SIEMENS

SIMATIC

WinCC Option
User Archives

Description 08.97

C79000-G8263-C113-01

C79000-G8263-C113-01
Printed in the Federal Republic of Germany

Siemens Aktiengesellschaft

Table of Contents

1 Introduction

WinCC is a process visualization software with which you can operate and monitor procedures in your automated process clearly and easily.

This description contains the configuration guide for the WinCC user archive option by means of an example. You can find parts of this example in the WinCC introductory system project on the WinCC CD-ROM. Additionally, this description contains the function description for the script functions and the message frames that are available for data exchange with the SIMATIC S5 programmable logic controllers.

Functional Extent of the User Archive Option

Displaying Values

- As tables (simple, direct assignment of table fields to archive fields)
- As forms (process pictures) / I/O fields (assignment of the archive fields by means of action scripts)

Reporting

- As tables (simple, direct assignment of table fields to archive fields)

Transferring from/to SIMATIC S5

- Complete data records of an archive
- Individual data fields of a data record
- Communication serial RK512/3964R or SINEC industrial ethernet (H1 layer 4)

Editing Possibilities

- In the table display
- In forms / I/O fields (with action scripts)

Operating

- In tables by means of a standardized button bar
- In forms by means of buttons with action scripts

Deleting or Adding Data Records

- Adding data records to tables by means of a button bar
- Adding and deleting data records in forms by means of buttons with action scripts

Legend

 Displays entries by means of the left mouse button

 Displays entries by means of the right mouse button

`Text` Text that you enter yourself and then select again is displayed in the `Courier` font.

File → New All of the menus, functions, and input boxes provided by WinCC that are selected are displayed in *italics*.

2 General Information about Action Scripts

Use the following steps to configure an action script (action):

R on the object on which you want to load an action (for example, button)

Select *Properties*

In the *Properties* or *Event* tab, select the element to be linked with an action and click on it with the left mouse button (for example, to configure an action for a left mouse click, select *Event / Mouse / Left Click*). The C code can then be input directly and compiled.

Use *OK* to end the configuration of the action.

3 Configuring User Archives

The first step is the configuration of a new archive of the user archive type. The following configuration steps are necessary for this:

- Creating a user archive
- Setting the archive fields
- Creating a table window
- Inserting the table in a process picture
- Working with the user archive in RUNTIME mode

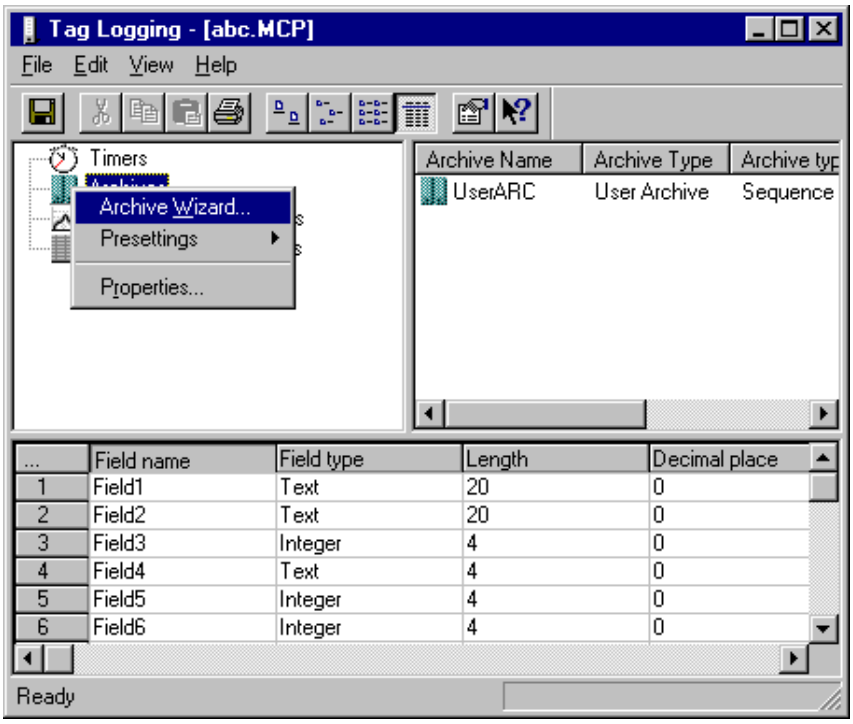
These steps are also described in the WinCC manual "Tag Logging" section.

3.1 Creating a User Archive

Start the "Tag Logging" editor from the WinCC Control Center.

R Click on *Tag Logging* and select the menu item *Open*.

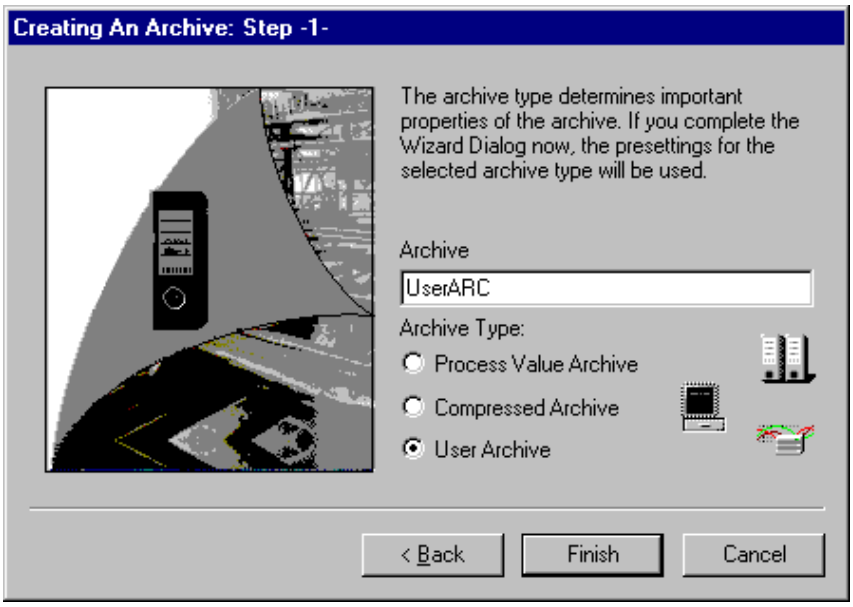
After the Tag Logging editor is started, you can create a user archive.



R Click on *Archives* and select the menu item *Archive Wizard...*.

This starts the Archive Wizard with which you can create different archive types in a simple manner.

Confirm the first page of the Archive Wizard by clicking on the *Next* > button.



You can create a user archive using the page that then appears.

Select *User Archive* as the archive type and enter the archive name `UserARC`.

Exit the archive creation with the *Finish* > button.

An empty user archive is created.

3.2 Setting the Archive Fields

The next step is setting the structure of the user archive data record. The individual archive fields are created for this purpose.


R In the "Tag Logging" window, click on the archive name `UserARC` and select the menu item *Properties...*.

In the User Archive Properties dialog, select the *Archive Fields* tab.

This selection provides an input table in which you can create the desired archive fields.


Double click on the first table element in the *Field name* column to enter any name for the archive fields. Finish the entry by using the Tab key.

The names you enter are used to later arrange the fields for the display in table form.

Note: You can configure up to 999 archive fields.


Double click on the second table element in the *Field type* column to enter the data type for the archive fields.

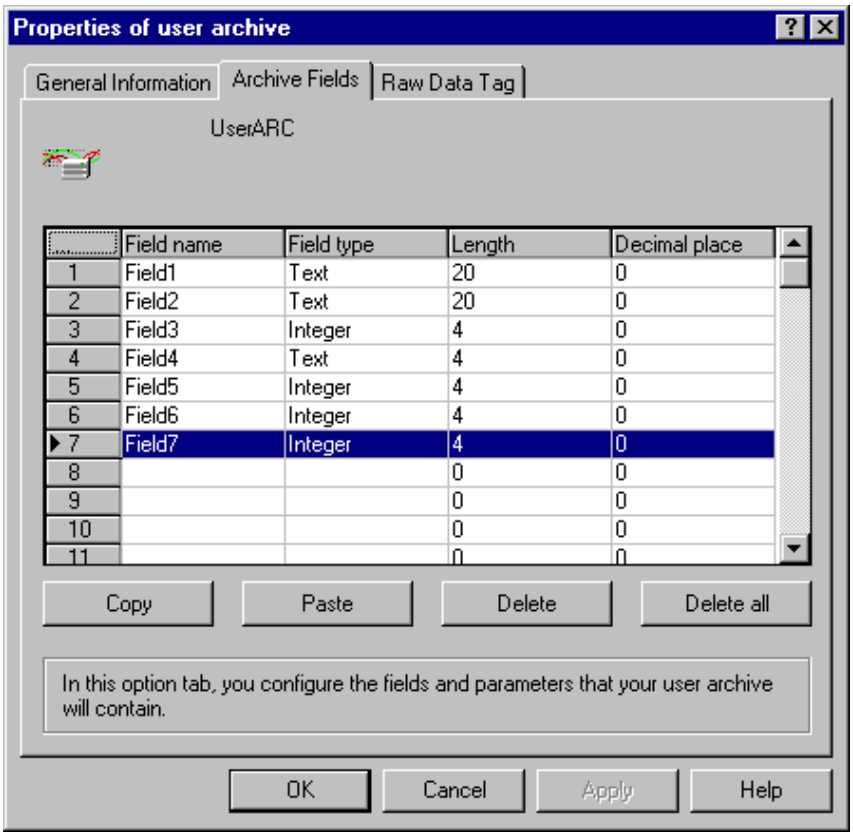
The following data types are available:
Integer filed, double field, text field, and date field.

Finish the entry by using the Tab key.

Note: After you select the data type, the *Length* (in bytes) table field is automatically set. This field depends on the data type and should only be changed in the case of the *Text* data type.

Tip: To save time, you can preset the *Field name*, *Length*, and *Decimal place* table fields by double clicking on the *Field type* table field, selecting the field type from the list, and exiting with the Tab key. This sets the *Field name* table field with the default name "Fieldx" (x is the sequential number of the table field) and the *Length* and *Decimal place* table fields receive the values appropriate to the selected field type.

