TELETEXT NETWORK API

LAST MODIFICATION DATE: 2018-11-16

Part 1

F.A. BERNHARDT GMBH

Teletext & Subtitling Products Group

Teletext Network API

TELETEXT & SUBTITLING PRODUCTS GROUP

Teletext Network API

Melkstattweg 27 • 83646 Bad Tölz, Germany
Telephone +49 8041 76890 • Fax +49 8041 768932
E-Mail: sales@fab-online.com
http://www.fab-online.com

Table of contents

Chapter 1

Introduction

Main features of the FAB Teletext Data Generator API

This document describes the API that allows accessing the FAB Teletext Data Generator over network, serial port or modem/ISDN from 3rd party applications that are running under Windows.

MAIN FEATURES

Information

Main features of the FAB Teletext Data Generator API:

- Complete functionality available over COM in all programming languages.
- Allows accessing the Teletext Data Generator over network, serial port, modem and ISDN.
- Allows using network functionality without the need to learn details about the network.
- Multiple network sessions are supported.
- Each session allows working with a teletext page that may contain many subpages..
- Read from and write to page operations are supported.
- Supports loading and saving of many teletext file formats.
- As an alternative a DLL is available that allows accessing all functions from any programming language that allows calling DLL functions including C, C++, Delphi, Word Basic, VBA, ...

License Agreement

1. The subject of this license agreement is the following:
    a. Source code provided within this document
    b. TCP/IP protocol information provided within this document
    c. Software provided as part of this product (including DLL files)
2. The above source code, TCP/IP protocol information and software (DLLs) will be called "The product" in the following text.
3. Use of "The product" is allowed on any computer that is owned by the same company that bought this product. The computers must be within 250m of radius.
4. For every group of computers owned by the same company that are within 250m of radius another license must be obtained.
5. Distribution of "The product" outside of the company that bought "The product" is strictly prohibited.
6. Products built by using parts of "The product", for example applications using the TCP/IP protocol described in this document or applications using the DLL or applications using the source code provided as part of "The product" can be distributed to other companies and individuals provided that the other company or individual also owns the license for "The product". If the other company or individual does not own the license for "The product" then the company that provided the application must buy an additional license for "The product" specifying the end user before its end user can start using the applications built on "The product".

If you do not accept this license agreement then return the product to your supplier within 30 days from receipt.

Chapter 2

Description of the API

Description of the API

General Overview

GENERAL OVERVIEW

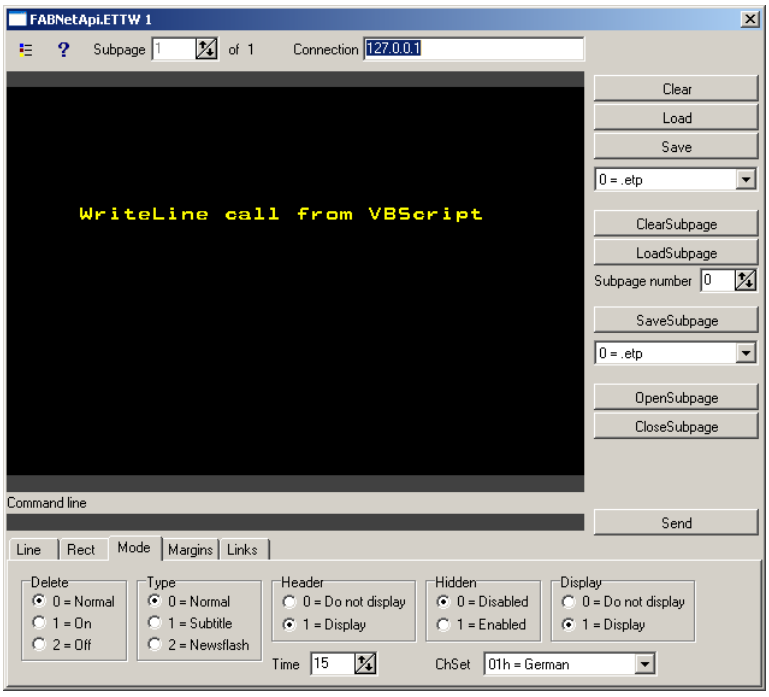
Information

The COM interface FABNetApi.ETTW is available in all programming languages that support interfacing to COM objects. A very simple example of a VBScript program shows how to use the COM object:

```
dim MyApi
set MyApi = CreateObject("FABNetApi.ETTW")
MyApi.WriteLine 4, 10, "WriteLine call from VBScript"
call MyApi.DebugWindow(10,10)
```

The above script creates an object of the type FABNetApi.ETTW, writes a line of text onto the teletext page and shows a Debug Window which can be of help during the development. You can run this script by saving it into a file, i.e. test.vbs and start it with:

```
cscript test.vbs
```



Basics

BASICS

Information

The Debug Window shown above enables you to check out most of the functions that are available in the Netapi. Every button corresponds to a call of a function/method of the FABNetApi.ETTW object.

By creating the FABNetApi.ETTW object an empty Teletext page is created where you can write text to and read text from the page. You can also load and save the page from/to a file.

Teletext Control Characters

TELETEXT CONTROL CHARACTERS

Information

Teletext control characters have codes below 32 according to the table below:

0	Alpha Black	16	Graphics Black
1	Alpha Red	17	Graphics Red
2	Alpha Green	18	Graphics Green
3	Alpha Yellow	19	Graphics Yellow
4	Alpha Blue	20	Graphics Blue
5	Alpha Magenta	21	Graphics Magenta
6	Alpha Cyan	22	Graphics Cyan
7	Alpha White	23	Graphics White
8	Flash	24	Conceal
9	Steady	25	Contiguous Graphics
10	End Box	26	Separated Graphics
11	Start Box	27	Twist / Escape
12	Normal Height	28	Black Background
13	Double Height	29	New Background
14	Double Width	30	Hold Graphics
15	Double Size	31	Release Graphics

Using FAB Network API in Common Programming Languages

The following examples show how to use the FAB Network API in different programming languages.

VBScript

```
dim MyApi
set MyApi = CreateObject("FABNetApi.ETTW")
MyApi.WriteLine 4, 10, "WriteLine call "
call MyApi.DebugWindow(10,10)
```

Python

```
(Install python and pypiwin32: pip install -U pypiwin32)

import win32com.client
o = win32com.client.Dispatch("FABNetApi.ETTW")
o.WriteLine (4, 10, "WriteLine call")
o.DebugWindow(10,10)
```

Perl

```
use strict;
use warnings;
use Win32::OLE;
use Win32;

my $netapi = CreateObject Win32::OLE ("FABNetApi.ETTW") ||
die ("CreateObject:$!");
$netapi->WriteLine(4, 10, "WriteLine call");
$netapi->DebugWindowShow(10,10);
```

Powershell

```
$FAB = New-Object -ComObject FABNetApi.ETTW
$FAB.WriteLine(4,8," WriteLine call")
$FAB.DebugWindow(10,10)
```

Chapter 3

API Functions / Methods

The functions / methods available in the API

The Interface

INTERFACE
Information

To be able to use "early binding" in a compiler like Delphi you can import the file FABNetApi.tlb which is installed by the installation program. The compiler will generate a header file which will allow the compiler to check for the correct syntax of the function call already during compile time.

