Memory card operation within Tecomat systems

TXV 003 43.01
third edition
May 2008
subject to alterations

Changes history

Date	Edition	Change description
February 2008	1	First edition
	2	Order number of the PM-1901 module corrected
May 2008	3	Order number of the TXV 003 43 documet added Title change from the original „Data saving to the memory card wi-thin Foxtrot systems" to the new title „Memory card operation within Tecomat systems"

CONTENT

1   INTRODUCTION

Tecomat Foxtrot and TC700 systems with the processor unit CP-7004 enable the usage of memory cards. The prime task of the card in these systems is the saving of web pages for the web server. The question is whether it is possible to use these cards for other purposes, too, for example for data saving from the PLC program. The answer is yes, however, certain conditions must be fulfilled. What are these conditions and what is good to be taken into consideration while programming is explained in this document.

1.1   The problem outline

Firs of all, some generally known data. Memory cards undertaken in the last few years a huge progress. The result is a great scale of card on the market, low price, big data capacity. Cards are mass-used, primarily, in cameras and mobile phones which to a great extend determines their features. For users of such devices are the most important criteria for the card selection the following parameters: card capacity, speed of a data access on the card (esp. during data entry), standard interface and the price of the card in comparison to its capacity. Card producers conformed to these criteria. One of the most spread cards nowadays are cards of the type SD, micro SD and MMC. For example, the vast majority of cameras support some of these formats, very often, any of them can be used. This, of course, influence the quantity of cards sold and thus also their price. The typical usage of cards in mobile phones and cameras then determines other features, too. Especially the file system used. The user wants to have a possibility to remove the card from the device and read it on the PC. So producers chose the FAT file system. It is notoriously known from the DOS operation system and is supported by all computers of the PC standard both by the Linux operation system and also by all Windows mutations. We postpone the question whether it was the best choice to someone else. If we want to use a common card, it will be formatted as FAT16 or FAT32. Until now, everything is without a problem. Where then is the difference between the usage of a memory card in for example a camera and the usage of it within the PLC? The first problem lies within the parameter which is usually opt out by camera users. This parameter is the maximum guaranteed number of entries onto the card which is in present round 100 000 entries. In other words, the card lifetime is specified mainly by the number of entries. For cameras or mobile phones usage it is a number so high that it is not important to the user. Simply, after taking 100 000 pictures, it might be the time to buy a new card. Within the PLC system, the situation is distinctively different. Let's suppose that the planned lifetime of the PLC in the technology is 10 year, the technology runs continuously and requires continuous saving of data recorded from the technology onto the card as a part of the documentation of a production process. Then, using a simplified mathematics, we found out that the number of entries to the card is = 100 000 : (365 x 10) = 27,397 daily. It means that if we undertake more than 24 entries daily, the card may not last the planned 10 years. The shown calculation is quite simplified because the parameter 100 000 entries involves each segment in the memory card so to ensure the calculation is valid, we would have to re-write only one region in the card during those ten years. During the real usage, the situation will be better because data will not be saved to one location on the card only. However, the reality remains that it is impossible to enter to the card new data each 100 ms because its lifetime excludes this. Another difference is connected with the entry speed. The photographer is interested in the speed of picture saving onto the card. If, during the saving, other operations take more time than other is, in this case, quite uninteresting. This implies the card producer who does the best to ensure a quick saving time of the whole

file. Therefore, the reality is, that not all operations during the entry onto the card take the same time, the vast majority of them run quickly, however, there are also moments when the card during the entry „contemplates" quite long. On the other hand, the PLC programmer will be interested in the information on how long can take an individual operation of the entry because this time will prolong the PLC cycle time. And the most important times will be right those when the card reacts slowly. These times can be identified just experimentally, the card producers do not issue them. The last significant difference between the usage of the memory card in the mobile device and the PLC is the situation when a device switch-off occurs. The supply of mobile devices is a battery so it is not a problem while switching the device off to finish all sathe user asks the operation system to do so. At that moment the operation system finish all entries and then announce that the card can be removed safely. When an electricity drop occurs, all users of PC knows that after a subsequent switch on, the operation system check the disc first. If during the cut off the entry was in progress then in better case the file is not saved or in a worse case the file system is damaged. In the worst situatio, the loss of not only the saving file but also of some other. This is commonly known. The conclusion is, that it is not possible to disconnect the system supply in any moment, otherwise, the data loss can occur. In PLC systems, the situation is more complicated because it includes the fact that the PLC system can be switched off without any risk anytime (not only by the Start button). The PLC processor is, of course, equipped by a battery but it serves to supply the circuit of real time and SRAM memory, so, to ensure the system to remember the date, time and stored variables assigned by the RETAIN program. The battery has not a sufficient capacity to maintain the system processor and other circuits running necessary to finish the entry onto the card. Consequently, a problem arises, how to ensure the file system on the card not to be damaged during the supply drop.

1.2   Task assignment

Shall we suppose the following assignment. The task for the PLC is to follow the chosen quantities of the controlled technology and save their values in files on the card. The saving can be either periodical (once per time unit) or on the occurrence basis (e.g. when the reqquantity changes its value by a set value). The data saving will be an integral part of the control. Files saved on the memory card must be easily transferable to the computer, both, to ensure the archivation during the process control or to enable further execution, for example, failure evaluation within the technology etc. To transfer files to the computer, the standard devices are used available on every computer. Files transferred to the computer must be removable from the card. The file access can be conditioned by a user name and password..

1.3   Proposed solution

It results from previous chapters that during the data saving onto the memory card, it is necessary to pay attention to the following problems:
- the frequency of file entries onto the card
- the extension of PLC cycle time during the file entry
- the supply switch-off during file saving onto the card
- the format of data saved
- the structure of directories where data will be saved to
- visibility of saved files for the web server

The frequency of file entries onto the card

To decrease the frequency of file entries it is recommended to save daty to the DataBox first. The DataBox is a memory of the SRAM type which content is backed-up in the Foxtrot system by an inbuilt storage battery and optionaly by a lithium battery (CR2032). The storage battery is able to retain the content of the DataBox for the period of the minimum of 500 hours. With the fitted battery the DataBox is backed-up for 20 000 hours. For details see TXV 00410. The system is standardly supplied without the battery. The battery can be added to the holder anytime. To the file on the card data form the DataBox will be saved usualy once or up to several times a day.