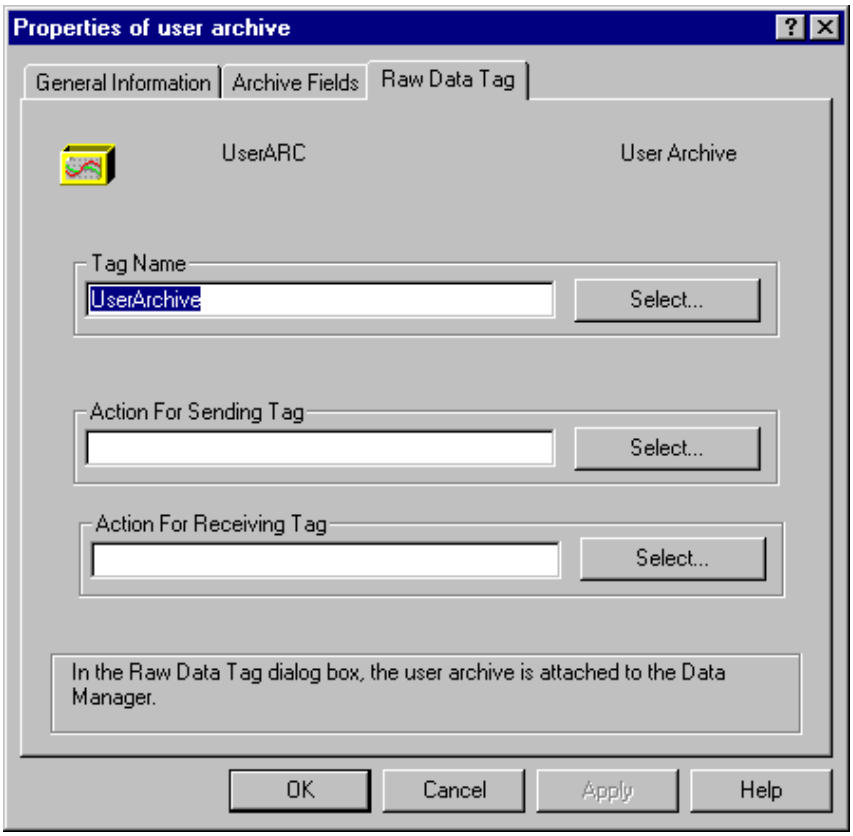
Enter the archive fields according to the following figure.



For later data exchange with SIMATIC S5 programmable logic controllers, a *raw data tag* type process tag is used.

Select the *Raw Data Tag* tab in the User Archive Properties dialog.

You receive the input dialog for entering the raw data tag.



Click on *Select...*

In the tag browser dialog that appears, you can select the desired raw data tag.

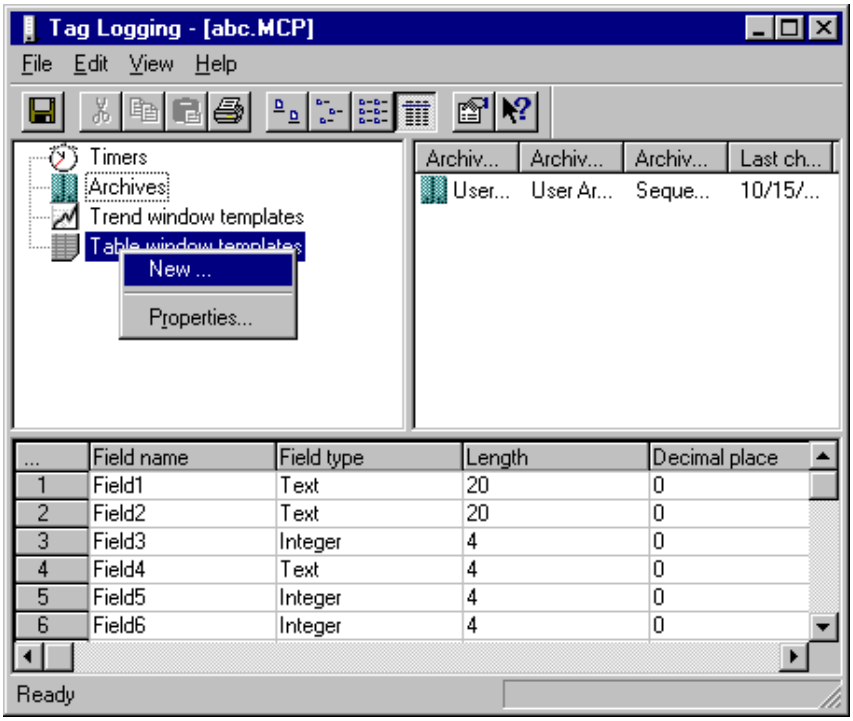
If you have not yet created any raw data tags, you can do so by clicking in the dialog on the  icon to add new tags. Then you can create a new raw data tag for the communication connection you selected.

This concludes the creation of the structure of the user archive.

Note: If you do not need any communication with the PLC but are using the user archive simply for structured data storage, you do not need to configure the raw data tag.

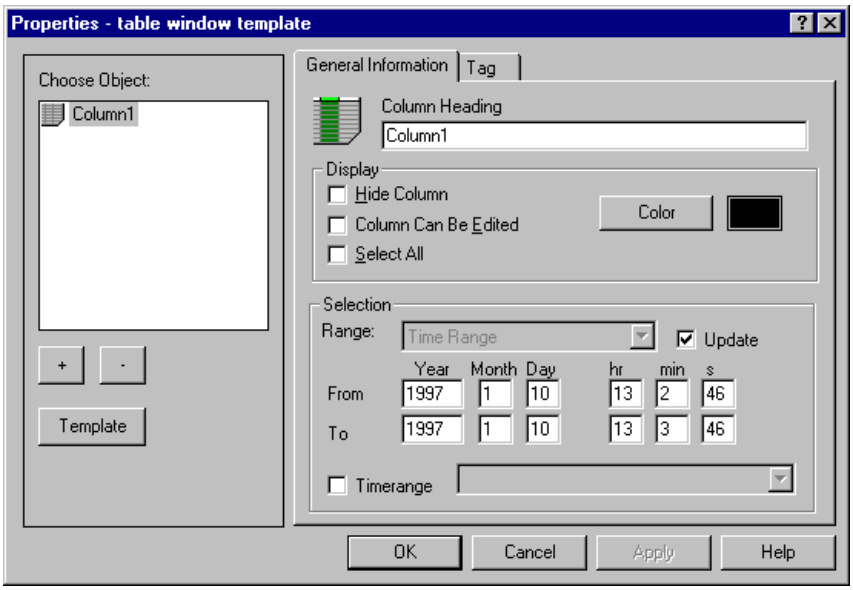
3.3 Creating a Table Window

The simplest and most comfortable way to display and maintain the data in a user archive is a WinCC table display. For this, you can use Tag Logging to create a table window template to link with the fields of the user archive.



R Click the right mouse button on *Table window templates* and select the *New*... menu item.

In the *Properties - table window template* dialog, you can create the columns for the table display.



In the *Choose Object* window section, you can see a column object named *Column1*.

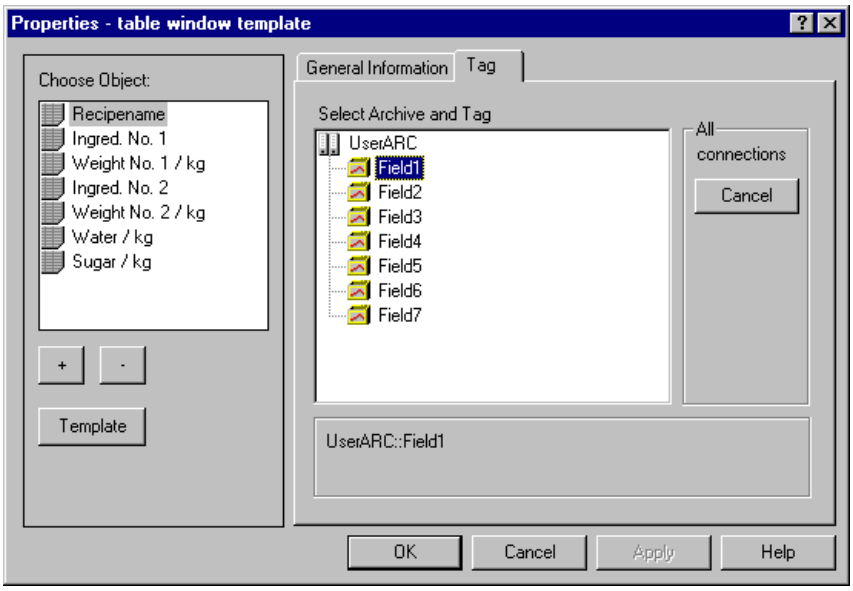
Click on the + button and add further column objects *Column2* to *Column7*.

Select the column objects sequentially and change the names according to the figure. This concludes the structure of the table title.

Now each column object can be assigned to an archive field.

Click on the *Tag* tab. In the dialog window that appears, you can double click to select the `UserARC` user archive. The archive fields available therein are displayed.

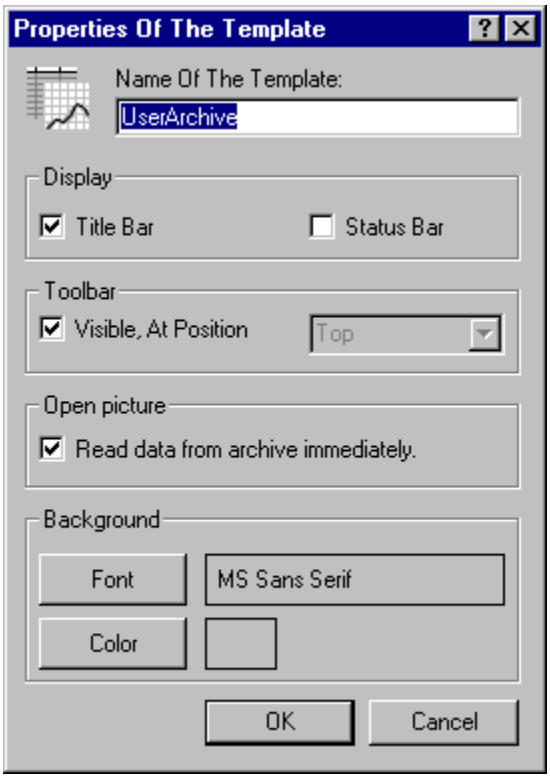
Click on each column object followed by the archive field to be linked to it (Figure: *Recipename + Field1*).