Basic Methods and Properties

BASIC METHODS
Information

Property Version: BSTR (Read Only)

The property Version returns the version of the file FABNetApi.dll as a Wide string

Procedure DebugWindow (x, y: LONG)

The procedure DebugWindow shows the debug window and stops the execution of the program. The window is shown at coordinates (x,y). You can now interactively use the window to execute the functions. Simply close the window to continue with the execution of the program.

Procedure DebugWindowShow (x, y: LONG)

The procedure DebugWindowShow shows the debug window but DOES NOT stop the execution of the program. The window is shown at coordinates (x,y). You can use the window to see the effects of the functions that are called by the program.

Procedure DebugWindowHide

The procedure DebugWindowHide hides the debug window.

Property Connection: BSTR

The property Connection contains the IP address or the network name of the Teletext Data Generator where commands and the Teletext page will be sent to when calling the Send method.

Property Command: BSTR (Read Only)

The property Command contains the command line which is returned by the Teletext Data Generator after a command has been sent by calling the function Send.

Function Send (Command: BSTR): LONG

The function SendCommand expects as a parameter a string which contains the command line which is sent together with the Teletext page to the Teletext Data Generator. The return value is <0 if a network error occurs (i.e. no Data Generator is available under the IP address specified in the property Connection). The return value is >= 0 if the command was sent to the Data Generator. If the Data Generator returns an error (the reply is in red colour) then the return value is 0. If the Data Generator has executed the command correctly then the return value is 1.

Function SendFiles (Files: BSTR): BOOL

The function SendFiles expects as a parameter a string which contains the filenames (with path) of files which are sent to the Teletext Data Generator. The return value is TRUE if the file transfer was OK and FALSE is a network error occurred. This function can be used to transfer the file CLOCK.CLK to the data generator.

Function ModemOnline: BOOL

The function ModemOnly returns TRUE if a modem connection is used and a modem connection is still established.

Function HangupModem: BOOL

The function HangupModem interrupts a modem connection if a modem is used.

Function LoadHiLevel (Filename: BSTR): BOOL

The function LoadHiLevel loads the file HILEVEL.L25 with Teletext Level 2.5 layouts from the filename that is specified as the parameter.

Page and Subpage Methods and Properties

Procedure Clear

The procedure Clear closes all subpages of the Teletext page so that only one subpage is open and clears the contents of the subpage.

Function Load (Filename: BSTR): BOOL

The function Load loads a Teletext page from a file.

Function Save (Filename: BSTR; Format: LONG): BOOL

The function Save saves the Teletext page into a file. The parameter Format defines the format of the file according to the following table. Please note that only the ETP file format supports multiple subpages in one file. All other formats only store one (the active) subpage.

0 = ETP Files (*.ETP) RECOMMENDED
1 = ETT Files (*.TTP)
2 = VTX Files
3 = EP1 Files
4 = Edixia Files
6 = TPG Files
7 = DTA Files
8 = TTI Files
9 = PAG Files
10 = WSP Files
11 = TRATEC Files
12 = BIN Files
13 = BMP Files
14 = JPG Files
15 = TSS Files
16 = PGW Files
17 = ORT Files

Property SubpageCount: LONG (Read Only)

The property SubpageCount contains the number of subpages of the Teletext page.

Property Subpage: LONG

The property Subpage contains the number of the currently active subpage, starting with 1. Assign a value to this property to change the active subpage. The value must be between 1 and SubpageCount.

Procedure ClearSubpage

The procedure ClearSubpage clears the contents of the currently active subpage.

Function LoadSubpage (Filename: String; index: LONG): BOOL

The function LoadSubpage loads only the subpage “index” from the file specified in the parameter Filename. Example: If the file test.etp contains 10 subpages and index is 5 then the fifth subpage of the file test.etp is loaded into the current subpage.

Function SaveSubpage (Filename: String; Format: LONG): BOOL

The function SaveSubpage saves only the current subpage into the file specified in the parameter Filename. The parameter Format defines the file format that will be used (see the function Save).

Procedure OpenSubpage

The procedure OpenSubpage opens a new subpage immediately after the current subpage.

Function CloseSubpage: BOOL

The function CloseSubpage closes the currently active subpage. This function only works if more than one subpage is open and returns TRUE in such case.

Function ReadPage: BSTR

The function ReadPage returns the currently open teletext page including all subpages as a string. The result can be used in the function WritePage to copy a teletext page with all subpages to a different instance of the COM object.

Function ReadSubpage: BSTR

The function ReadSubpage returns the currently active subpage of tzhe teletext page as a string. The result can be used in the function WriteSubpage to copy a subpage of the teletext page to a different instance of the COM object.

Procedure WritePage (text: BSTR)

The procedure WritePage writes a string with a teletext page including all subpages which was obtained by the function ReadPage to the currently open teletext page.

Procedure WriteSubpage (text: BSTR)

The procedure WriteSubpage writes a string with one subpage of a teletext page which was obtained by the function ReadSubpage to the currently open subpage of the teletext page.

Text Methods

Procedure WriteLine (x, y: LONG; text: BSTR)

The procedure WriteLine writes the text in the parameter text onto the Teletext page at the coordinates (x,y). Please note that the text must be in Unicode and it will be converted automatically into the correct teletext character set. Make sure that the property SubpageModeChSet is set to the correct character set.

Teletext control characters can be specified as codes below 32 or as tokens from the following list.

Token	Description	Token	Description
{Black}	Text colour black	{GBlack}	Graphics colour black
{Red}	Text colour red	{GRed}	Graphics colour red
{Green}	Text colour green	{GGreen}	Graphics colour green
{Yellow}	Text colour yellow	{GYellow}	Graphics colour yellow
{Blue}	Text colour blue	{GBlue}	Graphics colour blue
{Magenta}	Text colour magenta	{GMagenta}	Graphics colour magenta
{Cyan}	Text colour cyan	{GCyan}	Graphics colour cyan
{White}	Text colour white	{GWhite}	Graphics colour white
{Flash}	Flash on	{Conceal}	Conceal text
{Steady}	Flash off	{CG}	Contiguous graphics
{EB}	End box	{SG}	Separated graphics
{SB}	Start box	{ESC}	Escape
{NH}	Normal height	{BB}	Black Background
{DH}	Double height	{NB}	New background
{DW}	Double width (Level 2.5)	{Hold}	Hold graphics
{DS}	Double size (Level 2.5)	{Release}	Release graphics

Procedure WriteRect (ax, ay, dx, dy: LONG; text: BSTR)

The procedure WriteRect writes the text from the parameter text into the rectangle specified by the coordinates (ax, ay) and (ax+dx-1, ay+dy-1). The parameters dx and dy specify the with and height of the rectangle that should be written. The text will be written into a new line every dx characters. You can use the function TTXRectangulize to format the text into the correct colour and justify it. Please note that the Teletext page consists of 40 characters per row and 24 rows so that the dx and dy parameters will be adjusted accordingly. The parameters ax must be between 0 and 39 and the parameter ay must be between 1 and 24. The parameter dx must be between 1 and 40. The parameter dy must be between 1 and 24.