The extension of PLC cycle time during the file entry

During the file entry the cycle time is extended for a period necessary for the entry. This period is dependant on the card speed (see the appendix) and also on the operation that is currently undertaken. Due to this reason, it is necessary to spread some operation into several cycles so, that the total time would be as short as possible. Therefore, the described solution firstly tests doring several cycles whether set directories exist. If some does not, then it creates them and then opens the file for entry, saves data from the DataBox continuously into it and lastly, close the file. The file entry operation, therefore, takes several PLC cycles.

The supply switch-off during file saving onto the card

During the physical entry onto the card, it is not possible to switch the supply of the system off because there is a danger of damage of the file system on the card. This problem can be solved using the following ways.

The first possibility is to back-up the PLC supply (use UPS), monitor the transfer to the back-up supply and in case of a main power supply drop, the entry to the file can be ceased. In other words, condition the entry to the file by the existence of main power supply. It can be supposed that in case of power drop, not even the technology would be in operation, therefore, no data to be saved would exist. In case of back-up supply drop from the UPS everything is alright because there is no entry onto the card in progress, therefore, no data can be lost.

The second possible solution is to use a supportive supply module PM-1901 (order number TXN 119 01). The modulel PM-1901 is connected between the supply source and the basic module for Foxtrot. Its task is to ensure the supply of the basic Foxtrot module at the moment of main supply source drop for the time necessary for successful ceasation of entry operations. During the time when the saving is in progress, all data are still saved in the DataBox, so while the supply is switched off the entry of the whole file is not finished though, however, after the supply is switched on again the file can be saved once more. During the repeated saving, originaly saved data on the card are re-written.

The format of data saved

Theoretically, there is no restriction on format that data will be saved in. Data can be saved in the binary format, can be encrypted, comprised etc. Practically, it is advantageous to choose the format according to the way of further execution of files after their transmission to the computer. Such a commonly used format is CSV (Comma Separated Value) that is usually used for data trans-

fer between programs. This format is suitable for file import into table calculators and database programs. Data are saved as a text in lines, individual values in the line are divided by commas. Each line in the text corresponds to one line of values in the table calculator.

Example of values saved in the CSV format :

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

For data saving in this format, the data type STRING can be used where it is simple to transmit variables of different data types into. The crastion of one line of values from the above mentioned example can then look as follows:

```
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');
```

The advantage of the CSV format is its simplicity, easy creation, data readibility without any special tools (any text editor or viewer rather is suitable) and easy data transmissibility into other programs. The disadvantage is higher memory consumption and, consequently, higher demands on space on the card, e. g. in comparison to binary saved data.

The structure of directories on MMC/SD card

The root directory for file operations within PLC system is called ROOT. The PLC programmer can work only with such files and directories that are located in the ROOT directory. Other files and directories are not accessible from the PLC program. The ROOT directory is, as a consequence, the operation directory for the PLC programmer (it means that all paths set in the PLC program are paths within the ROOT directory). If it is supposed that data will be saved in longterm period of time, it is necessary to work with the structure of directories where data will be saved to. Generally, it can be advised that data saved by the PLC aplication program should separated from other files on the card, therefore, in the separate directory (e. g. DATA/). If one file is saved a day, it will be 365 files a year. It is then suitable to create an individual directory for each year and to ensure a synoptical overview also for each month. Such an example can be seen below.
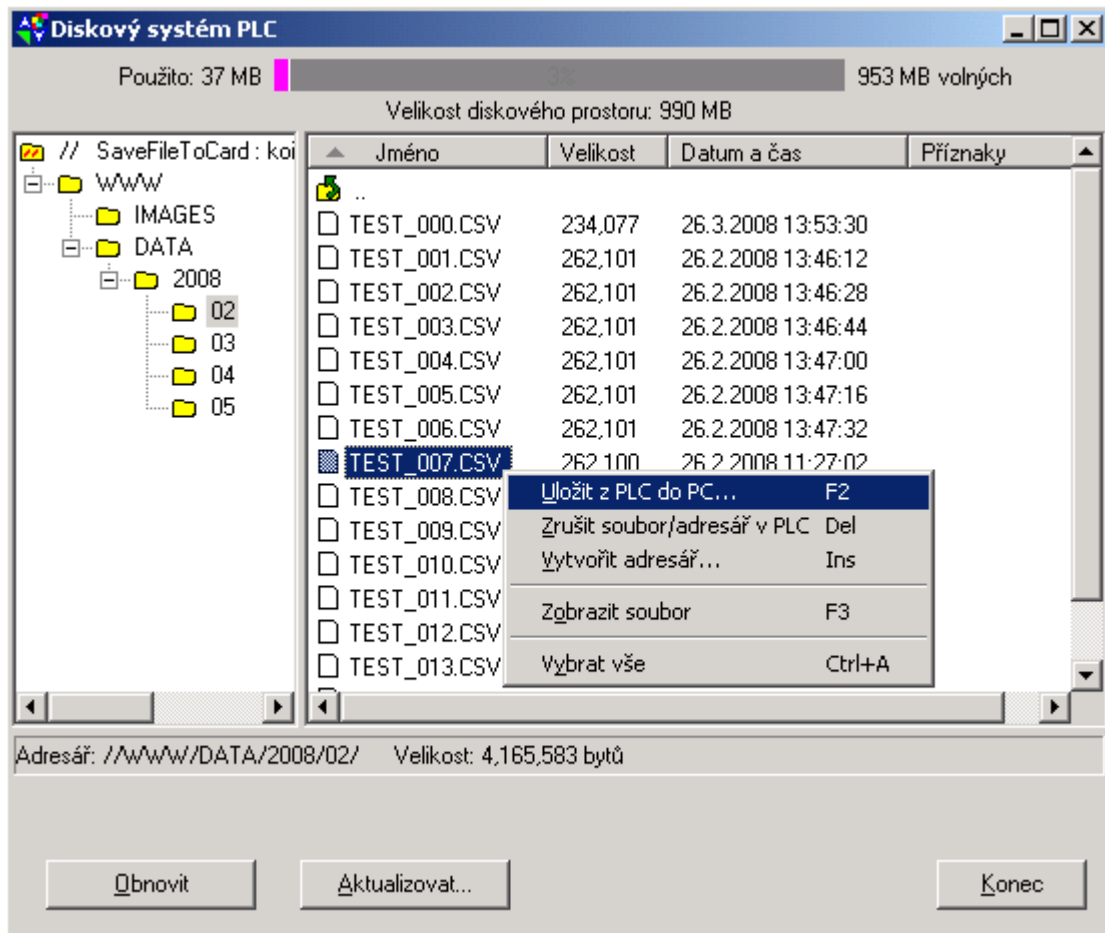
Visibility of saved files for web server

PLC web server uses as a root directory ROOT/WWW. If files are supposed to be accessible via the web interface, they must be saved in this path (e. g.. in the directory ROOT/WWW/DATA). If we want to view the directory content on the web page, it is sufficient to enter the link DATA/2008/. The maximum lenght of such presented files is 64. The dialog of the WEB Maker tool for setting the directory reference is shown on the following picture.