Once you have linked each column object with an archive field, the tag link of the table window with the user archive is finished.

Finally, you must configure the general properties of the table window.

Click on the Template button and enter `UserArchive` as the *Name Of The Template* and then close the dialog by clicking on *OK*.



This concludes the creation of the table template.

Close the configuration of the table template by clicking on *OK*.

3.4 Inserting a Table in the Process Picture

The table template you created is now inserted in a process picture. Open *Graphics Designer* with a new picture.

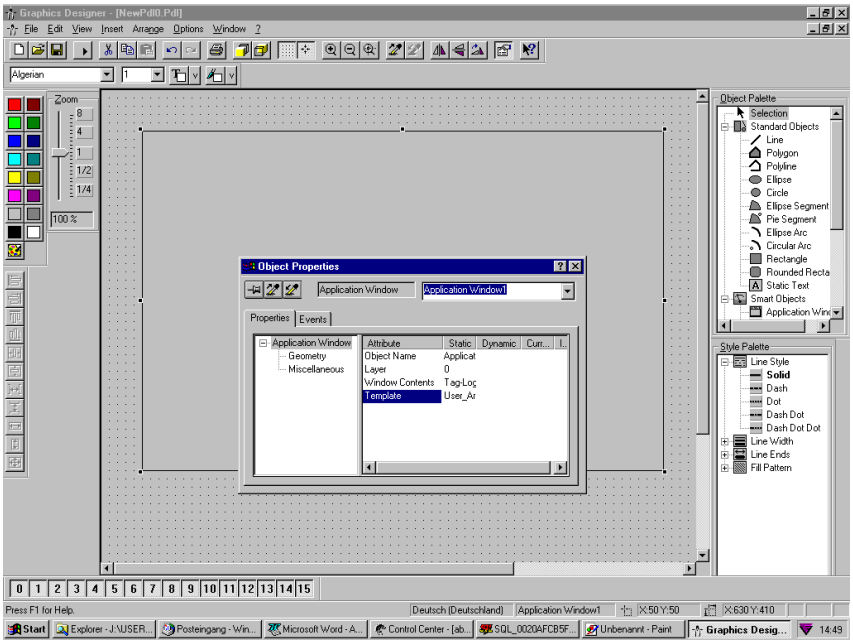
Note: The WinCC introductory system DEMO.MCP already contains two sample pictures, *UserARC.PDL* and *UserARC2.PDL*, that you can use for this.

Select the object group "Smart Objects" from the *Object Palette*.

Click on the *Application Window* object and create a window of sufficient size in the picture area by dragging the mouse.

In the browser that appears, select *Tag Logging* as the window contents and confirm your selection by clicking *OK*.

In the next browser, select *UserArchive* as the template and confirm your selection by clicking *OK*.



This concludes the creation of the table window for displaying and controlling the user archive.

Save the picture and start the WinCC run-time system.

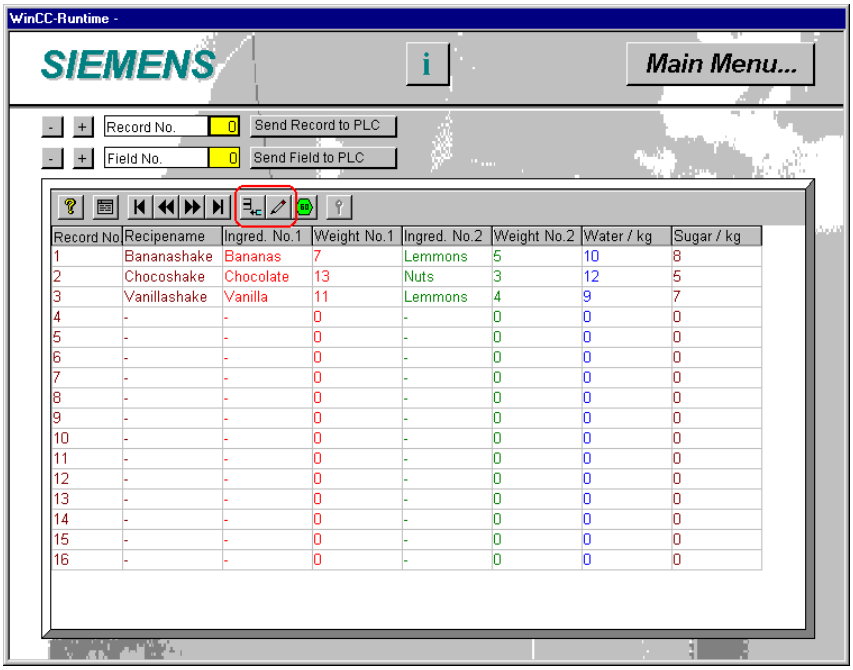
3.5 Working with the User Archive in RUNTIME Mode

The table window of the user archive offers special possibilities for operation:

Use this button to load new data records. You can enter one or multiple data records. The data records are then created and initialized. It takes approximately one second to load a data record.

Use this button to switch to input mode. If the button is clicked you can activate the input by double clicking on the data record fields. The input is concluded with *ENTER*.

Use these buttons to page forwards and backwards through the data records in the table window or select the start or end of the archive.



Record No	Recipename	Ingred. No.1	Weight No.1	Ingred. No.2	Weight No.2	Water / kg	Sugar / kg
1	Bananashake	Bananas	7	Lemmons	5	10	8
2	Chocoshake	Chocolate	13	Nuts	3	12	5
3	Vanillashake	Vanilla	11	Lemmons	4	9	7
4	-	-	0	-	0	0	0
5	-	-	0	-	0	0	0
6	-	-	0	-	0	0	0
7	-	-	0	-	0	0	0
8	-	-	0	-	0	0	0
9	-	-	0	-	0	0	0
10	-	-	0	-	0	0	0
11	-	-	0	-	0	0	0
12	-	-	0	-	0	0	0
13	-	-	0	-	0	0	0
14	-	-	0	-	0	0	0
15	-	-	0	-	0	0	0
16	-	-	0	-	0	0	0

The figure displays the completed configuration of a table with current data from the user archive.

4 Default Functions for User Archives

4.1 Overview and Scope

The default functions make it simple to use user archives. The basis for the default functions is the programmer interface of the user archives.

Use the default function TlgConnectToArchiv to create a connection to a user archive. This function constructs a manager that contains a "temporary data record," a position pointer, and an assignment table in which the archive name is assigned an archive identifier ("dwConnection").

The "temporary data record" is an image stored in memory of the data record on which the position pointer is located. Use the functions TlgGetRecord, TlgAddRecord, TlgNextRecord, TlgPrevRecord, TlgFirstRecord, and TlgLastRecord to move the position pointer.

By means of the archive identifier ("dwConnection"), a unique assignment to a user archive exists. This assignment also allows an indirect address like that needed for screen dialog boxes, for example.

The function TlgUpdateRecord stores the "temporary data record" in the database and overwrites the data record on which the position pointer is currently located. This data record must first be read by means of the functions TlgGetRecord, TlgNextRecord, TlgPrevRecord, TlgFirstRecord, and TlgLastRecord. You can read the number of the data record on which the position point is located by means of the default function TlgGetRecordFieldByNumber with the field number "0."

With the TlgSetRecordFieldByNumber function on the field number 0, the position pointer can be set to a specific data record that can then be loaded into the archive by means of the TlgAddRecord functions.

Use the TlgIsRecord function to determine if the corresponding data record exists in the archive.

Release a connection to a user archive by means of the default function TlgReleaseConnection.

The TlgIsConnected checks to see if there is a connection to "Tag Logging Run Time" and creates a connection if there is none.

In addition to the functions that work in the archive identification, there are two default functions, TlgGetUserFieldDataAsText and TlgSetUserArchivFieldDataAsText, that affect individual fields of the user archive directly by means of the archive name.

Scope

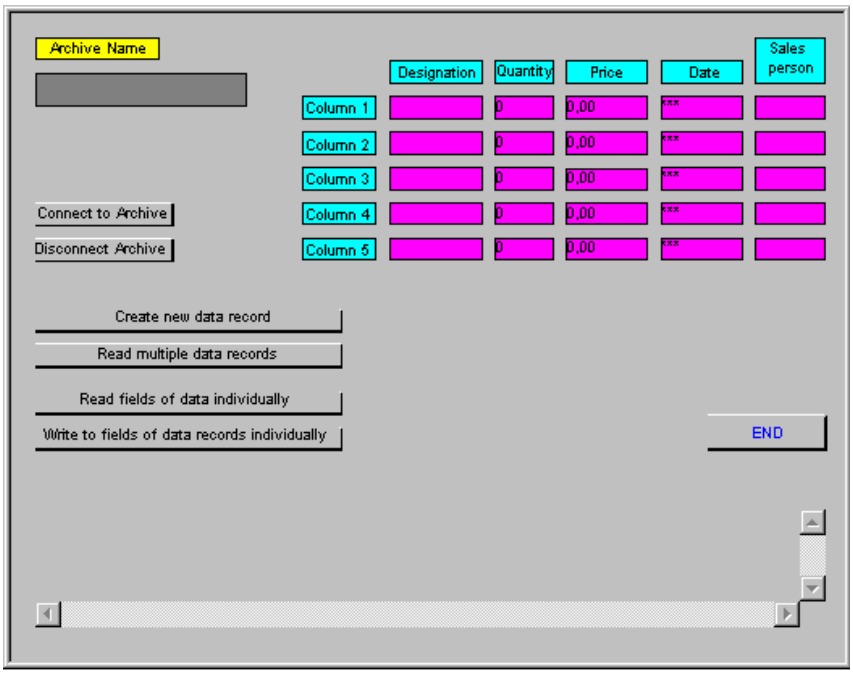
- There is no limit to the number of concurrently open user archives and data records in an archive in WinCC. The number of database fields is limited to 999.
- The communication between PLCs and user archives is limited to one connection per user archive.
- When setting up the communication with the PLC, the name of the user archive may not contain more than eight characters.
- The display of user archives in the table window is limited to 100 columns. It is possible to switch multiple table windows in a user archive.
- Terms that contain special characters as well as words that are reserved by the database are not allowed as field and table names.