Function TTXRectangulize (intext: BSTR; width, color, just: LONG; out linecount: LONG): BSTR

The procedure TTXRectangulize returns a string which is ready formatted for use in the procedure WriteRect. The text is broken into more lines where a new line occurs every “width” characters. Wordwrap is used and spaces are used to fill up the line. The parameter width must be between 1 and 40. The parameter color must be between 0 and 7 (Teletext colors). The parameter just specifies the justification: 0 = Left, 1=Center, 2=Right, 3=Block. The parameter linecount will contain the number of lines in the result string when the function returns.

Function TTXRectangulizeEx (intext: BSTR; width, color, just: LONG; out linecount: LONG): BSTR

The procedure TTXRectangulizeEx works exactly the same as TTXRectangulize. The only difference is that leading spaces are not removed from the intext parameter.

Function TTXRectangulizeText (intext: BSTR; width, color, just: LONG): BSTR

The procedure TTXRectangulizeText returns a string which is ready formatted for use in the procedure WriteRect. The text is broken into more lines where a new line occurs every “width” characters. Wordwrap is used and spaces are used to fill up the line. The parameter width must be between 1 and 40. The parameter color must be between 0 and 7 (Teletext colors). The parameter just specifies the justification: 0 = Left, 1=Center, 2=Right, 3=Block. The difference to the function TTXRectangulize is that the linecount will not be returned and, if necessary, has to be checked manually by calling TTXRectangulizeLC.

Function TTXRectangulizeTextEx (intext: BSTR; width, color, just: LONG): BSTR

The procedure TTXRectangulizeTextEx works exactly the same as TTXRectangulizeText. The only difference is that leading spaces are not removed from the intext parameter.

Function TTXRectangulizeLC (intext: BSTR; width, color, just: LONG): LONG

The procedure TTXRectangulizeLC returns the number of lines that would be occupied by a string which is returned by TTXRectangulizeText with the same parameters.

Function TTXRectangulizeLCEx (intext: BSTR; width, color, just: LONG): LONG

The procedure TTXRectangulizeLCEx works exactly the same as TTXRectangulizeLC. The only difference is that leading spaces are not removed from the intext parameter.

Function ReadLine (x, y, len: LONG): BSTR

The procedure ReadLine returns “len” characters, starting at the coordinates (x,y). The function stops at the end of the line and can therefore never return more than 40 characters.

Function ReadRect (ax, ay, dx, dy: LONG): BSTR

The procedure ReadRect returns the characters from the rectangular area defined by the rectangle (ax, ay) and (ax+dx-1, ay+dy-1). The parameters dx and dy specify the with and height of the rectangle that should be read. Please note that the Teletext page consists of 40 characters per row and 24 rows so that the dx and dy parameters will be adjusted accordingly. The parameters ax must be between 0 and 39 and the parameter ay must be between 1 and 24. The parameter dx must be between 1 and 40. The parameter dy must be between 1 and 24.

Page Parameters Methods and Properties

Property SubpageMarginLeft: LONG

This property defines the left margin that is used in the teletext editor.

Property SubpageMarginRight: LONG

This property defines the right margin that is used in the teletext editor.

Property SubpageMarginTop: LONG

This property defines the top margin that is used in the teletext editor.

Property SubpageMarginBottom: LONG

This property defines the bottom margin that is used in the teletext editor.

Property SubpageModeDelete: LONG

This property defines the mode for deletion of the teletext subpage. The value 0 (default) means that the subpage is transmitted with the C4 (clear) bit only when a new subpage is transmitted for the first time. The value 1 means that the subpage is always transmitted with C4 and 2 means that the page will never be transmitted with C4 (clear) bit.

Property SubpageModeType: LONG

This property defines the type of the teletext subpage (bits C5 and C6). The value 0 (default) means that it is a normal teletext page. 1 means it is a subtitle page and 2 means that it is a newsflash page. When using a subtitle or newsflash page the page is transparent. To write the text into a line make sure to use two consecutive Start Box control characters at the beginning of the line, otherwise the text will not be visible on the TV set.

Property SubpageModeHeader: LONG

This property defines the value of the header control bit C7. The value 1 (default) means that the header line will be displayed. The value 0 means that the header line will not be displayed.

Property SubpageModeHidden: LONG

This property defines the value of the header control bit C9. The value 0 (default) means that the page number will be visible when the Teletext decoder is showing the counter while searching for a page. The value 1 means that the page number will never be visible when the decoder is searching (waiting) to receive a page.

Property SubpageModeDisplay: LONG

This property defines the value of the header control bit C10. The value 1 (default) means that the content of the teletext page will be visible when displayed by the teletext decoder. The value 0 means that a black page will be displayed.

Property SubpageModeTime: LONG

This property defines the display time of the subpage in seconds. The value must be between 3 and 63.

Property SubpageModeChSet: LONG

This property defines the value of the header control bits C12, C13 and C14. The value must be one from the following table:

00h = English
01h = German
02h = Swedish
03h = Italian
04h = French
05h = Spanish
06h = Czech / Slovak
20h = Polish
22h = Estonian
23h = Lithuanian
25h = National Option 5
26h = Turkish
27h = Rumanian
34h = Serbian
37h = Greek
44h = Russian

Property SubpageLink1: BSTR

This property defines the page number that is used for the red FASTEXT button (i.e. 101 or any other page number, use an empty string to disable the link). This page number is transmitted in packet X/27. To define the visible line 24 of the teletext page use the procedure WriteLine (0, 24, text).

Property SubpageLink2: BSTR

This property defines the page number that is used for the green FASTEXT button (i.e. 102 or any other page number, use an empty string to disable the link). This page number is transmitted in packet X/27. To define the visible line 24 of the teletext page use the procedure WriteLine (0, 24, text).

Property SubpageLink3: BSTR

This property defines the page number that is used for the yellow FASTEXT button (i.e. 103 or any other page number, use an empty string to disable the link). This page number is transmitted in packet X/27. To define the visible line 24 of the teletext page use the procedure WriteLine (0, 24, text).

Property SubpageLink4: BSTR

This property defines the page number that is used for the cyan FASTEXT button (i.e. 104 or any other page number, use an empty string to disable the link). This page number is transmitted in packet X/27. To define the visible line 24 of the teletext page use the procedure WriteLine (0, 24, text).

Property SubpageLinkIndex: BSTR

This property defines the page number that is used for the index FASTEXT button (i.e. 105 or any other page number, use an empty string to disable the link). This page number is transmitted in packet X/27. To define the visible line 24 of the teletext page use the procedure WriteLine (0, 24, text).

Function ArchiveSearch(server: BSTR; maxhits: LONG; criteria: BSTR): LONG;

The function ArchiveSearch connects to FAB Archive Server using the IP address or network name specified in the server parameter. It uses the search criteria specified in the criteria parameter and it returns a maximum of maxhits pages where maxhits can not be more than 50. The result is smaller than 0 if an error occurred during the search. If the result is >= 0 then it specifies the number of found pages. Use the function ArchiveResult and ArchiveRead to read the data of the found pages.

The parameter criteria must contain the search criteria in the following format. It contains one or more strings delimited with ; which specify which pages to search for:

PAGE (page number)
SUBPG (subpage number)
CHANNEL (channel name)
ONAIR (onair time)
TEXT (page contents)
Length of each item is limited to 255 characters.