On the contrary, if files are saved on the same level as the directory WWW, they will not be accessible for web server. Such files can be transferred to the computer from the Mosaic programmable environment only (see the previous picture).
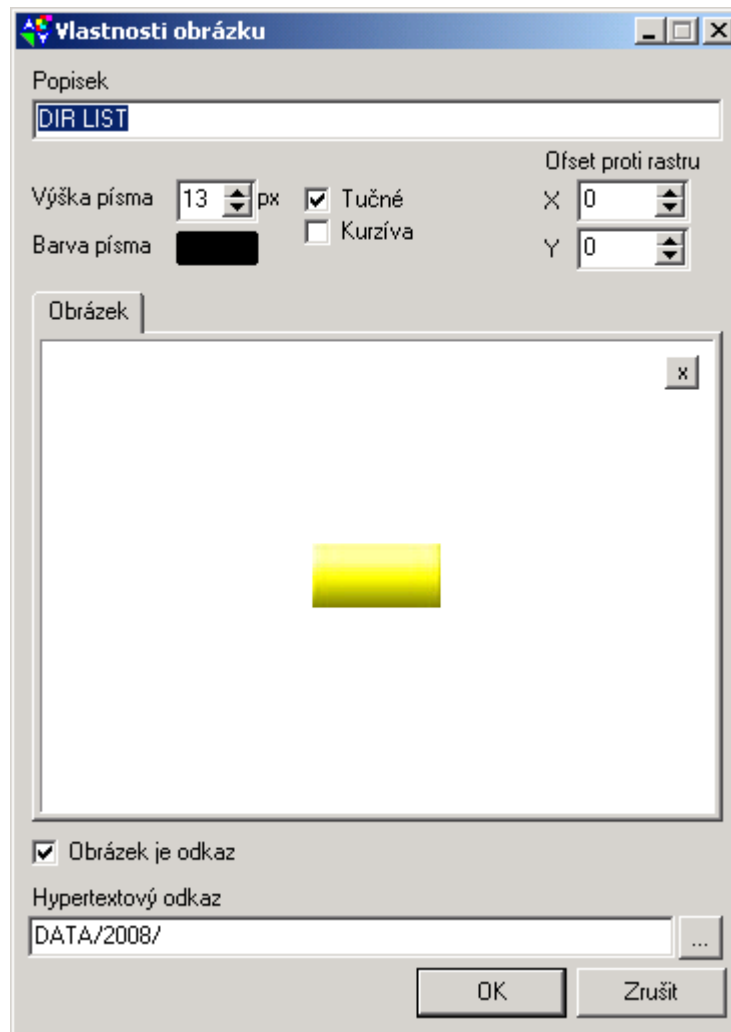
Other details are stated in descriptions of relevant PLC central units and in the description of the file operations library TXV00341_01.

The stated principles are used in the following example.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 1.4 Příklad řešení [/TITLE]
[GROUP] Úvod [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] HANDLE [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

1.4   Solution example

The example of solution of the task defined in the chapter.1.2 using principles from the chapter1.3 is possible to find in the project group MMC_SD_Card in the project SaveFileToCard. The example can be used as a foundation for task solution of a similar type as well as for parameters testing of a concrete memory card.

The main program saves into CSV file values of variables *tempVal* and *presVal* together with the time tag. The function block of *fbSaveRecToArchive* type is used for data saving which saves them into the buffer within the *DataBox* memory where, after the buffer is filled, they are entried to the file from. Function *PrepareRec* is used for the entry preparation which frames one sentence of the saved entry. Another function *CreateTestFileName* prepares the file name into which entries will be saved. The file name is compiled including the path where the file will be saved to.

The testing program is possible to control via the WebMaker tool window which serves either for web pages creation for the PLC or for debugging and control of the program from the Mosaic environment. Buttons in the right bottom part of the window have the following significance:

- START test start
- STOP test stopping
- CLEAR resetting and initialisation of buffers in the DataBox
- SAVE manual saving of the active buffer into the file
- DIR LIST directory link for directory content view on the web page (this button is functional only while the page is shown on the internet browser)
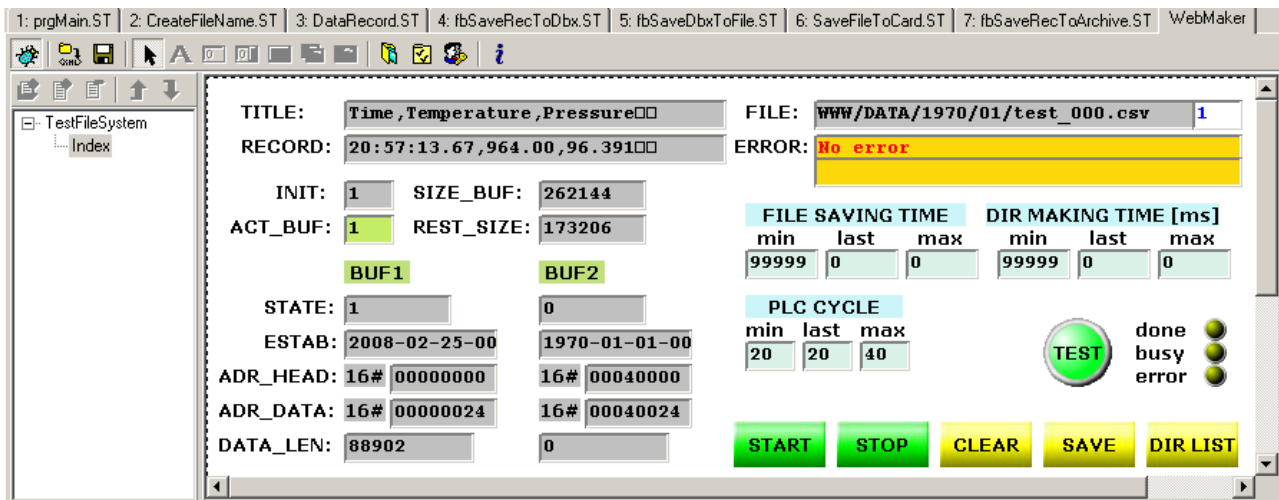
The significance of other buttons within the Web Maker window is as follows:
- TITLE saved entry header
- RECORD actual saved entry (time tag, the value of the variable *tempVal*, the value of the variable *přesVal*)
- FILE name of the file saved last
- ERROR error report