4.2 Examples of Default Functions

This section contains examples for the following tasks:

- Creating a new data record
- Reading the fields of a data record individually (in preparation)
- Writing the fields of a data record individually (in preparation)
- Reading multiple data records (in preparation)

In order to trigger the execution of the examples at run time, the graphic objects displayed in the following figure are necessary:



4.2.1 Creating a New Data Record

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
//#################################################
//####       Create new data record            ####
//#################################################
#define MAX_ZE 5                    //Max. Number Lines

DWORD dwConnection ;                //Archiv_ID
char szNameArchiv[ 200 ];           //Array for archive name
char szVarName[200];                //Array for tags of the DM
char szBuffer[ 200 ];               //Buffer of DM values
int i;

   //Get archive name from the I/O field
   strcpy( szNameArchiv,  GetTagChar("Archiv_Name"));

   //Build connection to the archive
   dwConnection = TlgConnectToArchiv(szNameArchiv);

   if( dwConnection == 0xFFFF )     //Connection to archive is OK
   {                                //no
    printf("\r\nError in TlgConnectToArchiv!");
    printf("\r\nValue in error case of Archiv_ID is: %d",
dwConnection);
   }
  else                              //yes
  {
    //write new Archiv_ID in tag
    SetTagDWord("Archiv_ID", dwConnection);

    //Fill record with values
    wsprintf( szVarName, "Bez_%d", 1 );
    strcpy(szBuffer,GetTagChar(szVarName));
    TlgSetRecordFieldByNumber(dwConnection,1,szBuffer);
   //Return type :BOOL

    wsprintf( szVarName, "Me_%d", 1 );
    strcpy(szBuffer,GetTagChar(szVarName));
    TlgSetRecordFieldByNumber(dwConnection,2,szBuffer);
   //Return type :BOOL

    wsprintf( szVarName, "Pr_%d", 1 );
    strcpy(szBuffer,GetTagChar(szVarName));
    TlgSetRecordFieldByNumber(dwConnection,3,szBuffer);
   //Return type :BOOL

    wsprintf( szVarName, "Da_%d", 1 );
    strcpy(szBuffer,GetTagChar(szVarName));
    TlgSetRecordFieldByNumber(dwConnection,4,szBuffer);
   //Return type :BOOL

    wsprintf( szVarName, "Ve_%d", 1 );
    strcpy(szBuffer,GetTagChar(szVarName));
    TlgSetRecordFieldByNumber(dwConnection,5,szBuffer);
   //Return type :BOOL
    if (TlgAddRecord(dwConnection) == 0xFFFF) //Return (-1) if
FALSE, otherwise if TRUE
    {
      printf("\r\nFehler in Fkt. TlgAddRecord ");
    }
    else
    {
      printf("\r\nFkt TlgAddRecord ist O.K. ");
    }//end if

   }//end if
//Release connection with archive
TlgReleaseConnection(dwConnection); //Return type :BOOL
}// end Script
```

4.3 Reference to the Default Functions for User Archives

The default functions provide the user with a simple way to edit user archives.

The following functions are available:

TlgAddRecord	Attaches the temporary data record to the connected archive and returns the number of the data record.
TlgDeleteRecord	Deletes the data record with the number ’dwRowNumber’ in the linked archive ’dwConnection’
TlgConnectToArchiv	Creates a connection to a user archive and returns a connection ID. Also creates a temporary hRecord handle on which all of the Tlg...Record functions work.
TlgDisconnect	If a connection to TLGRT (Tag Logging Runtime) exists, it will be canceled
TlgFirstRecord	Positions the hRecord on the first temporary data record.
TlgGetRecord	Reads the data record with the number ’dwRowNumber’ in the connected archive ’dwConnection’. The field data from the read data record can be manipulated by means of the TlgGetRecordField... or TlgSetRecordField....
TlgGetRecordFieldByNumber	Reads the field addressed with the field number ’dwFieldNumber’ from the temporary data record.

TlgGetUserFieldDataAsText	Reads the field from the user archive ’lpszArchivName’ at the position ’nRow, nColumn’.
TlgIsConnected	Tests whether there is a connection to TLGRT. If there is no connection, a connection is created.
TlgIsRecord	Tests whether the data record with the number ’dwRowNumber’ already exists in the connected archive.
TlgLastRecord	Positions the hRecord on the last temporary data record.
TlgNextRecord	Positions the hRecord on the next temporary data record.
TlgPrevRecord	Positions the hRecord on the previous temporary data record.
TlgReleaseConnection	Releases the connection to the archive ’dwConnection’.
TlgSetRecordFieldByNumber	Writes the text ’lpszText’ to the temporary data record addressed with the field number ’dwFieldNumber’.
TlgSetUserFieldDataAsText	Writes the text from Ipsz text to the field in the user archive ’lpszArchivName’ at the position ’nRow, nColumn’.
TlgUpdateRecord	Overwrites the data record with the number ’dwRowNumber’ in the connected archive ’dwConnection’. The field data of the read data record can be manipulated by means of the TlgGetRecordField... or TlgSetRecordField....

4.3.1 TlgGetUserFieldDataAsText

Description: Reads the field from the user archive 'IpszArchivName' at the position 'nRow, nColumn'.

```
char* TlgGetUserFieldDataAsText (
                    char*                   lpszArchivName,
                    int                     nRow,
                    int                     nColumn )
```

Parameter:

char* lpszArchivName
Name of the user archive

int nRow
Number of the field

int nColumn
Column of the field

4.3.2 TlgSetUserFieldDataAsText

Description: Writes the text from Ipsz Text to the field in the user archive 'IpszArchivName' at the position 'nRow, nColumn'.

