruktoru se nastaví položky objektu na následující hodnoty:
m_pArchive a m_SecDiff na předané hodnoty v pArchiveVirt a ChADesc,
m_FinishedTime, m_IxFinishedTime se nastaví na odpovídající hodnoty nejstaršího prvku v archivu,
m_NextRecordTime, m_LastRecordTime na hodnotu cMinValidTime,
m_IxReadFrom, m_IxReadTo na 0,
a příznaky m_ChAFlags se smažou.

8.2.1.2.2. Destruktor Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu. Destruktor objektu předchozí vrstvy se nevolá.

8.2.1.2.3. GetLastError

```
function GetLastError:Word;
```

Funkce vrací kód poslední chyby.

8.2.1.2.4. SetLastError

```
procedure SetLastError(wCode:Word);
```

Funkce nastaví položku m_wLastError na hodnotu wCode.

8.2.1.2.5. OpenArchive

```
procedure OpenArchive(BeginTime, EndTime:Longint; var FirstTime,
                            LastTime, Length:LongInt); virtual;
```

Funkce otevře archiv pro čtení v intervalu začínajícím BeginTime (včetně) a konče EndTime(mimo). Při této operaci se nastaví položky objektu m_IxReadFrom, m_IxReadTo, m_NextRecordTime a m_LastRecordTime. Funkce vrátí časy prvního a posledního záznamu, který se bude číst a jejich velikost (byte).

Při otvírání archivu se rozlišuje inkrementální čtení (přečtou se pouze nové záznamy) a absolutní (přečtou se všechny záznamy v zadaném intervalu). Při inkrementálním čtení se použijí položky m_IxFinished a m_FinishedTime.

Protože zápis dat do archivu probíhá nezávisle na čtení dat, může dojít k přepisu dat a velikost přenesených dat nemusí odpovídat vracené hodnotě Length.

Pokud čas EndTime je shodný nebo vyšší než aktuální čas v řídicím systému, vrací se záznamy, jejichž čas není starší jak (aktuální čas-m_SecDiff), viz cMinSecDiff.

8.2.1.2.6. CloseArchive

```
function CloseArchive:Boolean; virtual;
```

Funkce uzavře archiv po čtení záznamu a v případě inkrementálního čtení se nastaví následující položky m_IxFinished a m_FinishedTime.

Vracená hodnota signalizuje, zda při čtení došlo k přepisu dat.

8.2.1.2.7. Read

```
function Read(pRecord:Pointer;EndTime:Longint; var wLen:Word):
                              Boolean; virtual;
```

Funkce přečte z archivu záznam dle položek m_IxReadFrom, m_NextRecordTime. Další položky se použijí v případě, že došlo k přepisu dat.

Vrací se FALSE, pokud byl přečten poslední záznam nebo záznam nelze z archivu přečíst. Rozlišení těchto stavů se provádí pomocí wLen. Pokud wLen je 0, nelze záznam přečíst. V opačném případě byl přečten poslední záznam. Funkce vrací TRUE, pokud lze v dalším čtecím cyklu přečíst další záznam. Pozn. Přesto se může stát, že se při příštím čtení nevrátí žádný záznam, protože došlo k přepisu dat.

8.2.1.2.8. InitArchive

```
function InitArchive:Boolean; virtual;
```

Funkce nastaví archiv do inicializovaného stavu dle příznaků v zálohované paměti. Pokud je nastaven příznak af_SetFull, archiv se po inicializaci tváří jako plný. Pokud příznak af_SetFull není nastaven, archiv se tváři jako prázdný. V každém případě jsou záznamy zachovány a pomocí této funkce se nastavují pouze popisovače archivu v zálohované paměti.
Dále se nastaví následující položky:
m_FinishedTime, m_IxFinished na hodnoty odpovídající nejstaršímu záznamu v archivu,
m_NextRecordTime, m_LastRecordTime na hodnotu cMinValidTime,
m_IxReadFrom, m_IxReadTo na 0,
a příznaky m_ChAFlags se smažou.