- INIT buffers in the DataBoxu are initialised
- SIZE BUF the size of the buffer used for entry saving in bytes
- ACT BUF buffer number where entries are actually saved
- REST SIZE remaining space that is in the active buffer ready for entry saving

In columns BUF1 and BUF2 is then indicated the status of both saving buffers in the DataBoxu:

- STATE   buffer status (0 = FREE, 1 = ACTIVE, 2 = CLOSED)
- ESTAB   date and time of buffer activation
- ADR_HEAD address in the DataBox where buffer control information are saved
- ADR_DATA   addres in the DataBoxu where entries are saved
- DATA_LEN   actual lenght of data saved

The button FILE SAVING TIME shows the minimum and maximum time required for the file saving. The DIR MAKING TIME button shows times necessary to directory creation. And lastly, the PLC CYCLE button shows PLC cycles times during the testing. When the TEST field lights-up, it indicates that the test is in progress, BUSY indicates that the entry is in progress and ERROR indicates an error during the file entry.

The Graph Maker tool window is ready to record the progress of file saving onto the card. It follows mainly the time required for saving within individual PLC cycles (*lastCycle*, time in ms). Other shown courses of action signalize when the card entry is activated (*diskLed*) and when required directories are created and when the entry itself takes place (*checkPath*). The analyser entry is initiated by pressing the „flag" icon in the upper control bar of the tool. Afterwards, it is necessary to wait till the file is saved onto the card and the buffer of the analyser in the PLC central unit is filled. After the analyser is full, the dialog will appear automatically, requiring the loading of recorded data from the PLC into the computer.
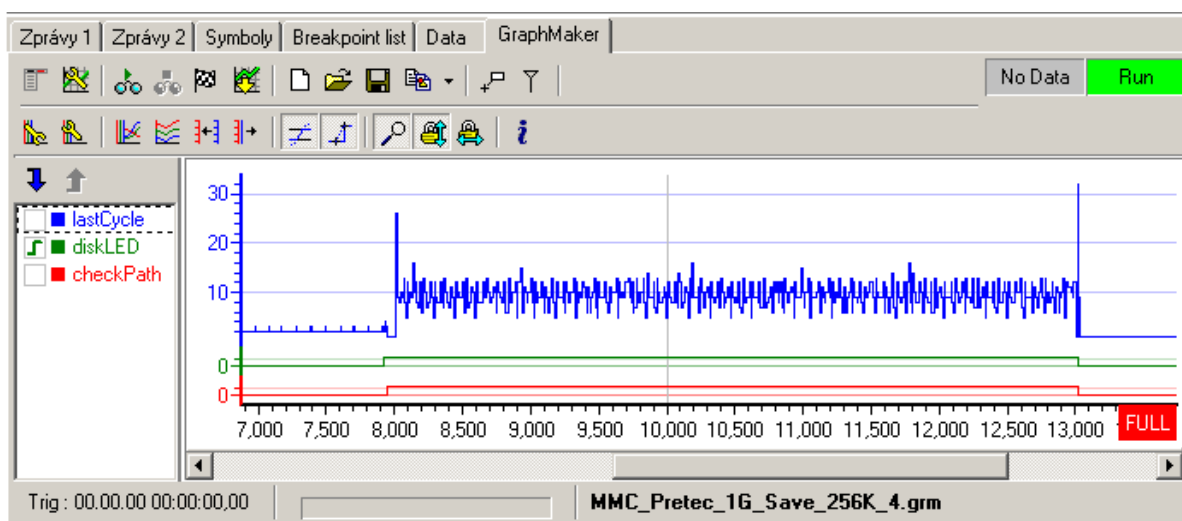
It is possible, for own tests, to create an identical copy of the whole project using the option Project | Copy project and by entering the name for the project copy. The copy then can by modified optionaly while the original project will stay saved unchanged and we can return to it anytime.
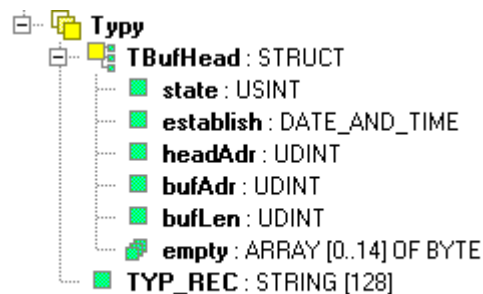
Following chapters describe all important ports of the program.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 2 Datové typy [/TITLE]
[GROUP] Datové typy [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
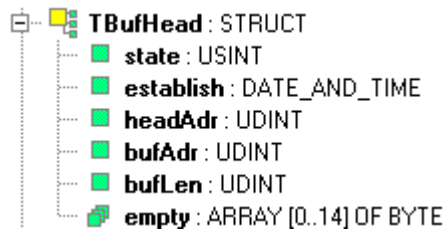[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

2 DATA TYPES

In the example SaveFileToCard are defined following data types:

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 2.1 Typ TYP_REC [/TITLE]
[GROUP] Datové typy [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] HANDLE [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

2.1 TYP_REC type

TYP_REC is a data type derived from the basic STRING type and is used for entry saved into the CSV file. Its size can be changed according to requirements, the maximum size is 255 characters.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 2.2 Typ TBufHead [/TITLE]
[GROUP] Datové typy [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] TBufHead [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

2.2   TbufHead type

TBufHead is a structure that is saved at the beginning of every buffer in the DataBoxu and contains information on actual buffer status.

```
TBufHead : STRUCT
    state : USINT
    establish : DATE_AND_TIME
    headAdr : UDINT
    bufAdr : UDINT
    bufLen : UDINT
    empty : ARRAY [0..14] OF BYTE
```