```
BOOL TlgSetUserFieldDataAsText (
                    char*                   lpszArchivName,
                    int                     nRow,
                    int                     nColumn,
                    lpsz                    Text )
```

Parameter:

char* lpszArchivName
Name of the user archive

int nRow
Number of the field

int nColumn
Column of the field

lpsz Text
Text to be written

4.3.3 TlgConnectToArchiv

Description: Creates a connection to a user archive and returns a connection ID. Also creates a tempory hRecord handle on which all of the Tlg...Record functions work.

DWORD TlgConnectToArchiv (
char* lpszArchivName)

Parameter: **char* lpszArchivName**
Name of the user archive

Return Value: ID of the archive (dwConnection)

4.3.4 TlgDisconnect

Description: If a connection to TLGRT (Tag Logging Runtime) exists, it will be canceled.

BOOL TlgDisconnect ()

4.3.5 TlgAddRecord

Description: Attaches the temporary data record to the connected archive and returns the number of the data record.

DWORD TlgAddRecord (
DWORD dwConnection)

Parameter: **DWORD dwConnection**
ID of the connected archive

Return Value: Number of the record that was written

4.3.6 TlgDeleteRecord

Description: Deletes the data record with the number ’dwRowNumber’ in the linked archive ’dwConnection’.

DWORD TlgDeleteRecord (
DWORD dwConnection,
DWORD dwRowNumber)

Parameter: **DWORD dwConnection**
ID of the linked archive

DWORD dwRowNumber
Number of the data record to be deleted

4.3.7 TlgUpdateRecord

Description: Overwrites the data record with the number 'dwRowNumber' in the connected archive 'dwConnection'. The field data of the read data recordcan be manipulated by means of the TlgGetRecordField... or TlgSetRecordField....

BOOL TlgUpdateRecord (
DWORD dwConnection,
DWORD dwRowNumber)

Parameter:

DWORD dwConnection
ID of the connected archive

DWORD dwRowNumber
Data record number

4.3.8 TlgGetRecord

Description: Reads the data record with the number 'dwRowNumber' in the connected archive 'dwConnection'. The field data from the read data recordcan be manipulated by means of the TlgGetRecordField... or TlgSetRecordField....

BOOL TlgGetRecord (
DWORD dwConnection,
DWORD dwRowNumber)

Parameter:

DWORD dwConnection
ID of the connected archive

DWORD dwRowNumber
Data record number

4.3.9 TlgIsRecord

Description: Tests whether the data record with the number 'dwRowNumber' already exists in the connected archive.

BOOL TlgIsRecord (
DWORD dwConnection,
DWORD dwRowNumber)

Parameter:

DWORD dwConnection
ID of the connected archive

DWORD dwRowNumber
Number of the data record to be tested

4.3.10 TlgReleaseConnection

Description: Releases the connection to the archive ’dwConnection’.

BOOL TlgReleaseConnection (
DWORD **dwConnection)**

Parameter: **DWORD dwConnection**
ID of the archive to be released

4.3.11 TlgGetRecordFieldByNumber

Description: Reads the field addressed with the field number ’dwFieldNumber’ from the temporary data record.

char* TlgGetRecordFieldByNumber (
DWORD **dwConnection,**
DWORD **dwFieldNumber)**

Parameter: **DWORD dwConnection**
ID of the archive

DWORD dwFieldNumber
Field number of the data record

Return value: Contents of the field as text

4.3.12 TlgSetRecordFieldByNumber

Description: Writes the text ’lpszText’ to the temporary data record addressed with the field number ’dwFieldNumber’.

BOOL TlgSetRecordFieldByNumber (
DWORD **dwConnection,**
DWORD **dwFieldNumber,**
char* **lpszText)**

Parameter: **DWORD dwConnection**
ID of the archive

DWORD dwFieldNumber
Field number of the data record

char* lpszText
Text to be written

4.3.13 TlgIsConnected

Description: Tests whether there is a connection to TLGRT. If there is no connection, a connection is created.

BOOL TlgConnected (void)

Parameter: none

4.3.14 TlgNextRecord

Description: Positions the hRecord on the next temporary data record.

BOOL TlgNextRecord (
DWORD dwConnection)

Parameter: **DWORD dwConnection**
ID of the archive

4.3.15 TlgPrevRecord

Description: Positions the hRecord on the previous temporary data record.

BOOL TlgPrevRecord (
DWORD dwConnection)

Parameter: **DWORD dwConnection**
ID of the archive

4.3.16 TlgFirstRecord

Description: Positions the hRecord on the first temporary data record.

BOOL TlgFirstRecord (
DWORD dwConnection)

Parameter: **DWORD dwConnection**
ID of the archive

4.3.17 TlgLastRecord

Description: Positions the hRecord on the last temporary data record.

BOOL TlgLastRecord (
DWORD dwConnection)

Parameter: **DWORD dwConnection**
ID of the archive

4.4 Reserved Words

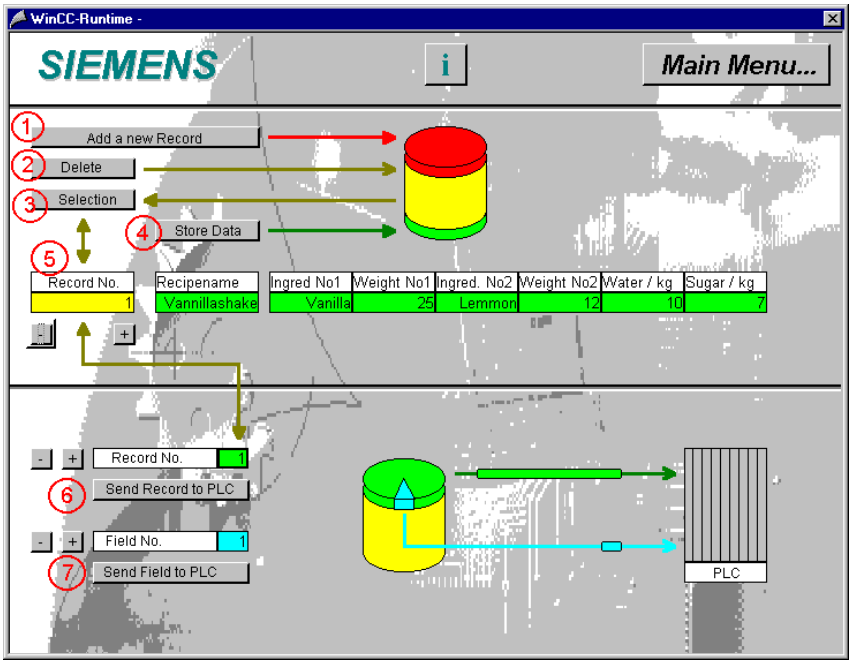
These reserved words are not allowed for field or table names:

add	all	alter	and
any	as	asc	begin
between	binary	break	by
call	cascade	cast	char
char_convert	character	check	checkpoint
close	comment	commit	connect
constraint	continue	convert	create
cross	current	cursor	date
dba	dbspace	deallocate	dec
decimal	declare	default	delete
desc	distinct	do	double
drop	else	elseif	encrypted
end	endif	escape	exception
exec	execute	exists	fetch
first	float	for	foreign
from	full	goto	grant
group	having	holdlock	identified
if	in	index	inner
inout	insert	instead	int
integer	into	is	isolation
join	key	left	like
lock	long	match	membership
message	mode	modify	named
natural	noholdlock	not	null
numeric	of	off	on
open	option	options	or
order	others	out	outer
passthrough	precision	prepare	primary
print	privileges	proc	procedure
raiserror	readtext	real	reference
references	release	remote	rename
resource	restrict	return	revoke
right	rollback	save	savepoint
schedule	select	set	share
smallint	some	sqlcode	sqlstate
start	stop	subtrans	subtransaction
synchronize	syntax_error	table	temporary
then	time	tinyint	to
tran	trigger	truncate	tsequal
union	unique	unknown	update
user	using	validate	values
varbinary	varchar	variable	varying
view	when	where	while
with	work	writetext	

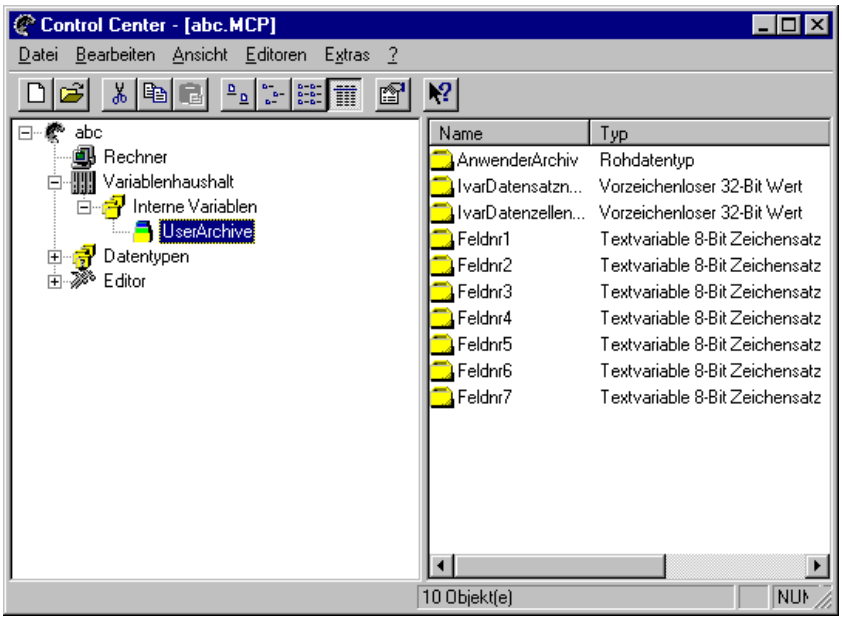
5 API Functions

5.1 Examples of Script Functions

The following sample action scripts show you the different application possibilities for individual creation of process pictures with data from the user archives. These action scripts were also used in the sample pictures (UserARC.PDL and UserARC2.PDL) of the WinCC demonstration project (Demo.MCP).



In order to use the sample action scripts, you must configure the internal tags listed in the following figure. The 8-bit text tags *Field1* through *Field7* must be 20 bytes in length.



Note:

In order for these sample action scripts to function, the function TLGConnect(NULL,NULL) must be executed before each selection of the process picture. When you exit the process picture, you must execute the function TLGDisconnect(NULL).

Samples of the following actions are available:

- Creating a new data record with number
- Getting the fields of a data record individually
- Setting the fields of a data record individually
- Deleting a data record
- Transferring an entire data record to SIMATIC S5
- Transferring one value of a data record to SIMATIC S5

5.1.1 Creating a New Data Record with Number

The sample shows an action script for creating a new data record (with number) in a user archive. In this sample, the script is linked to the event ’mouse click’ on a button (1).

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
   HARCHIVRECORD hRecord = NULL;     // Handle for data record
   char* lpszName = "UserARC";       // Name of the user archive
   DWORD dwFieldNr = 1;              // Database field number
                                     // 1 = Field with the data
record number
                                     // >1 = User data fields
   DWORD dwRecordNr = 0;             // the new data record
number:
                                     // 0 = add after highest data
record number

TLGConnect(NULL,NULL);
   // Select archive
   TLGOpenUserArchiv( lpszName , 0, NULL );

   // Get handle for data record
   TLGNewUserArchivRecord( &hRecord, lpszName,  NULL );

   // Enter the data ‘dwRecordNr’ in the data record ‘hRecord’ in
field ‘dwFieldNr’
   // This way additional data can be entered in the fields
   TLGSetUserArchivFieldData( lpszName,  hRecord, dwFieldNr,
&dwRecordNr, NULL );

   // Write data record to archive
   TLGAddUserArchivRecord( lpszName, hRecord, NULL );

   // Release the handle
   TLGFreeUserArchivRecord( hRecord,  NULL );
TLGDisconnect(NULL);
}
// End of Script
```

5.1.2 Getting the Fields of a Data Record Individually

The sample shows an action script for getting individual data fields from a data record in a user archive. In this sample, the script is linked to the event ’mouse click’ on a button(3). The values that are read are transferred to the internal tags ’Boxnr1’ through ’Boxnr7’.

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
   char Daten[21];                  // Buffer for data field
values
   char Temp[11] ;                  // Buffer for tag names
   char* lpszName = "UserARC";      // Archive name
   DWORD Row, Column;               // Counter tags

TLGConnect(NULL,NULL);

   Row= GetTagDWord("IvarDataRecordNumber");  // Get current data
record number from internal tag

   for (Column = 1; Column <= 7; Column ++)
   {
    sprintf ( Temp, " Boxnr%d", Column);   // Create tag name
    TLGGetUserArchivFieldText( lpszName, Row, Column, Data, 20,
NULL );
    SetTagChar ( Temp , Daten);  // Write data field value to the
                                 // tag
   }
TLGDisconnect(NULL);
}
// End of Script
```

5.1.3 Setting the Fields of a Data Record Individually

The sample shows an action script for setting individual data fields of a data record in a user archive. In this sample, the script is linked to the event ’mouse click’ on a button(4). The values to be set are taken from the internal tags ’Boxnr1’ through ’Boxnr7’.

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
   char* Daten;                     // Pointer for data field
values
   char Temp[21] ;                  // Buffer for tag names
   char* lpszName = "UserARC";      // Archive name
   DWORD Row, Column;               // Counter tags

TLGConnect(NULL,NULL);

   Row= GetTagDWord("IvarDataRecordNumber");  // Get current data
record number from internal tag
   for (Column = 1; Column <= 7; Column ++)
   {
    sprintf ( Temp, "Boxnr%d", Column);
    Data = GetTagChar( Temp );       // Get tag
    TLGSetUserArchivFieldText( lpszName, Row, Column, Data, NULL
);
   }
TLGDisconnect(NULL);
}
// End of Script
```

5.1.4 Deleting a Data Record

The sample shows an action script for deleting a data record in a user archive. In this sample, the script is linked to the event 'mouse click' on a button (2).