8.2.1.2.9. GetState

```
procedure GetState(var ArchiveState:TRecArchiveState); virtual;
```

Podle stavu archivu se nastaví položky struktury TRecArchiveState. Při nastavování stavu jsou volány metody objektu z předchozí vrstvy. Tato struktura se předává objektům komunikujícím s programem TheKing.

8.2.1.2.10. GetId

```
function GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.
Tento objekt vždy vrací chat_ChArchiveVirt.

8.2.1.3. Metody pro práci s popisovačem tabulky

Metody jsou volány objekty komunikujícími s programem TheKing.

8.2.1.3.1. GetHeader

```
procedure GetHeader(var DataHeader:TArchDataHeader); virtual;
```

Funkce nastaví popisovač tabulky, který se předává objektům komunikujícím s programem TheKing.
Při vytvoření potomka tohoto objektu musí být metoda vždy předefinována, jinak generuje RunError(211), call to abstract method.

8.2.1.4. Metody pro popis zobrazení záznamů v grafu

Metody jsou volány objekty komunikujícími s programem TheKing.

8.2.1.4.1. AddExDscr

```
function AddExDscr(pExDscrObj:pExDscr):Boolean; virtual;
```

Funkce přidá další typ popisovače grafu. Pokud už byl přidán stejný typ nebo se přidává více typů popisovačů než bylo zadáno při inicializaci objektu, vrací se FALSE. V opačném případě se vrací TRUE.

Při vytvoření potomka tohoto objektu musí být metoda vždy předefinována, jinak generuje RunError(211), call to abstract method.

8.2.1.4.2. GetExHeader

```
procedure GetExHeader(w:word; var ExDataHeader:TArchDataExHeader);
                              virtual;
```

Funkce nastaví požadovaný popisovač grafu na pozici W. Ten se pak předá objektům komunikujícím s programem TheKing.

Při vytvoření potomka tohoto objektu musí být metoda vždy předefinována, jinak generuje RunError(211), call to abstract method.

8.2.1.4.3. GetExDscrCnt

```
function GetExDscrCnt:Word; virtual;
```

Funkce vrací počet zadaných popisovačů grafu.

Při vytvoření potomka tohoto objektu musí být metoda vždy předefinována, jinak generuje RunError(211), call to abstract method.

8.3. Knihovna ChADscr

V knihovně je implementován objekt tDscrChArchive. Tento objekt na rozdíl od svého předka implementuje funkce pro podporu popisovačů tabulky. Pokud použijete instanci tohoto objektu, budou se záznamy v programu TheKing zobrazovat v tabulce.

8.3.1. Objekt tDscrChArchive

Potomek tChArchiveVirt implementuje funkce pro podporu popisovačů tabulky. Pokud použijete instanci tohoto objektu, budou se záznamy v programu TheKing zobrazovat v tabulce.

8.3.1.1. Položky

```
m_pDscrObj  : pDscr;
```

Ukazatel na objekt spravující popisovače tabulky.

```
m_strArchiveName            : tArchName;
```

V položce je uloženo jméno archivu, které se bude zobrazovat v programu TheKing.

8.3.1.2. Metody

8.3.1.2.1. Konstruktor Init

```
constructor Init(pArchive:pArchiveVirt; var ChADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

V těle konstruktoru se volá konstruktor předka a poté se nastaví hodnoty položek m_pDscrObj a m_strArchiveName podle hodnot v ChADesc.

8.3.1.2.2. GetHeader

```
procedure GetHeader(var DataHeader:TArchDataHeader); virtual;
```

Funkce nastaví popisovač tabulky, který se předává objektům komunikujícím s programem TheKing.

V těle funkce se nastaví povolení inkrementálního čtení pro daný archiv podle nastaveného příznaku archivu af_DisableInc. Dále se s použitím metod objektu předchozí vrstvy nastaví další stavové proměnné. Popisovač tabulky je nastaven objektem tDscr.