The significance of individual items of the *TBufHead* structure is as follows:
- *state*   buffer status
- *establish*   date and time of buffer creation (first entry saving)
- *headAdr*   address of this structure in the DataBox
- *bufAdr*   address of the first entry saved
- *bufLen*   actual lenght of data saved
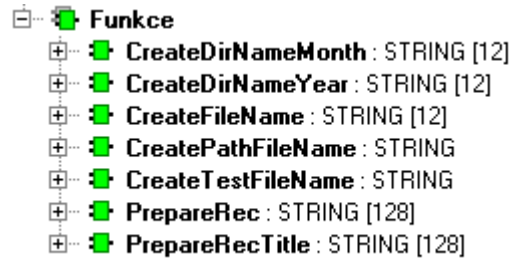- *empty*   reserve for other information

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 3 Konstanty [/TITLE]
[GROUP] Konstanty [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

3   CONSTANTS

In the FileLib library are defined following constants:

```
VAR_GLOBAL CONSTANT
    BUF_ACTIVE : USINT := 1
    BUF_CLOSED : USINT := 2
    BUF_FREE : USINT = 0
    CRLF : STRING [2] := '$0D$0A'
    DIR_PREFIX : STRING [9] := 'WWW/DATA/
    SEPARATOR : STRING [1] := ','
VAR_GLOBAL RETAIN
```
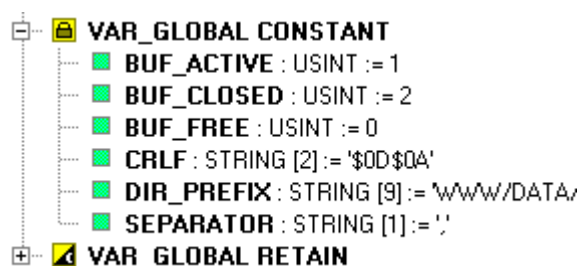
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 3.1 Konstanty BUF_ACTIVE, BUF_CLOSED a BUF_FREE [/TITLE]
[GROUP] Konstanty [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] BEGIN_POS [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

3.1   BUF_ACTIVE, BUF_CLOSED and BUF_FREE constant

BUF_ACTIVE, BUF_CLOSED and BUF_FREE constants are of the USINT type and are used as a buffer status for data saving into the DataBoxu.

Significance and values of constants are as follows:
BUF_FREE   *0,* buffer is free, buffer entries are saved in the file
BUF_ACTIVE   *1,* buffer is active, i. e. data saving is in progress
BUF_CLOSED   *2,* buffer is closed, data are ready for the file entry

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 3.2 Konstanta CRLF [/TITLE]
[GROUP] Konstanty [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]

[HIDDEN] CRLF [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

3.2   CRLF constant

CRLF constant is of the STRING[2] type, contains ASCII signs "vehicle return" and "new line" and is used as a line separator wihtin entries saved into the file.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 3.3 Konstanta SEPARATOR [/TITLE]
[GROUP] Konstanty [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] INVALID_HANDLE_VALUE [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

3.3   SEPARATOR constant

SEPARATOR constant is of the STRING[1] type, contains sign "comma" and is used as a separator of values within entries saved into the file. This constant can be, when necessary, modified to another separator, e. g. sign "tab".

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 3.4 Konstanta DIR_PREFIX [/TITLE]
[GROUP] Konstanty [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

3.4   DIR_PREFIX constant

DIR_PREFIX constant is of the STRING[9] type, has a value "WWW/DATA/" and sets the beginning of the path within saved file names. Also this constant can be modified as required.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 4 Úvod [/TITLE]
[GROUP] Funkce [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]

[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

4 FUNCTION

In the SaveFileToCard example, there are following functions defined:

The example contains followin functions for creation of names of files and directories:
- *CreateDirNameMonth*   Creates directory name according to the month
- *CreateDirNameYear*   Creates directory name according to the year
- *CreateFileName*   Creates file name according to the date
- *CreatePathFileName*   Creates file name incl. path according to the date
- *CreateTestFileName*   Creates testing file name

The example contains following functions for preparation of data to be saved:
- *PrepareRec*   prepares record in the CSV format
- *PrepareRecTitle*   prepares the record header in the CSV format

All functions are returned by STRING and can be modified as needed.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 4.1 Funkce PrepareRec [/TITLE]
[GROUP] Funkce [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

4.1   PrepareRec function

This function is neccessary to be rewritten according to the actual requirement.

The **PrepareRec** function prepares the record in the CSV format according to actual values of input variables *temp* and *pres*.

The **PrepareRec** function returns the string beginning with time data behind which values of *temp* and *pres* variables follow. Individual data are separated by a sign according to the *SEPA-RATOR* constant. The whole string is ended by the *CRLF* constant.

PrepareRec function :

```
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
```

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE]  4.2 Funkce PrepareRecTitle  [/TITLE]
[GROUP] Funkce [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN]  [/HIDDEN]
[HIDDEN_GLOBALS]  [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]
```

4.2   PrepareRecTitle function

This function is neccessary to be rewritten according to the actual requirement.

The **PrepareRecTitle** function prepares the header for an entry in the CSV format.

The **PrepareRecTitle** function returns the string: „`Time,Temperature,Pressure`" ended by the *CRLF* constant.

PrepareRecTitle function :