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
  HARCHIVRECORD hRecord = NULL;           // Handle for data
record
  char* lpszName = "UserARC";             // Archive name

  DWORD Row;                              // Counter tags

  Row= GetTagDWord("IvarDataRecordNumber");     // get current
                                          // data record number
                                          // from internal
                                          // tag

  TLGOpenUserArchiv (lpszName, 0,NULL);

  // Delete data record in archive
  TLGDeleteUserArchivRecord( lpszName, hRecord, 0, &Row, NULL );

  TLGCloseUserArchiv (lpszName, NULL);
}
// End of Script
```

5.1.5 Transferring an Entire Data Record to SIMATIC S5

The sample shows an action script for transferring an entire data record to SIMATIC S5. The raw data tag configured for the user archive is automatically used for the receipt location. In this sample, the script is linked to the event 'mouse click' on a button(6).

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
   char* lpszName = "UserARC";       // Archive name
   DWORD Row, Column=0;              // Counter tags

   Row= GetTagDWord("IvarDataRecordNumber");  // get current data
record number from internal tag
   TLGTransferUserArchivData( lpszName, 1, &Column, &Row, 1,
NULL, 0, NULL);
}
// End of Script
```

5.1.6 Transferring One Value of a Data Record to SIMATIC S5

The sample shows an action script for transferring a value from a data record to SIMATIC S5. The raw data tag configured for the user archive is automatically used for the receipt location. In this sample, the script is linked to the event ’mouse click’ on a button(7).

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
   char* lpszName = "UserARC";       // Archive name
   DWORD Row, Column;                // Counter tags

   Row= GetTagDWord("IvarDatarecordnumber");  // get current data
record number from internal tag
   Column= GetTagDWord("IvarDataCellNumber");    // get current
data field number from internal tag
   TLGTransferUserArchivData ( lpszName, 2, &Column, &Row, 1,
NULL, 0, NULL);
}
// End of Script
```

5.2 Reference to API Functions

If you want data display of the values in output fields within individual WinCC pictures, there are functions available that can be used in script configuration for access to data records of a user archive. For example, these functions can assign internal tags that then output the values by means of I/O fields and make it possible to input new setpoint values.

Except for a few functions, all functions return a pointer to an error structure. In case of an error, the structure contains detailed entries about the error that occurred.

5.2.1 Error Handling and Error Structure

All API functions return a Boolean result. If the function was executed without errors, the result is TRUE. Otherwise, the result is FALSE.

If multiple errors can occur with an API call, they are transferred to an error structure of the function that initiated the call. The calling application must provide a pointer to an error structure when the function is called.

The error structure contains detailed information about the error that occurred. The structure CMN_ERROR, which is the same in all of the modules of WinCC, is used as the error structure.

The error codes returned by the API functions are listed in the description of the functions.

The expanded error structure contains the error code and an error text for the error that occurred. Each application can use the error structure for evaluation or to output error messages.

```
struct CMNERRORSTRUCT {
        DWORD       dwError1,
        DWORD       dwError2,
        DWORD       dwError3,
        DWORD       dwError4,
        DWORD       dwError5;
        TCHAR       szErrorText[MAX_ERROR_LEN];
        }
```

Parameter:

dwError1 .. dwError5
These entries can be used by the API functions at any time. The API descriptions must then describe which values contain the entries in case of an error.

szErrorText
Buffer for textual description of the error cause. The contents are derived from the resources and are therefore language-dependent.

5.2.2 Overview of the Functions

The functions are divided into the following categories:

Structures	
TLG_ARCHIV_FIELDINFO	
General Functions in Archives	
TLGConnect	
TLGDisconnect	
General Functions in User Archives	
TLGCloseUserArchiv	Close user archive
TLGCreateUserArchiv	Create user archive
TLGDeleteUserArchiv	Delete user archive
TLGOpenUserArchiv	Open user archive
TLGResetUserArchiv	Reset user archive
TLGTransferUserArchivData	Data transfer with user archive
Functions in Data Records of a User Archive	
TLG_ENUMRECORDS_CALLBACK	
TLGAddUserArchivRecord	Write user archive data record
TLGDeleteUserArchivRecord	Delete user archive data record
TLGFreeUserArchivRecord	Delete (temporary) user archive data record
TLGGetNumUserArchivRecords	Read user archive data record number
TLGGetUserArchivRecord	Read user archive data record
TLGGetUserArchivRecords	Get user archive data records
TLGInsertUserArchivRecord	Insert user archive data record
TLGNewUserArchivRecord	Create user archive data record
TLGUpdateUserArchivRecord	Update user archive data record

Functions in Columns of User Archives	
TLG_ENUMFIELD_CALLBACK	
TLGEnumUserArchivFields	List columns of a user archive
TLGGetNumberOfUserArchivFields	Get number of columns in a user archive
TLGGetUserArchivFieldData	Read field of a user archive (temporary data record)
TLGGetUserArchivFieldDataText	Read field (character string) of a user archive (temporary data record)
TLGGetUserArchivFieldText	Read field (character string) of a user archive
TLGGetUserArchivFieldTextByNumber	Get field of a user archive (temporary data record)
TLGSetUserArchivFieldTextByNumber	Write field of a user archive (temporary data record)
TLGSetUserArchivFieldData	Write field of a user archive (temporary data record)
TLGSetUserArchivFieldDataText	Write field (character string) of a user archive (temporary data record)
TLGSetUserArchivFieldText	Write columns of a user archive for text
TLGSetUserArchivFieldTextByName	Write field (character string) of a user archive (temporary data record)

5.2.3 Structures

5.2.3.1 TLG_ARCHIV_FIELDINFO

Brief Description Column information, structure (TLGCS)

```
struct TLG_ARCHIV_FIELDINFO {
                    DWORD                   dwFieldTyp;
                    DWORD                   dwFieldLength;
                    DWORD                   dwFieldDec;
                    DWORD                   dwPassword;
                    TCHAR                   szFieldName[ 128
                    + 1 ];
                    TCHAR                   szVariableName[
                    128 + 1 ];
                    }
```

Members

dwFieldTyp
Data type of the field

dwFieldLength
Field length in bytes

dwFieldDec
Number of decimal points

dwPassword
Password protection for access

szFieldName
Field name

szVariableName
Connection to tags from MCP

Includefile pdertdef.h

The following API functions use this structure:

- TLGCreateUserArchiv
- TLG_ENUMFIELD_CALLBACK

5.2.4 General Functions of Archives

5.2.4.1 TLGConnect

Description: The function initializes the archive system and loads internal memory area for it. The function is necessary in order to execute all of the following functions in action scripts of process pictures.

BOOL TLGConnect (

HWND	**hwndParent,**
PCMN_ERROR	**lpError)**

Parameter:

HWND hwndParent
Handle of the process window: NULL

PCMN_ERROR lpError
Pointer to error data structure: NULL

Return Value

TRUE
Executed without error

FALSE
Error

Includefile: pdertcli.h

Note: You can only log off or release memory areas by using the TLGDisconnect function.

5.2.4.2 TLGDisconnect

Description: Ends the initialization carried out by the TLGConnect function and releases the created internal memory.

BOOL TLGDisconnect (

PCMN_ERROR	**lpError)**

Parameter:

PCMN_ERROR lpError
Pointer to error data structure: NULL

Return Value

TRUE
Executed without error

FALSE
Error

Includefile: pdertcli.h

5.2.5 General Functions on User Archives

5.2.5.1 TLGCloseUserArchiv

Description: Closes an open user archive.

BOOL WINAPI TLGCloseUserArchiv (
LPCTSTR lpszArchivName,
LPCMN_ERROR lpoes);

Parameter:

lpszArchivName
Pointer to the name of the archive

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Archive closed

FALSE
Error

Includefile pdertcli.h

5.2.5.2 TLGCreateUserArchiv

Description: Creates a new user archive.

BOOL WINAPI TLGCreateUserArchiv (

LPCTSTR	**lpszArchivName**
DWORD	**dwNumberOfFields**
PTLG_ARCHIV_FIELDINFO	**lpPUF**
DWORD	**dwNumberOfRecords**
DWORD	**dwFlags**
LPCTSTR	**lpszRawVarName**
PCMN_ERROR	**lpError**

Parameter:

lpszArchivName
Pointer to the name of the archive

dwNumber
Number of fields in archive

lpPuf
Address of the structure TLG_ARCHIV_FIELDINFO with the field information

dwNumberOfRecords
Number of data records

0:	Continuous archive
!0	Short-term archive

dwFlags
Additional specifiers for later expansion

lpszRawVarName
Raw data tag for communication

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Archive created

FALSE
Error

Includefile pdertcli.h

5.2.5.3 TLGDeleteUserArchiv

Description: Deletes a user archive.

TLGDeleteUserArchiv (

LPCTSTR	**lpszArchivName,**
LPCMN_ERROR	**lpoes);**

Parameter:

lpszArchivName
Pointer to the name of the archive

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Archive deleted

FALSE
Error

Includefile pdertcli.h

5.2.5.4 TLGOpenUserArchiv

Description: Opens a user archive.

BOOL WINAPI TLGOpenUserArchiv (

LPCTSTR	**lpszArchivName**
DWORD	**dwFlags ,**
LPCMN_ERROR	**lpoes)**

Parameter:

lpszArchivName
Pointer to the names of the archive

dwFlags
Currently not supported

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Note: This function is necessary if the functions TLGAddUserArchivRecord, TLGFreeUserArchivRecord, TLGGetUserArchivRecord, TLGNewUserArchivRecord, or TLGSetUserArchivFieldData are used in action scripts.

5.2.5.5 TLGResetUserArchiv

Description: Deletes all of the entries of a user archive.

TLGResetUserArchiv (

LPCTSTR	**lpszArchivName**
PCMN_ERROR	**lpError**

Parameter:

lpszArchivName
Pointer to the name of the archive

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
User archive reset

FALSE
Error

Includefile pdertcli.h

5.2.5.6 TLGTransferUserArchivData

Description: The addressed data record or the addressed field of a data record is transferred to the controller by means of the raw data tag assigned to the archive.