"condition" for PAGE and SUBPG keys must be in form <interval>{,<interval>}, where <interval> is either <number> or <number>-<number> (meaning interval range).

"condition" for CHANNEL must be in form <name>{,<name>}

"condition" for ONAIR key must be in form <start_time>[<end_time>]. Time must be in the yyyy-mm-dd hh:nn:ss format

"condition" for TEXT can be any text ('+' prefix means AND, '-' prefix means NOT, no prefix means OR).

Example (all data must be written in a single line without CR/LF):

```
PAGE=110,113-114;SUBPG=2;CHANNEL=ORF1,ORF2;
ONAIR=2007-02-23 12:20:00 2007-02-26 13:07:54;
TEXT=+Berlin +Wien
```

Function ArchiveResult(id: LONG): BSTR;

The function ArchiveResult returns the data about the found page. The parameter id must be >= 0 and smaller than the number of results returned by ArchiveSearch. The result is in the same format as the query.

Example of the result:

```
PAGE=110;SUBPG=1;ONAIR=2007-02-23 12:20:00
```

Function ArchiveRead(id: LONG): BOOL;

The function ArchiveRead reads the page from the Archive Server into the current teletext page. The parameter id must be >= 0 and smaller than the number of results returned by ArchiveSearch. The result is FALSE if an error occurred when reading the page from the Archive Server.

Chapter

4

Description of the ETTWAC32.DLL

Description of the ETTWAC32.DLL that allows using the API in an old fashion way

General Overview

GENERAL OVERVIEW

Information

For compatibility with older applications FAB provides a DLL (ETTWAC32.DLL) which contains functions that can be called from user programs. FAB recommends that new applications use the COM object FABNetApi.dll as described before in this document.

The DLL ETTWAC32.DLL that is supplied is also available as part of the FAB Teletext Editor for Windows FT-ETTWIN 2.5. However this API will allow you to use most of the functions that are available also without the need to have the Teletext Editor installed on the same PC.

A demonstration program called ETT for Windows Control Panel 16/32 bit is provided as a complete application (ETTWCP16.EXE and ETTWCP32.EXE) as part of the FAB Teletext Editor for Windows. If you do not have the Teletext Editor then simply download the Demo version of the Teletext Editor from http://www.fab-online.com. The Source code for the demonstration program can be found in the SOURCE subdirectory. The application can be compiled by Borland Delphi 2 and 3.

The function prototypes that you can use can be found in the file SOURCE\ETTWACIM.PAS. If you wish to use the DLL by any other language but Pascal or Delphi then you will most probably have to translate the file to your language (i.e. in C/C++ you will need a .H header file). Note that almost all parameters are long (32 bit signed) parameters.

The files SOURCE\WinWord6.txt contains a demonstration macro for WinWord 6 and the file SOURCE\WinWord7.txt contains a demonstration macro for WinWord 7. Included are all function prototypes. You can also use the same files with Visual Basic.

Make sure to copy the file ETTWAC32.DLL into the directory WINDOWS\SYSTEM32. Only then you will be able to call the DLL functions from your application.

DLL Basics

The ETT for Windows Control Panel enables you to check out most of the functions that are available in the DLL. So first start ETT for Windows Control Panel.

To be able to access the DLL functions you first have to open a session in the DLL. This is done by executing the function ETTW_OpenSession. The functions returns a SESSION number which must be used as a parameter to all following commands to identify the correct session.

To close the session simply execute the function ETTW_CloseSession.

Your program should be calling the function ETTW_IDLE for each session at least once per second to make sure that the DLL can perform functions that are required to process network requests in the background. The DLL function ETTW_IDLE is very quick and will return with almost no delay.

Note that you are allowed to open more than one session from your program. This is very useful if your application is using more than one thread. Please note that you should only call one DLL function at a time and if you will be using more threads than an external synchronisation of threads is required. The best way is to open a separate session for each thread.

Teletext Control Characters

TELETEXT CONTROL CHARACTERS

Information

Teletext control characters have codes below 32 according to the table below:

0	Alpha Black	16	Graphics Black
1	Alpha Red	17	Graphics Red
2	Alpha Green	18	Graphics Green
3	Alpha Yellow	19	Graphics Yellow
4	Alpha Blue	20	Graphics Blue
5	Alpha Magenta	21	Graphics Magenta
6	Alpha Cyan	22	Graphics Cyan
7	Alpha White	23	Graphics White
8	Flash	24	Conceal
9	Steady	25	Contiguous Graphics
10	End Box	26	Separated Graphics
11	Start Box	27	Twist / Escape
12	Normal Height	28	Black Background
13	Double Height	29	New Background
14	Double Width	30	Hold Graphics
15	Double Size	31	Release Graphics

DLL Functions

DLL FUNCTIONS

Information

Remember that all functions return a Longint value. It is >0 when the function has executed successfully. It is <=0 if the DLL function could not be executed.

Only the function ETTW_OPENSESSION is an exception to this rule. It will return 0 if a session could not be opened. Otherwise it will return a session number which is not equal to 0.

There may be more commands available in the function prototype file ETTWACIM.PAS that describes the interface to the DLL functions than in this manual. The additional functions are either not implemented or have not been tested yet and should therefore not be used.

Chapter 5

API Functions

The functions available in the API

Session Commands

SESSION COMMANDS
Information

ETTW_OPENSESSION

function ETTW_OPENSESSION: Longint;

This function opens a new session. This function must be called before any other function can be used. The return value is the Session number which should be used as the parameter for all following functions. The session opens a teletext page that will allow writing to the page, reading from the page, transmitting the page to the Teletext Data Generator over network and more.

NOTE: The function has failed if the return value is 0. This is not according to the common convention, that the function succeeded if the return value is > 0.

ETTW_CLOSESESSION

function ETTW_CLOSESESSION (Session: Longint): Longint;

This function closes a session which was previously opened by the function ETTW_OPENSESSION. You should call this function when your application is terminated. The parameter Session is the same as returned by the function ETTW_OPENSESSION.

ETTW_DEBUGWINDOW

function ETTW_DEBUGWINDOW (Session, X, Y: Longint): Longint;

This function displays a window which displays the current content of the teletext page buffer of the current session. This function can be called on any position in the source code of the application to enable debugging and displays the current teletext page. The function shall never be called in the production application because it will block the execution of the program until the window is closed manually.

Communication Commands

ETTW_SENDCOMMAND

function ETTW_SENDCOMMAND (Session: Longint; Command: PChar): Longint;

This function transmits a command to the teletext system. The Command parameter must be an array of 40 bytes in the active Windows codepage. If you need to specify colour characters then use the character codes below 32.

NOTE: You must call ETTW_SETCONNECTION first to define to which Teletext Data Generator on the network the command should be sent to.

ETTW_READCOMMAND

function ETTW_READCOMMAND (Session: Longint; Cmd: PChar): Longint;

This function reads the text from the command line in the teletext page of the active session. The parameter Cmd must be pointing to an array of 40 bytes. The characters returned will be in the active Windows codepage and also control characters with codes below 32 will be returned if present in the command line.

To be able to find out quickly whether the last command was executed successfully you should call ETTW_ISFABERROR. This function only works when communicating to the FAB Teletext System.