```
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
```

```
[TECO_HTML_TO_HTML_GENERATOR]
[TITLE]  5 Úvod  [/TITLE]
[GROUP] Funkční bloky pro ukládání souboru [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN]  [/HIDDEN]
[HIDDEN_GLOBALS]  [/HIDDEN_GLOBALS]
```

[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

5   FUNCTION BLOCKS FOR FILE SAVING

In the example SaveFileToCard, there are defined following function blocks:
- *fbSaveRecToDbx*   data saving to the buffer in the DataBox
- *fbSaveDbxToFile*   buffer saving from the DataBox into the file
- *fbSaveRecToArchive* record saving to the file with the usage of the buffer in the Da-
  taBox,   this function block uses both previous blocks

Function blocks use basic file functions from the FileLib library and analyse reading or file entry rather into more cycles so, that the time consumption of these operations is optimized.

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.1 Funkční blok fbSaveRecToDbx [/TITLE]
[GROUP] Funkční bloky pro ukládání souboru [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

5.1   fbSaveRecToDbx function block

The **fbSaveRecToDbx** function block is used for record from variable *rec* into the active
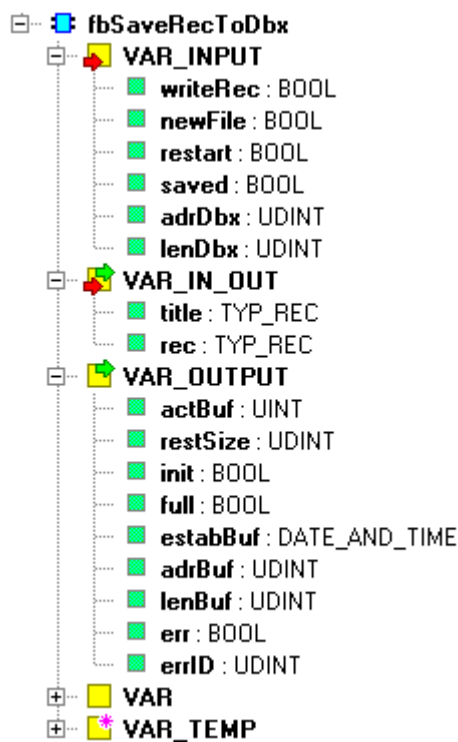
buffer in the DataBox. The record can be represented by any string and its saving is undertaken if there is a TRUE value in the *writeRec* variable. The record is always saved at the end of the buffer (behind the last saved record). After saving, the output variable *lenBuf* is increased of the lenght of the string saved.

In the *title* variable the header (first record) is awaited for the CSV format which is saved at the beginning of the buffer. If this string is empty, records will be saved without the header.

The TRUE value in the *newFile* variable causes that, firstly, the active buffer in the DataBox is closed and output variables are set as follows: *full* variable to the TRUE value (indicates that the buffer is ready for file entry), *estabBuf* variable sets the date and time of buffer creation, *adrBuf* variable contains buffer address where saved records begin and *lenBuf* variable shows the actual lenght of records saved. Then, the second buffer from the pair is set as active. Date and time of the buffer creation is saved in the header of this buffer, furthermore, the record from the *title* variable is saved at the beginning of this buffer and, eventually, the record from the *rec* variable is added (if the variable *writeRec* is TRUE). Closed buffer waits for saving till in the input variable *saved* the TRUE value appears. Afterwards, the status of the closed buffer changes to free buffer and it is ready for records saving again.

If the active buffer is filled up, the function block **fbSaveRecToDbx** will automatically close this buffer in the same way as a control variable *newFile* is set.

The TRUE value in the *restart* variable causes the initialisation of control structures for buffers in the DataBox, the first buffer is asigned as active, the second one as free, data saved in buffers are lost. In the DataBox, there are two buffers created, the first one can be found on the address shown in the *adrDbx* variable, the *lenDbx* variable states the total lenght of the space that will be used for both bufers. It means that each buffer will have the size of *lenDbx/2*.

During the first call of the instance **fbSaveRecToDbx,** after the supply switch-on, the initialization of control structures for buffers is undertaken in the DataBox according to the actual DataBox state, so, the record saving will continue where it ceased before switch-off.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	writeRec	BOOL	Record entry to the active buffer in DataBox requirement
	newFile	BOOL	Active buffer closure requirement and preparation of the closed buffer for file entry
	restart	BOOL	Buffenrs reset requirement
	saved	BOOL	Closed buffer was saved into the file
	adrDbx	UDINT	Memory address where in the DataBox buffers for records are placed
	lenDbx	UDINT	Space size for buffers in bytes
VAR_IN_OUT			
	title	TYP_REC	Title of saved records
	rec	TYP_REC	Saved record

	Variable	Type	Signification
VAR_OUTPUT			
	actBuf	BOOL	Active buffer number (1 or 2)
	restSize	UDINT	Free space size in the active buffer
	init	BOOL	TRUE indicates that control structures for buffers were initialized and the function block is ready to be used
	full	BOOL	TRUE indicates that one of the buffers in the DataBox was closed and is ready for file entry
	estabBuf	DT	Date and time of closed buffer extablishment *full = FALSE* the *estabBuf* has a value of DT#1970-01-01-00:00:00
	adrBuf	UDINT	DataBox address where records in the closed buffer are placed *full = FALSE* the *adrBuf* has a value of 0
	lenBuf	UDINT	Saved records in the closed buffer lenght *full = FALSE* the *lenBuf* has a value of 0
	err	BOOL	Error flag during the record saving to the buffer in the DataBox
	errID	UDINT	Error number (0 indicates no errors)

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.2 Funkční blok fbSaveDbxToFile [/TITLE]
[GROUP] Function blocks for file saving [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] WriteToFile [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

5.2   *fbSaveDbxToFile function block*
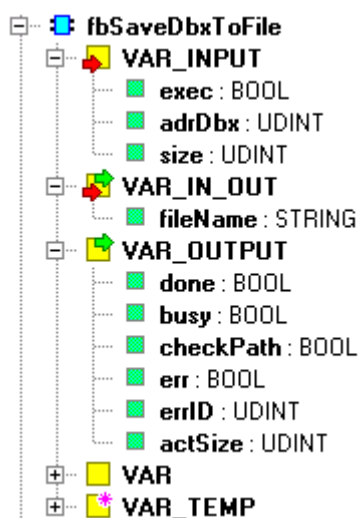