8.3.1.2.3. GetId

```
function GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací chat_DscrChArchive.

8.4. Knihovna ChAEDscr

V knihovně je implementován objekt tExDscrChArchive. Tento objekt na rozdíl od svého předka implementuje funkce pro podporu popisovačů grafu. Pokud použijete instanci tohoto objektu, můžou se záznamy v programu TheKing zobrazovat v tabulce nebo v grafu.

8.4.1. Objekt tExDscrChArchive

Potomek tDscrChArchive implementuje funkce pro podporu popisovačů grafu. Pokud použijete instanci tohoto objektu, můžou se záznamy v programu TheKing zobrazovat v tabulce nebo v grafu.

8.4.1.1. Položky

```
m_pExDscrArr                : Pointer;
```

Ukazatel na paměť, do které se budou ukládat různé typy popisovačů grafu.

```
m_wExDscrCnt                : Word;
```

V položce je uložen počet zadaných popisovačů grafu.

```
m_wMaxExDscrCnt             : Word;
```

V položce je zadán max. počet různých typů popisovačů grafu. Tato hodnota se zadává při inicializaci objektu.

8.4.1.2. Metody

8.4.1.2.1. Konstruktor Init

```
constructor Init(pArchive:pArchiveVirt; var ChADesc);
```

Konstruktor slouží k dokončení inicializace objektu s virtuálními metodami a jeho položek.

V těle konstruktoru se volá konstruktor předka a poté se nastaví hodnota položky m_wMaxExDscrCnt na hodnotu v ChADesc. Položka m_wExDscrCnt se nastaví na 0. Nakonec se alokuje paměť o velikosti určené počtem položek m_wMaxExDscrCnt. Ukazatel na tuto paměť je uložen do položky m_pExDscrArr.

8.4.1.2.2. Done

```
destructor Done; virtual;
```

Destruktor slouží ke zrušení objektu a uvolnění paměti pro objekty tExDscr spravující popisovače grafu. Na tuto paměť ukazuje m_pExDscrArr. Destruktor objektu předchozí vrstvy se nevolá.

8.4.1.2.3. GetHeader

```
procedure GetHeader(var DataHeader:TArchDataHeader); virtual;
```

Funkce nastaví popisovač tabulky, který se předává objektům komunikujícím s programem TheKing.

V těle funkce se volá metoda předka, která nastaví strukturu TArchDataHeader. Poté se nastaví počet zadaných popisovačů grafu do položky ExHeaderCount ve struktuře TArchDataHeader.

8.4.1.2.4. AddExDscr

```
function AddExDscr(pExDscrObj:pExDscr):Boolean; virtual;
```

Funkce přidá další typ popisovače grafu. Pokud už byl přidán stejný typ nebo se přidává více typů popisovače než bylo zadáno při inicializaci objektu, vrací se FALSE. V opačném případě se vrací TRUE.

8.4.1.2.5. GetExHeader

```
procedure GetExHeader(w:word; var ExDataHeader:TArchDataExHeader);
                                virtual;
```

Funkce nastaví požadovaný popisovač grafu na pozici W. Ten se pak předá objektům komunikujícím s programem TheKing.

Pokud požadovaný popisovač grafu nebyl zadán, vrací se FALSE a struktura TArchDataExHeader je vynulována. V opačném případě se volá metoda objektu tExDscr uloženém na pozici W v položce m_pExDscrArr. Dále je nastavena položka NextExHeaderFl ve struktuře TArchDataExHeader dle požadovaného popisovače W a počtu zadaných popisovačů grafu m_wExDscrCnt.

8.4.1.2.6. GetExDscrCnt

```
function GetExDscrCnt:Word; virtual;
```

Funkce vrací počet zadaných popisovačů grafu, viz m_wExDscrCnt.

8.4.1.2.7. GetId

```
function GetId:Word; virtual;
```

Funkce vrací identifikaci archivu.

Tento objekt vždy vrací chat_ExDscrChArchive.