ETTW_ISFABERROR

function ETTW_ISFABERROR (Session: Longint): Longint;

This function returns >0 if the Teletext System returned an ERROR when the last command was executed, otherwise it returns 0 when there was no error or <0 when the function can not be executed.

For example if you send the command GET 12345 to the FAB Teletext System then ETTW_ISFABERROR will return >0 because the command GET 12345 is invalid.

This function only works when communicating to the FAB Teletext System.

ETTW_SETCONNECTION

function ETTW_SETCONNECTION (Session: Longint;
ConnectionName: PChar): Longint;

This function will define the network name of the Data Generator to which all commands within this session should be sent to.. The network name must be specified in ConnectionName. ConnectionName must be a zero terminated string (i.e. G1 or NC1, the name should NOT be preceded by FAB+).

ETTW_GETCONNECTION

function ETTW_GETCONNECTION (Session: Longint;
ConnectionName: PChar; MaxLen: Longint): Longint;

This function will return the name of the current connection of the active session in the DLL. The parameter ConnectionName must point to an array of characters of the length which is specified in MaxLen.

Text Commands

ETTW_WRITETOPAGELINE

function ETTW_WRITETOPAGELINE (Session: Longint; AText: PChar; ALen, AX, AY: Longint): Longint;

This function writes the text which is specified in the parameter AText (an array of characters of the length ALen) on the position AX (0-39) and AY (0-23). If the text is longer then what would fit into the specified row it will be truncated.

The Text must be in the current Windows codepage. Control characters can be specified as codes below 32.

ETTW_WRITETOPAGERECT

function ETTW_WRITETOPAGERECT (Session: Longint; AText: PChar; ALen, AX, AY: Longint): Longint;

This function writes the text which is specified in the parameter AText (an array of characters of the length ALen) on the position AX (0-39) and AY (0-23). If the text is longer then what would fit into the specified row then the text will continue in the next row at the position AX.

The Text must be in the current Windows codepage. Control characters can be specified as codes below 32.

ETTW_READFROMPAGELINE

function ETTW_READFROMPAGELINE (Session: Longint; AText: PChar; MaxLen: Longint; AX, AY: Longint): Longint;

This function will read the text from the teletext page at the position AX (0-39) and AY (0-23). The parameter AText must point to an array of characters (bytes) of the length MaxLen. The return value is >0 when successful and it is the number of characters returned. It is <0 if the function could not be executed.

The text will be returned in the active Windows codepage. Control characters will be returned with codes below 32.

ETTW_READFROMPAGERECT

function ETTW_READFROMPAGERECT (Session: Longint; AText: PChar; MaxLen: Longint; AX, AY: Longint): Longint;

This function will read the text from the teletext page at the position AX (0-39) and AY (0-23). The parameter AText must point to an array of characters (bytes) of the length MaxLen. If MaxLen is longer then what would fit into the specified start row AY then the text will continue in the next row at the position AX. The return value is >0 when successful and it is the number of characters returned. It is <0 if the function could not be executed.

The text will be returned in the active Windows codepage. Control characters will be returned with codes below 32.

ETTW_WRITETOPAGERECTANGLE

function ETTW_WRITETOPAGERECT (Session: Longint; AText: PChar; ALen, AX, AY, DX, DY: Longint): Longint;

This function writes the text which is specified in the parameter AText (an array of characters of the length ALen) on the position AX (0-39) and AY (0-23). A maximum of DX characters will be written into each row and a maximum of DY rows will be created. If the text is longer then what would fit into the specified start row AX then the text will continue in the next row at the position AX.

The Text must be in the current Windows codepage. Control characters can be specified as codes below 32.

ETTW_READFROMPAGERECTANGLE

function ETTW_READFROMPAGERECTANGLE (Session: Longint; AText: PChar; MaxLen: Longint; AX, AY, DX, DY: Longint): Longint;

This function will read the text from the teletext page at the position AX (0-39) and AY (0-23). A maximum of DX characters will be read from one row and a maximum of DY rows will be read. The parameter AText must point to an array of characters (bytes) of the length MaxLen. If MaxLen is longer then what would fit into the specified start row AY then the text will continue in the next row at the position AX. The return value is >0 when successful and it is the number of characters returned. It is <0 if the function could not be executed.

The text will be returned in the active Windows codepage. Control characters will be returned with codes below 32.

ETTW_WRITETOPAGELINKS

function ETTW_WRITETOPAGELINKS (Session: Longint; Link1, Link2, Link3, Link4, IndexLink: PChar): Longint;

This function will write the FASTEXT links into row 27 of the teletext page. The parameters Link1, Link2, Link3, Link4 and IndexLink must either be NIL or a pointer to an array of characters that holds at least 4 bytes.

The function requires a null terminated string for each link:

NIL or empty string if the link should be disabled
The page number as a null terminated string (i.e. 100, 200, …).
Subpages are not allowed.

To disable all links all parameters should be NIL. By disabling links the packets X/24 and X/27 will not be transmitted.

To write the FASTEXT text into row 24 of the teletext page use one of the ETTW_WRITETOPAGE functions with the Y parameter set to 24.

ETTW_READFROMPAGELINKS

function ETTW_READFROMPAGELINKS (Session: Longint; Link1, Link2, Link3, Link4, IndexLink: PChar): Longint;

This function will read the FASTEXT links from the teletext page. The parameters Link1, Link2, Link3, Link4 and IndexLink must either be NIL or a pointer to an array of characters that can hold at least 4 bytes.

The function will return the following null terminated string for each link:

Empty string if the link is disabled
The page number as a null terminated string (i.e. 100, 200, …)

If all links are disabled the function will return 0

To read the FASTEXT text from row 24 of the teletext page use one of the ETTW_READFROMPAGE functions with the Y parameter set to 24.

ETTW_STOREBLOCK

function ETTW_STOREBLOCK (Session: Longint; ABlock: Pointer; ABlockSize: Longint): Longint;

This function is required for storing a buffer of data that is later transmitted as data broadcast data in packets 30 and 31.

The function requires following data:

ABlock
Is a pointer to a data block in memory which is copied into the internal buffer of the FAB Network API

ABlockSize
Is the size of the data block that should be copied. The size of the data block must be a multiple of 40 and it may not be larger than 1800 bytes. This means that following data block sizes are allowed: 40, 80, 120, …, 1760, 1800. If a block size does not follow these rules the function will return an error.

The function ETTW_STOREBLOCK requires the data size to be a multiple of 40 bytes. The reason is that every 40 bytes are transmitted within one VBI line and the Teletext Data Generator always copies 40 bytes from the buffer and transmits the block of 40 bytes in one VBI line.

See the section “Data Broadcast” later in this manual to read about what else is required to be able to transmit data broadcast.

Note as well that this function is not implemented in FAB Teletext Editor for Windows API and if you will try to emulate it by using the FAB Teletext Editor it will not work correctly. This function is only implemented in the FAB Teletext Network API and only works correctly when the FAB Teletext Editor for Windows is bypassed.

Page Commands

ETTW_CLEARSUBPAGE

function ETTW_CLEARSUBPAGE (Session: Longint): Longint;

This function will clear the active subpage in the active session. Please note that more than one subpage can be open in each session by using the command ETTW_OPENSUBPAGE.