The **fbSaveDbxToFile** function block enters onto the entering edge of the zapíše *exec* variable a memory block from DataBox into the file. The address of the memory block beginning is stated by the *adrDbx* variable, the lenght of entered data is stated by the *size* variable. The file name (incl. path) where data will be saved to is stated by the *fileName* variable. Data entry into the file is not undertaken within one PLC cycle, the number of cycles is dependant on the lenght of entered data. If directories stated in the *fileName* variable do not exist, then the function block **fbSaveDbxToFile** creates those directories. If there is a file of the same name saved on the card within the *fileName* variable, then the content of this original file is overwritten with new data. During the file entry the output variable *busy* is set to the TRUE value. The *actSize* variable indicates how many bytes were entered to the file already. Providing that all data are successfuly entered and the file is closed, the output variable *done* is set.

The instance of the **fbSaveRecToDbx** function block is used in the **fbSaveRecToArchive** function block**.**

Variable description :

	Variable	Type	Signification
VAR_INPUT			
🔶	*exec*	BOOL	Control variable. Entering edge (transfer from FALSE to TRUE value) initiates the file entry.
🔶	*adrDbx*	UDINT	DataBox address where data are saved
🔶	*size*	UDINT	The size of data entered (number of bytes)
VAR_IN_OUT			
🔶	*fileName*	STRING	File name where data will be saved (incl. path)
VAR_OUTPUT			
🔶	*done*	BOOL	Has a TRUE value at the moment of the entry ceasation. Otherwise returns FALSE

	Variable	Type	Signification
	busy	BOOL	Má hodnotu TRUE během zápisu do souboru
	checkPath	BOOL	Sets itself to TRUE value at the moment when all required directories for file entry exist (for testing purposes)
	err	BOOL	The error flag during the file entry. If the operation was successfull, it has FALSE value, otherwise, TRUE.
	errID	UDINT	Error code. If the operation was successfull, errID = 0. In case of an error, errID <> 0.
	actSize	UDINT	Number of actual size of data entered

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 5.3 Funkční blok fbSaveRecToArchive [/TITLE]
[GROUP] Funkční bloky pro ukládání souboru [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

5.3   fbSaveRecToArchive function block
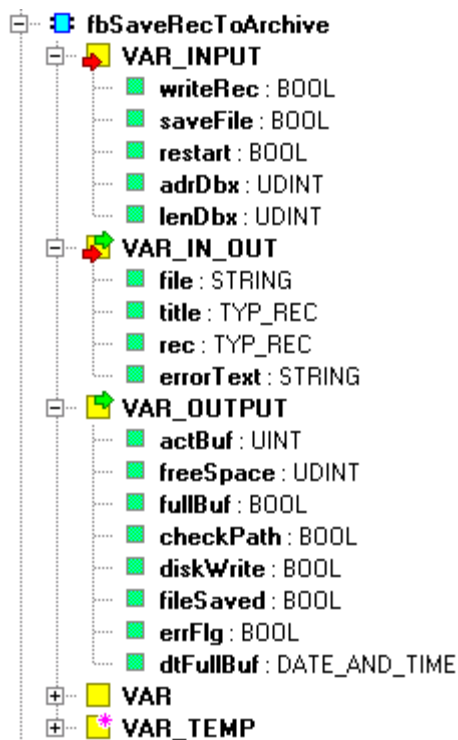
The **fbSaveDbxToArchive** function block uses blocks **fbSaveRecToDbx** and **fbSaveDbxToFile** to save records into the file using the memory of the DataBox for temporary data saving. Features of this block are therefore similar to features of blocks used.

The **fbSaveDbxToArchive** function block saves the record from the *rec* variable into the active buffer in the DataBox under the condition that the *writeRec* variable has a TRUE value. The record is always saved at the end of the buffer (after the last saved record). During saving the output variable *freeSpace* is decreased of the lenght of the string saved.

In the *title* variable the header (first record) is awaited for the CSV format which is saved at the beginning of the buffer. If this string is empty, records will be saved without the header.

The TRUE value within the *newFile* variable causes that, firstly, the active buffer in the DataBox is closed and output variables are set as follows: *fullBuf* variable = TRUE (indication that the buffer is saved into the file) and *dtFullBuf* variable contains date and time of the buffer creation. The file entry itself is initiated with a delay of one cycle after the *fullBuf* is set to enable the main

program to prepare the file name in the variable *fileName*. The second buffer in the DataBox is asigned as active. Into the header of this buffer the date and time of its creation is entered, then, at the beginning of the buffer the record from the *title* variable is saved and if needed then the record from the *rec* variable is added, too (when the variable *writeRec* is TRUE). The number of the actually active buffer is in the variable *actBuf*.

When the active buffer is filled, the function block Pokud **fbSaveDbxToArchive** closes this buffer automatically in the same way as if the control variable *newFile* is set.

The TRUE value in the *restart* variable causes the initialisation of control structures for DataBox buffers. The first buffer is asigned as active, the second one as free. Data saved in buffers will be lost. There are two buffers created in the DataBox, the first one can be found within the address shown in the *adrDbx* variable while the *lenDbx* variable states the total lenght of space that will be used for both buffers. That means that each buffer will have the size *lenDbx/2*.

The **fbSaveDbxToArchive** function block enters the closed buffer into the file automatically. The file name (incl. path) where data will be saved to is stated by the *fileName* variable. Data entry into the file is not undertaken within one PLC cycle, the number of cycles is dependant on the lenght of entered data. If directories stated in the *fileName* variable do not exist, then the function block **fbSaveDbxToFile** creates those directories. If there is a file of the same name saved on the card within the *fileName* variable, then the content of this original file is overwritten with new data. During the file entry the output variable *diskWrite* is set to the TRUE value. Providing that all data are successfuly entered and the file is closed, the output variable *fileSaved* is set.