```
BOOL WINAPI TLGTransferUserArchivData (
                    LPCTSTR          lpszArchivName,
                    DWORD            dwJob,
                    LPDWORD          lpdwColumn,
                    LPDWORD          lpdwLine,
                    DWORD            dwJobCount,
                    LPCTSTR          lpszSqlString,
                    DWORD            dwFlags,
                    LPCMN_ERROR      lpoes);
```

Parameter:

lpszArchivName
Pointer to the name of the archive

dwJob
Job identifier

1 Write a data record to the controller
2 Write a field of a data record to the controller

lpdwColumn
Pointer to the number of the archive field to which the function relates

lpdwLine
Pointer to the number of the data record to which the function relates

dwJobCount
Job counter

lpszSqlString
Pointer to the SQL command: NULL

dwFlags
NULL

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

5.2.6 Functions in Data Records of a User Archive

5.2.6.1 TLGAddUserArchivRecord

Description: Adds a data record to a user archive. The temporary data record in attached to the user archive.

BOOL WINAPI TLGAddUserArchivRecord (

LPCTSTR	**lpszArchivName,**
HARCHIVRECORD	**hRecord,**
LPCMN_ERROR	**lpError**

Parameter:

lpszArchivName
Pointer to the name of the archive

hRecord
Handle of a data record description

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Data record added

FALSE
Error

Includefile pdertcli.h

Note: The hRecord data record must first have data added by means of the function TLGSetUserArchivFieldData.

5.2.6.2 TLGDeleteUserArchivRecord

Description: Deletes the value of a data record field or the entire data record.

BOOL WINAPI TLGDeleteUserArchivRecord (

LPCTSTR	**lpszArchivName**
HARCHIVRECORD	**hRecord**
DWORD	**dwKeyField**
LPVOID	**lpValue**
LPCMN_ERROR	**lpError)**

Parameter:

lpszArchivName
Pointer to the name of the archive to be edited

hRecord
Handle of the data record to be deleted

dwKeyField
Number of the data record field
- =1 lpValue contains data record number of the database
- >1 Fields of the data record

lpValue
Pointer to the data value. If *dwKeyField* = 1, then *lpValue* must contain the data record number.

lpError
Address of the CMN_ERROR that contains the error information in case of an error.

Return Value

TRUE
Data record deleted

FALSE
Error

Includefile pdertcli.h

5.2.6.3 TLGFreeUserArchivRecord

Description: Deletes a temporary data record and releases the internal memory that was used by the function TLGNewUserArchivRecord for a data record hRecord.

BOOL WINAPI TLGFreeUserArchivRecord (
HARCHIVRECORD hRecord,
LPCMN_ERROR lpoes)

Parameter:

hRecord
Handle of a data record description

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

5.2.6.4 TLGGetNumUserArchivRecords

Description: Returns the number of data records in the user archive.

BOOL WINAPI TLGGetNumUserArchivRecords (
LPCTSTR lpszArchivName,
DWORD* lpdwNumRecord,
LPCMN_ERROR lpoes)

Parameter:

lpszArchivName
Pointer to the name of the archive

lpdwNumRecord
Address of a DWORD in which the number of data records is written.

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Number returned

FALSE
Error

Includefile pdertcli.h

5.2.6.5 TLGGetUserArchivRecord

Description: Reads a data record from a user archive.

BOOL WINAPI TLGGetUserArchivRecord (

LPCTSTR	**lpszArchivName,**
HARCHIVRECORD	**hRecord,**
LPCTSTR	**lpszSqlString,**
LPCMN_ERROR	**lpoes)**

Parameter:

lpszArchivName
Pointer to the name of the archive

hRecord
Handle to a data record description

lpszSqlString
SQL statement

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Data read

FALSE
Error

Includefile pdertcli.h

Note: Give the selection command (SQL command) for selecting a data record as follows, for example:
Selecting the data record with the number 1: "c=1"
Selecting the largest data record: "c > 0 order by c desc"
Selecting the smallest data record: "c > 0 order by c"
The first four bytes of the data record buffer contain the data record number of the type DWORD database. Then the user archive data are attached. The function must be prepared by means of the functions TLGOpenUserArchiv and TLGNewUserArchivRecord.

5.2.6.6 TLGGetUserArchivRecords

Description: Reads the data records of a user archive (TLGRT).

BOOL WINAPI TLGGetUserArchivRecords (

LPCTSTR	**lpszArchivName,**
LPCTSTR	**lpszSqlString,**
TLG_ENUMRECORDS_CALLBACK	**fnCallback,**
DWORD	**dwMaxData,**
LPVOID	**lpUser**
LPCMN_ERROR	**lpoes);**

Parameter:

lpszArchivName
Name of the archive

lpszSqlString
SQL statement

fnCallback
Pointer to your callback function that is called for each data record in the archive

dwMaxData
dwMaxData = 0: All

lpUser
Pointer to application-specific data that is sent to the callback function. This pointer is not evaluated by the function but is made available again in the callback function.

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Related Functions

TLG_ENUMRECORDS_CALLBACK	Read the data records of a user archive, Callback Function

5.2.6.7 TLG_ENUMRECORDS_CALLBACK

Description: In order to evaluate the data records listed by the system, you must prepare a callback function of the TLG_ENUMRECORDS_CALLBACK type.

```
BOOL * TLG_ENUMRECORDS_CALLBACK (
                    LPTSTR                  lpszArchivName,
                    HARCHIVRECORD           hRecord,
                    PVOID                   lpUser);
```

Parameter:

lpszArchivName
Name of the archive

hRecord
Handle to a temporary data record

lpUser
Pointer to application-specific data that is sent to the callback function. This pointer is not evaluated by the function but is made available again in the callback function.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivRecords	Read the data records of a user archive

5.2.6.8 TLGInsertUserArchivRecord

Description: Adds a data record of a user archive (TLGRT).

BOOL WINAPI TLGInsertUserArchivRecord (

LPCTSTR	**lpszArchivName,**
HARCHIVRECORD	**hRecord,**
DWORD	**dwRecordNum,**
LPCMN_ERROR	**lpoes);**

Parameter:

lpszArchivName
Name of the archive

hRecord
Handle to a temporary data record

dwRecordNum
The temporary data record is added to the user archive at the location identified by dwRecordNum.

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

5.2.6.9 TLGNewUserArchivRecord

Description: Creates a temporary data record.

BOOL WINAPI TLGNewUserArchivRecord (

HARCHIVRECORD*	**lphRecord,**
LPCTSTR	**lpszArchivName,**
LPCMN_ERROR	**lpoes)**

Parameter:

lphRecord
Address of a handle to a temporary data record description

lpszArchivName
Pointer to the name of the archive

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Note: This function creates internal memory. This memory is released by means of the function TLGFreeUserArchivRecord.

5.2.6.10 TLGUpdateUserArchivRecord

Description: The specification of the data record to be updated is carried out by means of the value of a field. Through this function, all of the data records that contain the value *IpValue* in the *dwKeyField* column are replaced by the temporary data record. You must ensure that *hRecord* as well as *lpValue* are valid.

BOOL WINAPI TLGUpdateUserArchivRecord (

LPCTSTR	**lpszArchivName,**
DWORD	**dwKeyField,**
PVOID	**lpValue,**
HARCHIVRECORD	**hRecord**
LPCMN_ERROR	**lpoes)**

Parameter:

lpszArchivName
Pointer to the name of the archive

dwKeyField
Number of the column that contains the value to be used to specify the data record to be updated

lpValue
Pointer to the value that is used to specify the data record to be updated

hRecords
Pointer to the temporary data record

lpoes
Pointer to the data of the expanded error message in the structure CMN_ERROR. In case of an error, the system sets the error information in this structure. Do not forget to reserve space for this structure.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

5.2.6.11 Functions in Columns of User Archives

5.2.6.12 TLGEnumUserArchivFields

Description: Numbers the columns of the user archive provided.

BOOL WINAPI TLGEnumUserArchivFields (

LPCTSTR	**lpszArchivName,**
TLG_ENUMFIELD_CALLBACK	**lpfnCallback,**
PVOID	**lpUser,**
PCMN_ERROR	**lpError)**

Parameter:

lpszArchivName
Pointer to the name of the archive

lpfnCallback
Pointer to your callback function that is called for each field in the archive

lpUser
Pointer to application-specific data that is sent to the callback function. This pointer is not evaluated by the function but is made available again in the callback function.

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Archive entries listed

FALSE
Error

Includefile pdertcli.h

Related Functions		
	TLG_ENUMFIELD_CALLBACK	Read the data records of a user archive, Callback Funktion

5.2.6.13 TLG_ENUMFIELD_CALLBACK

Description: In order to evaluate the columns listed by the system, you must prepare a callback function of the TLG_ENUMFIELD_CALLBACK type.

```
BOOL * TLG_ENUMFIELD_CALLBACK (
                PTLG_ARCHIV_FIELDINFO    lpFieldInfo,
                PVOID                    lpUser );
```

Parameter:

lpFieldInfo
Pointer to the structure TLG_ARCHIV_FIELDINFO with the field information

lpUser
Pointer to application-specific data that is sent to the callback function. This pointer is not evaluated by the function but is made available again in the callback function.

Return Value

TRUE
Archive entries listed

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGEnumUserArchivFields	Read the data records of a user archive

5.2.6.14 TLGGetNumberOfUserArchivFields

Description: Provides the number of columns in a user archive.

```
BOOL WINAPI TLGGetNumberOfUserArchivFields (
                    LPCTSTR                 lpszArchivName,
                    DWORD*
                                            lpdwNumberOfFiel
                    ds,
                    PCMN_ERROR              lpError );
```

Parameter:

lpszArchivName
Pointer to the name of the archive

lpdwNumberOfFields
Address of a DWORD in which the number of archive fields is entered

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Number received

FALSE
Error

Includefile pdertcli.h

5.2.6.15 TLGGetUserArchivFieldData

Description: The function reads a field of a temporary data record independent of the field number. A type conversion to the field type of the database is carried out. In contrast to the function TLGGetUserArchivRecordFieldData no field parameters are provided.

BOOL WINAPI TLGGetUserArchivFieldData (

LPCTSTR	**lpszArchivName,**
HARCHIVRECORD	**hRecord,**
DWORD	**dwFieldNumber,**
LPVOID	**lpData,**
DWORD	**dwSizeOfData,**
LPCMN_ERROR	**lpoes)**

Parameter:

lpszArchivName
Pointer to the name of the archive

hRecord
Handle to a temporary data record

dwFieldNumber
Field number of the field in the temporary data record

lpData
Pointer to the buffer in which the value of the field is to be loaded

dwSizeOfData
Size of the data buffer in bytes

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Data read

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivRecordFieldData	Read the field of a user archive (temporary data record)
TLGSetUserArchivFieldData	Write the field of a user archive (temporary data record)

5.2.6.16 TLGGetUserArchivFieldDataText

Description: Use this function to read a field of the temporary data record as a character string. Addressing is carried out by means of the field name (column name).

BOOL WINAPI TLGGetUserArchivFieldDataText (

LPTSTR	**lpszArchivName,**
LPTSTR	**lpszColumnName,**
HARCHIVRECORD	**hRecord,**
LPTSTR	**lpszText,**
DWORD	**dwSizeOfText,**
LPCMN_ERROR	**lpcmnError)**

Parameter:

lpszArchivName
Pointer to the name of the archive

lpszColumnName
Pointer to the field name (column name)

hRecord
Handle to a temporary data record

lpszText
Pointer to a buffer in which the character string is to be loaded

dwSizeOfText
Size of the *szText* buffer.

lpcmnError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Data read

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGSetUserArchivFieldDataText	Write the field (character string) of a user archive (temporary data record)

5.2.6.17 TLGGetUserArchivFieldText

Description: Reads a field from the database table directly as a character string. Addressing of the field is carried out by means of row and column number.