ETTW_LOADSUBPAGE

function ETTW_LOADSUBPAGE (Session: Longint; FileName: PChar; SubpageNumber: Longint): Longint;

This function will load the file specified as a zero terminated string in the parameter FileName into the active session of the DLL. The SubpageNumber should be 0 to load the first subpage of the file.

ETTW_SAVESUBPAGEAS

function ETTW_SAVESUBPAGEAS (Session: Longint; FileName: PChar; FileType: Longint): Longint;

This function will save the current subpage in the active session into a file on the disk. The parameter FileName must be a zero terminated string. The parameter FileType defines the file format in which the file will be stored:

0 = ETP Files (*.ETP) RECOMMENDED
1 = ETT Files (*.TTP)
2 = VTX Files
3 = EP1 Files
4 = Edixia Files
6 = TPG Files
7 = DTA Files
8 = TTI Files
9 = PAG Files
10 = WSP Files
11 = TRATEC Files
12 = BIN Files
13 = BMP Files
14 = JPG Files
15 = TSS Files
16 = PGW Files
17 = ORT Files

ETTW_OPENSUBPAGE

function ETTW_OPENSUBPAGE (Session: Longint): Longint;

This function opens a new subpage in the active session of the DLL and makes the new subpage active.

ETTW_CLOSESUBPAGE

function ETTW_CLOSESUBPAGE (Session: Longint): Longint;

This function closes the active subpage in the active session of the DLL. Please note that this function will only work if the number of subpages in the window is higher than 1.

ETTW_SETCURSUBPAGE

function ETTW_SETCURSUBPAGE (Session: Longint; Subpage: Longint): Longint;

This function makes the subpage (starting with 0) specified in the parameter Subpage active.

ETTW_GETCURSUBPAGE

function ETTW_GETCURSUBPAGE (Session: Longint): Longint;

This function returns the number of the current subpage (starting with 0) in the active session of the DLL.

ETTW_GETSUBPAGECOUNT

function ETTW_GETSUBPAGECOUNT (Session: Longint): Longint;

This function returns the number of subpages in the window in the active session of the DLL.

ETTW_GETPARAMETER

function ETTW_GETPARAMETER (Session, Option: Longint; Value: PLongint): Longint;

This function returns the value of the selected page parameter of the teletext page in the window in the active session of the DLL. The page parameter is selected in the Option parameter of the function. The page parameter is returned in the Value parameter of the function.

Values of the Option and Value parameter:

Option	Description	Value
0	Page Delete C4	0 = Normal (Default) 1 = On 2 = Off
1	Page Type C5, C6	0 = Normal (Default) 1 = Subtitle 2 = Newsflash
2	Page Header C7	0 = Do not display 1 = Display (Default)
3	Page Hidden C9	0 = Disabled (Default) 1 = Enabled
4	Page Display C10	0 = Do not display 1 = Display (Default)
5	Page Time	3 to 63 (seconds)
6	Character Set C12,C13, C14	00h = English 01h = German 02h = Swedish 03h = Italian 04h = French 05h = Spanish 06h = Czech / Slovak 20h = Polish 22h = Estonian 23h = Lithuanian 25h = National Option 5 26h = Turkish 27h = Rumanian 34h = Serbian 37h = Greek 44h = Russian

ETTW_SETPARAMETER

function ETTW_SETPARAMETER (Session, Option, Value: Longint): Longint;

This function sets the value of the selected page parameter of the teletext page in the window in the active session of the DLL. The page parameter is selected in the Option parameter of the function. The page parameter value is set in the Value parameter of the function.

Values of the Option and Value parameter:

Option	Description	Value
0	Page Delete C4	0 = Normal (Default) 1 = On 2 = Off
1	Page Type C5, C6	0 = Normal (Default) 1 = Subtitle 2 = Newsflash
2	Page Header C7	0 = Do not display 1 = Display (Default)
3	Page Hidden C9	0 = Disabled (Default) 1 = Enabled
4	Page Display C10	0 = Do not display 1 = Display (Default)
5	Page Time	3 to 63 (seconds)
6	Character Set C12,C13, C14	00h = English 01h = German 02h = Swedish 03h = Italian 04h = French 05h = Spanish 06h = Czech / Slovak 20h = Polish 22h = Estonian 23h = Lithuanian 25h = National Option 5 26h = Turkish 27h = Rumanian 34h = Serbian 37h = Greek 44h = Russian

Chapter 6

Examples

Examples of how to use the API

File Conversion and Client/Server Protocol

FILE CONVERSION
Information

Converting teletext files to ETP Format

FAB Teletext programs use the ETP file format as the main file format for storing teletext information. Files in ETP format are text files with binary information stored within the text file encoded in a similar way as email messages are encoded with MIME encoding.

It is very easy to write a program that will convert teletext files from other formats into ETP format by using the API. The following example shows how to do it.

```
function ConvertFile (InFile, OutFile: String): Boolean;
begin
  Result := false;
  Session := ETTW_OPENSESSION;
  i := 1;
  while ETTW_LOADSUBPAGE (Session, InFile+Str (i)+'.ep1') > 0 do
  begin
    ETTW_OPENSUBPAGE (Session);
    inc (i);
  end;
  if ETTW_SUBPAGECOUNT (Session) > 1 then begin
    ETTW_CLOSESUBPAGE (Session);
    Result := ETTW_SAVEPAGE (Session, OutFile) > 0;
  end;
  ETTW_CLOSESESSION (Session);
end;

ConvertFile (C:\Files\Page100_', 'C:\Files\Page100');
```

The above program will scan for files Page100_1.EP1, Page100_2.EP1 and more subpages and store all subpages into a single ETP file.

Transferring Teletext Pages to FAB Teletext Data Generator or FAB Teletext Network Controller

To transfer teletext pages to FAB Teletext Data Generator or to transfer them to FAB Teletext Network Controller the client/server protocol has to be used which is supported by these systems. The API supports transferring commands and pages. It is however important to use commands that are supported by the Data Generator and Network Controller. These commands are described in detail in the manual of the Data Generator and Network Controller.

The following example shows how to transfer pages over network:

```
procedure TransferPgs (PageDir: String; FromPg, ToPg: Integer);
var
  i: Integer;
begin
  Session := ETTW_OPENSESSION;
  ETTW_SETCONNECTION (Session, 'DGNetName');
  ETTW_SENDCOMMAND (Session, 'REG username,password');
  for i := FromPg to ToPg do
  begin
    j := 0;
    if ETTW_LOADPAGE (Session, PageDir + Str (i)+'.etp') > 0 do
    begin
      while j < ETTW_SUBPAGECOUNT (Session) do
      begin
        ETTW_SETCURSUBPAGE (Session, j);
        ETTW_SENDCOMMAND (Session, 'PUT ' + Str (i) + '.' + Str (j));
        inc (j);
      end;
    end;
    ETTW_SENDCOMMAND (Session, 'TRU ' + Str (i) + '.' + Str (j));
  end;
  ETTW_CLOSESESSION (Session);
end;

TransferPgs ('C:\Files\', 100, 199);
```

The above program transfers all pages 100.etp, 101.etp, … in directory C:\Files to the Data Generator over network. The commands PUT and TRU are used to store and truncate the pages on the data generator.

"PUT ppp.s" is used to store a subpage. "TRU ppp.s" is used to cut off all subpages higher than s.