During the first call of the instance **fbSaveDbxToArchive,** after the supply switch-on, the initialization of control structures for buffers is undertaken in the DataBox according to the actual DataBox state, so, the record saving will continue where it ceased before switch-off.

If the system supply was switched-off during data saving from the closed buffer into the file, then, during the repeated supply switch-on, this buffer is saved whole into the file again. The application program must in this case ensure that the name of the file in the *file* variable is identical. The outputvariable *dtFullBuff* ,that contains date and time of actually saved buffer, can be used for this
In case of an error during the file entry, the *errFlg* variable is set to the TRUE value and in the *errorText* variable is the error report.

Variable description :

	Variable	Type	Signification
VAR_INPUT			
	writeRec	BOOL	Record entry requirement into the active buffer in the DataBox
	newFile	BOOL	Active buffer closing requirement and entry of the closed buffer into the file
	restart	BOOL	Buffers reset requirement in the DataBox
	adrDbx	UDINT	Memory address where buffers for records in the DataBox are situated
	lenDbx	UDINT	The space size for buffers in bytes
VAR_IN_OUT			

	Variable	Type	Signification
	file	STRING	File name incl. path
	title	TYP_REC	The header of records saved
	rec	TYP_REC	Record saved
	errText	STRING	Error report
VAR_OUTPUT			
	actBuf	BOOL	The number of active buffer (1 or 2)
	freeSpace	UDINT	Free space size in the active buffer
	fullBuf	BOOL	TRUE indicates that one of the buffers in the DB was closed and the file entry is running
	checkPath	BOOL	Is set to TRUE value ath the moment when all required directories for file entry exist (for testing purposes)
	diskWrite	BOOL	TRUE value indicates active file entry
	fileSaved	BOOL	TRUE indicates ceasation of the file entry
	errFlg	BOOL	Error flag during data saving into the file
	dtFullBuf	DT	Date and time of closed buffer creation
	err	BOOL	Error flag during record saving into the buffer in the DataBox

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 6 Úvod [/TITLE]
[GROUP] Přílohy [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

6   ENCLOSURES

[TECO_HTML_TO_HTML_GENERATOR]
[TITLE] 6.1 Testování MMC a SD karet [/TITLE]
[GROUP] Přílohy [/GROUP]

[KEYWORDS] [/KEYWORDS]
[GLOBALS] [/GLOBALS]
[HIDDEN] [/HIDDEN]
[HIDDEN_GLOBALS] [/HIDDEN_GLOBALS]
[NOTHING] [/NOTHING]
[/TECO_HTML_TO_HTML_GENERATOR]

6.1   MMC and SD cards testing

Test conditions

The lenght of the file entered 256 KB. The file was saved to WWW/DATA/2008/02. Number of entries during testing 16. The testing program created required directories first and then entered 16 files with the lenght 256 KB. Within one PLC program cycle, max. one sector was saved, i. e. 512 bytes. All times shown in the table are in ms.

Cards tested

Types and sizes of cards tested were selected with the aim to test as wide scale of different cards as possible. Therefore, cards had different capacity. Cards with lower capacity were usually also historically older.

Parameters followed

Maximum time required for directories creation, minimum and maximum time needed for saving of one file and lastly, maximum time of one PLC cycle was followed. Tests were undertaken on CP-7004 with the firmware 3.0. Values measured are valid for the Foxtrot system, too, because the hardware of the processor unit is the same.

In contrast to parameters stated by producers who indicate in general the average speed of the card (average data flow), the user of the PLC is interested in the high peak demand on time, above all, the card entry operation. Right this time can prolong the period of PLC cycle that enters the file onto the card. This PLC cycle prolongation must be taken into the consideration during technology control.

Aims of testing

To verify functioning of the controllers for MMC/SD cards, functioning of the library for file operations, to compare features of individual memory cards. Other aim was to create the data file where can be compared parameters measured within the new, not yet tested, card type and thus investigate whether the new card is suitable for PLC use or not.

Tests evaluation

Cards tested show apparent differences. The improper card format can significantly influence the usage of the card within the PLC system (see the Corsair card results). The „so called" winner is the SD card Cheetah 133x 1G Pretec producer which showed good overall time of entry whereas max times added to the cycle period during the entry were on the level of 65 miliseconds. Typically, it was round 7ms per one cycle when the entry took place within. Very good was also the MMC card MyFlash 128 MB A-Data producer, disadvantege was the card capacity.
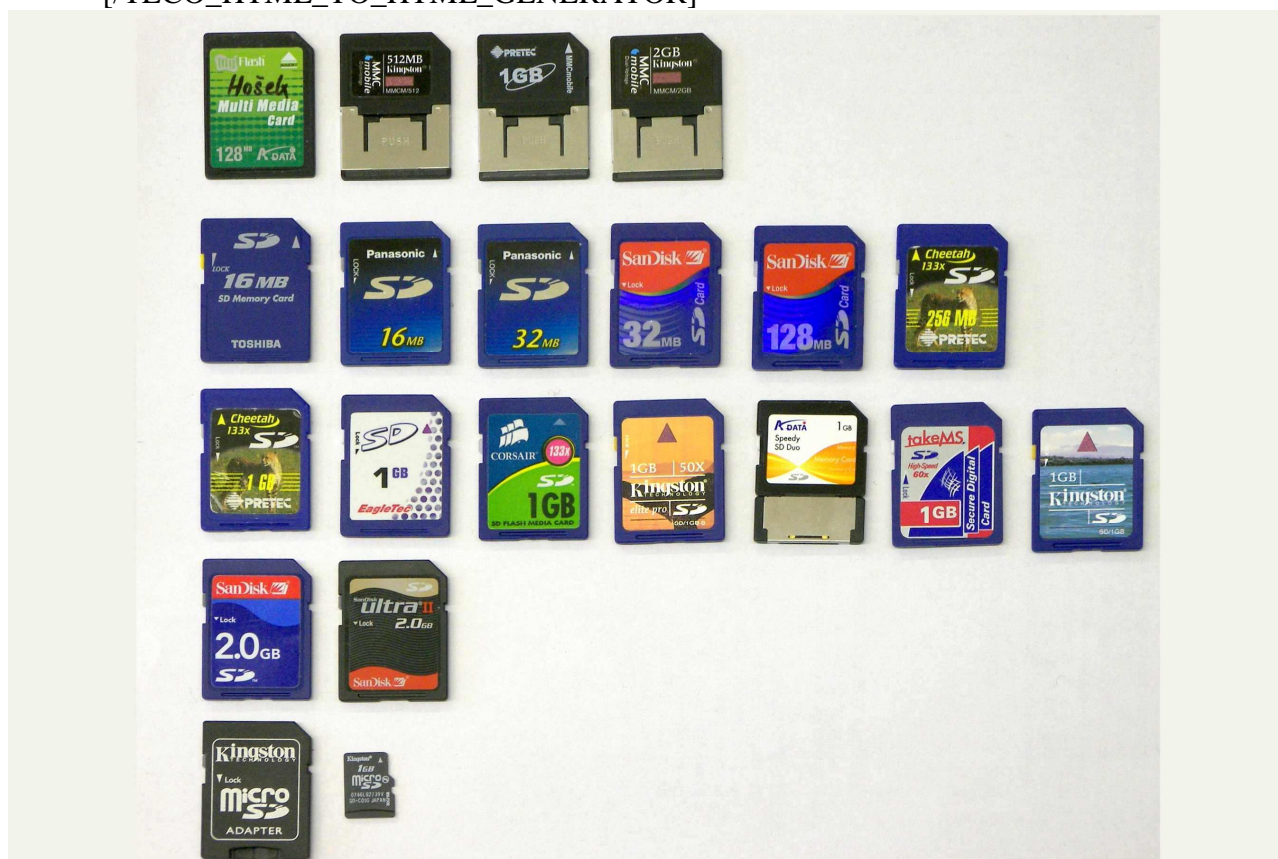
Non-coloured lines of the table were measured by the first version of the testing SW, therefore, they can contain slightly worse values than if they were tested by the final SW version.

CARD	PRO-DUCER	CAPAC-ITY	SAVING TIME MIN	SAVING TIME MAX	CY-CLE MIN	CY-CLE MAX	CREATE DIR TIME
SD Ultra II.	SanDisk	2 GB	1793	1823	1	41	
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

CARD	PRO-DUCER	CAPAC-ITY	SAVING TIME MIN	SAVING TIME MAX	CY-CLE MIN	CY-CLE MAX	CREATE DIR TIME
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

[TECO_HTML_TO_HTML_GENERATOR]
[/TECO_HTML_TO_HTML_GENERATOR]

Obr.1 Cards tested