```
BOOL WINAPI TLGGetUserArchivFieldText (
                    LPCTSTR          lpszArchivName,
                    DWORD            dwRow,
                    DWORD            dwColumn,
                    LPCTSTR          lpszText,
                    DWORD            dwMaxTextLength,
                    LPCMN_ERROR      lpError);
```

Parameter:

lpszArchivName
Pointer to the archive name

dwRow
Row number in the user archive

dwColumn
Column number in the user archive

lpszText
Pointer to the buffer in which the character string is to be loaded

dwMaxTextLength
Size of the buffer *szText*

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGSetUserArchivFieldText	Write the column of a user archive (for text)

5.2.6.18 TLGGetUserArchivFieldTextByNumber

Description Reads a field of the temporary data record as a character string. Addressing of the field is carried out by means of the field number (column number).

```
BOOL WINAPI TLGGetUserArchivFieldTextByNumber (
                LPTSTR          lpszArchivName,
                DWORD           dwField,
                HARCHIVRECORD   hRecord,
                LPTSTR          lpszText,
                DWORD           dwSizeOfText,
                LPCMN_ERROR     lpcmnError
                );
```

Parameter:

lpszArchivName
Pointer to the name of the archive

dwField
Number of the field to be read

hRecord
Handle to the temporary data record

lpszText
Pointer to a buffer to accept the contents of the field

dwSizeOfText
Size of the buffer in bytes

lpcmnError
Pointer to the data of the expanded error message in the structure CMN_ERROR. In case of an error, the system sets the error information in this structure. Do not forget to reserve space for this structure.

Return Value

TRUE
The function was executed without errors.

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGSetUserArchivFieldTextByNumber	Write the field of a user archive (temporary data record)

5.2.6.19 TLGSetUserArchivFieldTextByNumber

Description Writes a character string in a field of the temporary data record. Addressing of the field is carried out by means of the field number (column number).

```
BOOL WINAPI TLGSetUserArchivFieldTextByNumber (
                    LPTSTR          lpszArchivName,
                    DWORD           dwField,
                    HARCHIVRECORD   hRecord,
                    LPTSTR          lpszText,
                    LPCMN_ERROR     lpcmnError
                    );
```

Parameter

lpszArchivName
Pointer to the name of the archive

dwField
Number of the field to be changed

hRecord
Handle to a temporary data record

lpszText
Pointer to the character string to be written in the field

lpcmnError
Pointer to the data of the expanded error message in the structure CMN_ERROR. In case of an error, the system sets the error information in this structure. Do not forget to reserve space for this structure.

Return Value

TRUE
Function was executed without errors.

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivFieldTextByNumber	Read the field of a User Archive (temporary data record)

5.2.6.20 TLGSetUserArchivFieldData

Description: Writes a field of a temporary data record depending on the field number. Since only character strings are loadedin the database (and in the temporary data record), an automatic type conversion is carried out at write access.

```
BOOL WINAPI TLGSetUserArchivFieldData (
                LPCTSTR          lpszArchivName,
                HARCHIVRECORD    hRecord,
                DWORD            dwFieldNumber,
                LPVOID           lpData,
                LPCMN_ERROR      lpoes)
```

Parameter:

lpszArchivName
Pointer to the name of the archive

hRecord
Handle to a temporary data record

dwFieldNumber
Field number (>= 1)

lpData
Pointer to a buffer in which the data to be written are loaded

lpoes
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Field set

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivFieldData	Read the field of a user archive (temporary data record)

5.2.6.21 TLGSetUserArchivFieldDataText

Description: Writes a character string in a field of the temporary data record. Addressing of the field is carried out by means of the field name (column name).

BOOL WINAPI TLGSetUserArchivFieldDataText (

LPTSTR	**lpszArchivName,**
LPTSTR	**lpszColumnName,**
HARCHIVRECORD	**hRecord,**
LPTSTR	**lpszText,**
LPCMN_ERROR	**lpcmnError)**

Parameter:

lpszArchivName
Pointer to the name of the archive

lpszColumnName
Pointer to the field name (column name)

hRecord
Handle to a temporary data record

lpszText
Pointer to the character string to be written to the temporary data record

lpcmnError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Data written

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivFieldDataText	Read the field (character string) of a user archive (temporary data record)

5.2.6.22 TLGSetUserArchivFieldText

Description: Writes a character string directly in a field of the database. Addressing of the field is carried out by means of row and column numbers.

```
BOOL WINAPI TLGSetUserArchivFieldText (
                    LPCTSTR          lpszArchivName,
                    DWORD            dwRow,
                    DWORD            dwColumn,
                    LPCTSTR          lpszText,
                    PCMN_ERROR       lpError);
```

Parameter:

lpszArchivName
Pointer to the name of the archive

dwRow
Row number in the user archive

dwColumn
Column number in the user archive

lpszText
Pointer to the character string to be written to the database

lpError
Pointer to the error structure CMN_ERROR. Since this structure is currently not supported, you should enter the value NULL.

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

Related Functions

TLGGetUserArchivFieldText	Read the field (character string) of a user archive

5.2.6.23 TLGSetUserArchivFieldTextByName

Description: Writes a character string in a field of the temporary data record. Addressing is carried out by means of the field name (column name).

BOOL WINAPI TLGSetUserArchivFieldTextByName (
);

Return Value

TRUE
Executed without errors

FALSE
Error

Includefile pdertcli.h

6 Reference to the SIMATIC S5 Message Frame Interface

The data exchange between user archives and SIMATIC S5 programmable logic controllers occurs by means of the S5 transport (layer 4) connection type or the serial connection RK512/3964R.

The data exchange is carried out by means of WinCC raw data tags. These are sent spontaneously by the controller by means of a Read/Write-Send request to WinCC. These message frames contain one or more requirements to the WinCC archive system. These can be write or read requests. As an answer to these requests, WinCC returns the the requested data or a processing acknowledgement.

You can find information about the following subjects:

- Sending requests/data to WinCC
- Sending process acknowledgement/data to SIMATIC S5
- Structure of the message frame headers

6.1.1 Sending Requests/Data to WinCC

Structure of the raw data tag for sending requests and data from the SIMATIC S5 PLC to WinCC:

Message frame header
Request header 1
Possible data of request 1
Possible request header 2
Possible data of request 2
Request n

6.1.2 Sending Process Acknowledgement/Data to SIMATIC S5

Structure of the raw data tag for sending process acknowledgements and data from WinCC to the SIMATIC S5 PLC (this format is also used by WinCC in the TLGTransferUserArchivData function):

Acknowledgement header
Possible data of the acknowledgement

6.1.3 Structure of the Message Frame Headers

Structure of the message frame blocks in detail (divided into bytes):

Message Frame Header:

Function of the Field	Comment
Message frame length in bytes LSB	Length of the field 4 bytes
.	max. length 4091 bytes (due to S5-Transport)
.	.
Message frame length in bytes MSB	.
Transfer type	1 from WinCC, 2 from the controller
Reserved	
Number of requests in message frame LSB	Length of the field 2 bytes
Number of requests in message frame MSB	.
Name of the archive 1st character	The name is provided in ASCII
.	Length of the field 8 bytes
.	.
.	.
.	.
.	.
.	.
Name of the archive 8th character	.

Request Header:

Function of the Field	Comment
Request length in bytes LSB	Length of the field 2 bytes
Request length in bytes MSB	.
Request type	See description
Reserved	
Field number LSB	Length of the field 2 bytes
Field number MSB	.
Data record number LSB	Length of the field 4 bytes
.	.
.	.
Data record number MSB	.
Selection criterion LSB	Field number by which selection is made (not with 0)
Selection criterion MSB	Length of the field 2 bytes

Data of the Request:
The data of the request correspond to the contents of a data record (or to the addressed field).

Note:

Numbers must be transferred in Intel format (LSB first, MSB last). An integer field has a length of 4 bytes; a double field has 8 bytes. Text fields do not end with \0.

The data shift by the length of the field that is chosen as the selection criterion if the selection criterion has a value not equal to 0. If the selection criterion is used, the start of the data range is added as a selection value to the field size of the selection criterion.

Acknowledgement Header:

Function of the Field	Comment
Message frame length in bytes LSB	Length of the field 4 bytes
.	.
.	.
Message frame length in bytes MSB	.
Transfer type	1 from WinCC, 2 from the controller
Reserved	
Error code	See description
Request type	See description
Reserved	
Reserved	
Field number LSB	Length of the field 2 bytes
Field number MSB	.
Data record number LSB	Length of the field 4 bytes
.	.
.	.
Data record number MSB	.
Name of the archive 1st character	The name is provided in ASCII
.	Length of the field 8 bytes
.	.
.	.
.	.
.	.
.	.
Name of the archive 8th character	.

Data of the Acknowledgement:
The acknowledgement contains either the data record or the addressed field (with a read request) or it is empty (write request, archive request).

Description of the Request Types:

Type	Description
4	Test archive for availability
5	Delete all data records from the archive
6	Read (get) data record
7	Write (set) data record
8	Delete data record
9	Read (get) data record field
10	Write (set) data record field

Description of the Error Codes:

Group	Nr.	Description
General	0	Function was executed
Archive	1	Invalid data
Archive	2	Data not available
Data record	101	Invalid data
Data record	102	Data not available
Field	201	Invalid data
Field	202	Data not available
General	254	Function not available
General	255	Undefined error

Index

S

T