Transferring Teletext Pages to FAB Teletext Data Generator for Interactive Teletext

To transfer teletext pages to FAB Teletext Data Generator the client/server protocol has to be used which is supported by the data generator. The API supports transferring commands and pages. It is however important to use commands that are supported and designed for interactive teletext.

The following example shows how to transfer pages interactively (very quickly) over network:

```
function TransmitPg (MaskPage, Text: String; Pgno: Integer): Boolean;
var
  i: Integer;
begin
  Session := ETTW_OPENSESSION;
  ETTW_SETCONNECTION (Session, 'DGNetName');
  ETTW_SENDCOMMAND (Session, 'REG username,password');
  ETTW_WRITETOPAGELINE (Session, 2, 5, PChar (Text));
  Result := ETTW_SENDCOMMAND (Session, 'SIP ' + Str (i)) > 0;
end;

TransmitPg ('C:\Files\mask.etp', 'Hello', 777);
```

The above program loads the file C:\Files\mask.etp, writes "Hello" onto the page and transmits it interactively (without delay) as page 777.

The command "SIP ppp" is used to transmit the page interactively.
To transmit the page without the clear bit use "SIP pppN"

Transmitting Teletext Subtitles

To transmit teletext subtitles to FAB Teletext Data Generator the client/server protocol has to be used which is supported by the data generator. The API supports transferring commands and pages. It is however important to use commands that are supported and designed for interactive teletext / subtitles.

The following example shows how to transfer pages interactively (very quickly) over network:

```
function TransmitPg (MaskPage, Text: String; Pgno: Integer): Boolean;
var
  i: Integer;
begin
  Session := ETTW_OPENSESSION;
  ETTW_SETCONNECTION (Session, 'DGNetName');
  ETTW_SENDCOMMAND (Session, 'REG username,password');
  ETTW_WRITETOPAGELINE (Session, 0, 23,
    PChar (chr (StartBox) + chr (StartBox) + Text + chr (EndBox)));
  Result := ETTW_SENDCOMMAND (Session, 'SIP ' + Str (i)) > 0;
end;

TransmitPg ('C:\Files\mask.etp', 'Hello', 777);
```

The above program loads the file C:\Files\mask.etp, writes "Hello" onto the page and transmits it interactively (without delay) as page 777. Make sure that the page "mask.etp" has the subtitle bit set. You can set this bit by setting it in the FAB Teletext Editor in Edit / Page parameters.

The command "SIP ppp" is used to transmit the page interactively.
To transmit the page without the clear bit use "SIP pppN"

The StartBox character has code 11 (decimal) and the EndBox character has the code 10 (decimal). Two StartBox characters are required to produce a subtitle, otherwise the text is not visible because the box for the subtitle will not appear. The EndBox character terminates the box for the subtitle.

Transmitting Data Broadcast

Data broadcast is a special application which uses the VBI to transmit any data encapsulated into teletext packets.

The FAB Network API and FAB Teletext Data Generator can be used to transmit data broadcast. The Network API allows storing any kind of data into a buffer and transmitting the buffer of network/modem/ISDN/serial line to the FAB Teletext Data Generator.

The FAB Teletext Data Generator will transmit the buffered data in packets 30 or 31 in the VBI according to the teletext specification.

It is important to understand that the Network API will only store the data that is provided into packets 30 and 31. The program itself must take care about forming the structure of the Packets 30/31 so that the decoders are able to decode the data correctly.

The following program shows how to transmit data in packet 1/31.

```
function ReadData (data: Pointer; var size: Longint): Boolean;
begin
  // user function that fills in the data to be sent
  // also returns the number of bytes that are filled into the buffer
  // the block size must be a multiple of 40 but not more than 1800 bytes
  // returns false when there is no more data to be sent
end;

var
  data: Pointer;
  size: Longint;
begin
  Session := ETTW_OPENSESSION;
  ETTW_SETCONNECTION (Session, 'DGNetName');
  ETTW_SENDCOMMAND (Session, 'REG username,password');
  GetMem (data, 1800); // allocates 1800 bytes for the buffer
  while ReadBlock (data, size) do
  begin
    ETTW_STOREBLOCK (Session, data, size);
    ETTW_SENDCOMMAND (Session, 'I31 1');
  end;
  FreeMem (data);
  ETTW_CLOSESESSION (Session);
end;
```

Note that after using the function ETTW_STOREBLOCK the function ETTW_SENDCOMMAND with the parameter “I31 1” is used. This means that the data block will be transmitted in packet 1/31. The following commands should be used to transmit the data block in other packets:

Packet	Command	Packet	Command
1/30	I30 1	1/31	I31 1
2/30	I30 2	2/31	I31 2
3/30	I30 3	3/31	I31 3
4/30	I30 4	4/31	I31 4
5/30	I30 5	5/31	I31 5
6/30	I30 6	6/31	I31 6
7/30	I30 7	7/31	I31 7
8/30	I30 8	8/31	I31 8

Any I3X command can also be sent in the form: I30 1 1. In such case the second “1” specifies the data stream that will be used (1-4).

You have noticed that the function ETTW_STOREBLOCK requires the data size to be a multiple of 40 bytes. The reason is that every 40 bytes are transmitted within one VBI line and the Teletext Data Generator always copies 40 bytes from the buffer and transmits the block of 40 bytes in one VBI line.

Using Spare Capacity to Transmit Data Broadcast

Teletext data generator has to transmit teletext in a way that all teletext decoders can decode the transmitted teletext pages. There are many teletext decoders that require a 20ms pause after transmission of the header line (row 0) and the other rows. This means that after the row 0 no more data can be transmitted even if there are free lines in the VBI available.

FAB Teletext Data Generator with software version 4 or higher supports transmission of packets 30 and 31 in this spare capacity. Software versions lower than 4 will not use spare capacity and therefore data broadcast will always slow down the normal Teletext.

There are three configuration parameters for data broadcast on the Data Generator:

- Maximum number of VBI lines per two seconds that can be used from the spare capacity. Setting this parameter to any value (like 2000) will never slow down the transmission of the normal teletext.
- Guaranteed number of VBI lines per 2 seconds that will be allocated to this stream even if no spare capacity is available. Setting this parameter to a value higher than 0 may slow down the transmission of the normal Teletext if not enough spare capacity is available but only if data is really transmitted.
- Maximum number of VBI lines per field that can be used from the spare capacity. Setting this parameter to any value (like 20) will never slow down the transmission of the normal teletext.

The parameters can be changed automatically with a time command which means that for example during night time the transmission parameters can be changed so that more VBI lines will be used for data broadcast and teletext will be slowed down. During daytime the parameters can be changed again so that only spare capacity will be used and teletext will not be slowed down.

The client application uses the ETTW_SENDCOMMAND function to transmit data. The function may return an error (this must be checked by calling ETTW_ISFABERROR) if the transmission buffer for data broadcast is full. In such case the client application must repeat the command after a few moments when the transmission buffer may have some empty space again.

The following program shows how to transmit data in packet 1/31.

```
function ReadData (data: Pointer; var size: Longint): Boolean;
begin
  // user function that fills in the data to be sent
  // also returns the number of bytes that are filled into the buffer
  // the block size must be a multiple of 40 but not more than 1800 bytes
  // returns false when there is no more data to be sent
end;

var
  data: Pointer;
  size: Longint;
begin
  Session := ETTW_OPENSESSION;
  ETTW_SETCONNECTION (Session, 'DGNetName');
  ETTW_SENDCOMMAND (Session, 'REG username,password');
  GetMem (data, 1800); // allocates 1800 bytes for the buffer
  while ReadBlock (data, size) do
  begin
    ETTW_STOREBLOCK (Session, data, size);
    repeat
      ETTW_SENDCOMMAND (Session, 'I31 1');
    until not ETTW_ISFABERROR (Session);
  end;
  FreeMem (data);
  ETTW_CLOSESESSION (Session);
end;
```

Please note the usage of the function ETTW_ISFABERROR which will check if the ETTW_SENDCOMMAND function was successful. ETTW_ISFABERROR will return true if the data generator reported an error saying that the transmission buffer for data broadcast is currently full and can not accept more data.

The above program is a demonstration program only. In addition please note that all ETTW_ functions return a value which has to be checked to see if the function was successful at all.

Chapter 7

TCP/IP Protocol

Description of the TCP/IP Protocol for communication with FAB Teletext Data Generator and FAB Network Controller

Client/Server TCP/IP Socket Protocol

PROTOCOL OVERVIEW

Information

Protocol Overview

PLEASE NOTE THAT SOME DATA IS NOT DISCLOSED IN THIS DOCUMENT. THE COMPLETE DESCRIPTION OF THE PROTOCOL IS ONLY AVAILABLE WHEN YOU BUY THE FT-NETAPI FAB TELETEXT NETWORK API PRODUCT.

FAB Teletext Data Generator and Network Controller (called "Server" in the following text) accept TCP/IP socket connections on TCP port XXXX.

The REQUEST packet that is sent to the server must contain a command and a teletext page. The server executes the command and returns a REPLY packet that contains a teletext page and a confirmation/error message for the executed command. The server can accept multiple connections at the same time.

The server acts in the following way:

1. Listen on TCP port XXXX.
2. Accept an incoming connection.
3. Wait to receive the REQUEST packet. If the complete packet is not received within 30 seconds close the connection and goto 1.
4. Execute the command in the REQUEST packet.
5. Send the REPLY packet.
6. Wait to receive another incoming REQUEST packet. If no data is received within one second close the connection and goto 1. If the complete packet is not received within 30 seconds close the connection and goto 1.
7. When complete REQUEST packet is received goto 4.

The client must work in the following way:

1. Open TCP socket connection to server IP, port XXXX.
2. Prepare request packet.
3. Send request packet.
4. Wait to receive reply packet.
5. If there are more request packets to be sent goto 2.
6. Close connection.

REQUEST PACKET

Information

Request Packet Structure

The client sends the following REQUEST packet to the server. The REQUEST packet contains binary data. PSIZE is the size of the complete REQUEST packet.

Byte offset	No. of bytes	Name	Std. value	Description
0	4	VER	1	Transport version
4	4	BSIZE	PSIZE-8	Length of data that follow in this packet (in bytes)
8	4	CMD	0	Command
12	4	TAG	2	Command Tag
16	4	TSIZE	BSIZE-12	Length of data that follow in this Tag
20	16	SESS		Session ID, must be filled with binary 0 on first request. The following requests should contain the ID from the REPLY packet
36	4	RES1	0	Reserved, must be 0
40	4	RES2	0	Reserved, must be 0
44	TSIZE-24	DATA		Teletext command and page buffer

The DATA structure is described later on.

The length of the REQUEST packet is:

44 bytes in the header preceding the DATA structure
76 bytes in the DATA structure
n*41 optional bytes (LINES) in the DATA structure (0 <= n <= 47)

The minimum length of the REQUEST packet is 120 bytes.
The maximum length of the REQUEST packet is 2088 bytes.

The SESS field contains the session ID. The first time the session is established the REQUEST packet must contain an empty SESS packet (filled with binary 0). The REPLY packet will contain the session ID which must be used in all following REQUEST packets.

The session ID is valid only for a limited time which can be defined on the Teletext Data Generator with the command “CFG SESSTOUT x” where x is the number of minutes for the validity of a session.

If the REQUEST packet contains a session ID which is invalid (has expired or has never been assigned) then the REPLY packet will return “E98-INV.SESSION” in the cmd field of the DATA.

Reply Packet Structure

The server sends the following REPLY packet to the client after receiving the REQUEST packet and executing the command from the REQUEST packet. The REPLY packet contains binary data. PSIZE is the size of the complete REPLY packet.

Byte offset	No. of bytes	Name	Std. value	Description
0	4	VER	1	Transport version
4	4	CMD	1	Command
8	4	BSIZE	PSIZE-12	Length of data that follow in this packet (in bytes)
12	4	TAG	2	Command Tag
16	4	TSIZE	BSIZE-8	Length of data that follow in this Tag
20	16	SESS		Session ID, returned by server, should be used in the following request packets
36	4	RES1	0	Reserved, must be 0
40	4	RES2	0	Reserved, must be 0
44	TSIZE-24	DATA		Teletext command and page buffer

The DATA structure is described later on.

The length of the REPLY packet is:

44 bytes in the header preceding the DATA structure
76 bytes in the DATA structure
n*41 optional bytes (LINES) in the DATA structure (0 <= n <= 47)

The minimum length of the REPLY packet is 120 bytes.
The maximum length of the REPLY packet is 2088 bytes.

DATA Structure

The data structure contains the following data (the data is byte aligned, it is NOT word aligned):

```
struct TcpipRecord
{
   char          undisclosedata[35];
   char          cmd[40];
   char          len;
   char          lines[48*41];
};
```

The size of the types used in the structure is:

```
char    8 bit
int     16 bit
long    32 bit
```

The meaning of the fields in the structure is the following:

undisclosed data
The contents of this data is described in the documentation that you receive when buying the FT-NETAPI product.

cmd
REQUEST: contains 40 bytes of teletext command. The command is a string (not zero terminated, simply 40 characters, add spaces (0x20) at the end). All characters must be from 7-bit ASCII codepage (only characters with codes 0 to 127)
REPLY: same as request, but the confirmation/error of the executed command is returned. The reply may have odd parity which means that the highest bit value must be stripped off.

len
REQUEST: number of text lines of the teletext page that follow. This value can be from 0 to 48. Each text line contains 41 bytes.
REPLY: same as request

lines
REQUEST: every line contains 41 bytes. The number of lines of 41 bytes is defined by the “len” field. The first of the 41 bytes is the line number (valid values: 0 to 28), the following 40 bytes contain the text. The value of the “codepage” field defines the codepage of the characters. If the codepage is 0 then no conversion is done and the characters will be transmitted exactly as present in the packet. If the codepage is not 0 then all characters will be converted from the ANSI codepage (1250-1257) into the teletext codepage. Generally you should always use the ANSI codepage because conversion into the teletext codepage can be very difficult. Codepage 0 should only be used for data broadcast applications that will transmit binary data. Note that if you use control characters for graphic teletext colours the conversion to teletext codepage will not be done where the graphic colour is active. You should use the teletext codepage for graphic characters for those characters.
REPLY: same as request