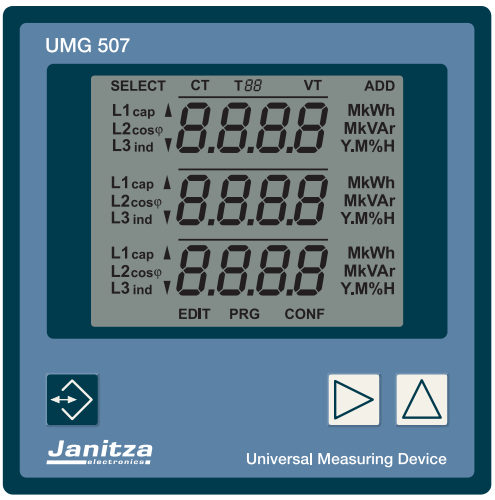
Universal Measuring Device

UMG507

Functional Description

Webserver



Dok. Nr. 1.032.309.0

Janitza electronics GmbH
Vor dem Polstück 1
35633 Lahnau
Support Tel. (06441) 9642-22
email: info@janitza.de
internet: http://www.janitza.de

Generals

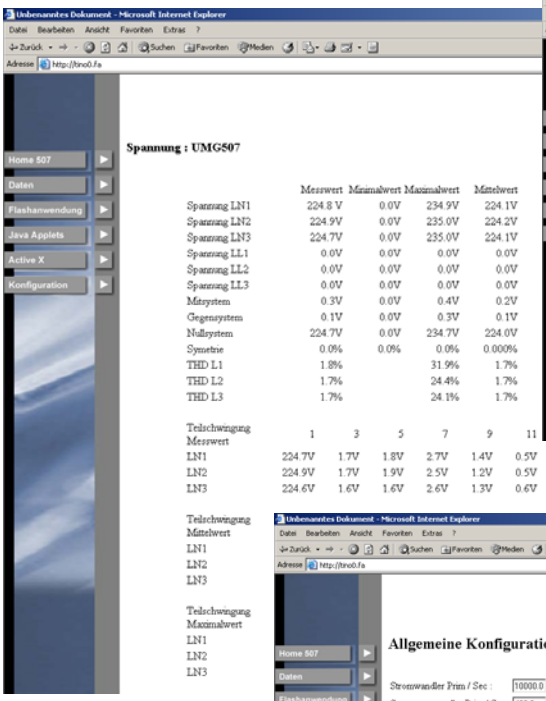
The UMG507 supports the protocols Modbus RTU, Modbus TCP/IP, Modbus over TCP/IP (Modbus Gateway) or Profibus DP V0, depending in the version. This functional description is an addition to the manual and describes the configuration of the corresponding function step by step.
More functional descriptions can be found on the CD-ROM PSWbasic/professional. At present, the following functional descriptions are available:

- UMG507 used as remote data display for external Modbus slaves
- OPC Server Port 502
- OPC Server Port 8000 (Modbus Gateway Function)
- The webserver of UMG507
- Description of the storage of UMG507
- Description of Profibus with examples

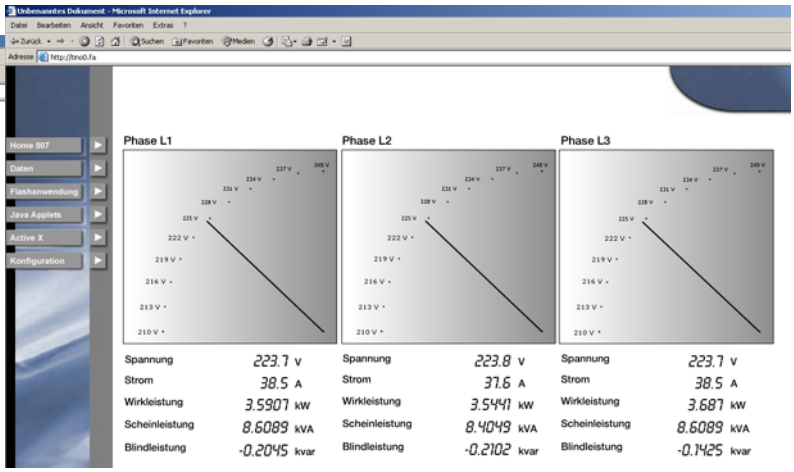
Issue Note:	
18.11.2004	First edition / Wagner
22.11.2004	Correction

The Webserver of UMG507E

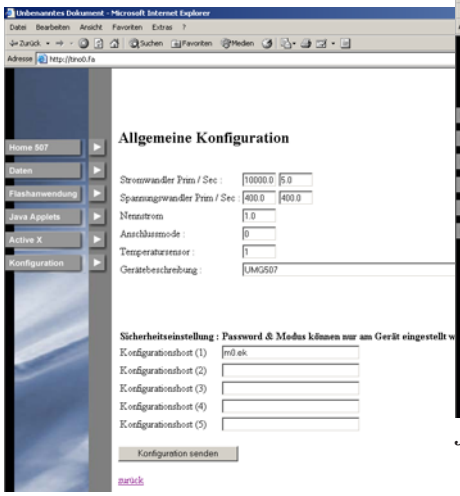
The use is able to deposit at HTML pages, JAVA applets, Active X components and Macromedia Flash Files onto the internal of UMG507E/EP at will. Via intranet or internet, the pages can be called up by an internet browser. The complete programming of the instrument can also be carried out on the website of the device. An upload program for transmission of your own websites can be found on the CD-ROM PSWbasic/professional as well as an example for HTML pages. Those examples might serve as a sample and may be changed for your own applications.



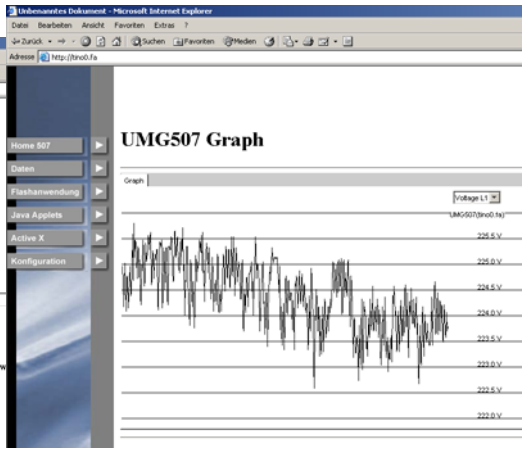
Actual measured values



Macromedia Flash Application



Configuration



JAVA Application

The measured values are bound into the corresponding website by a symbolic name. The parameter „rep“ followed by the symbolic name, e.g. „ul1“ is replaced by the actual measured value while calling up the website. An address list containing all symbolic names can be found on CD PSWbasic/professional.

Examples from address list:

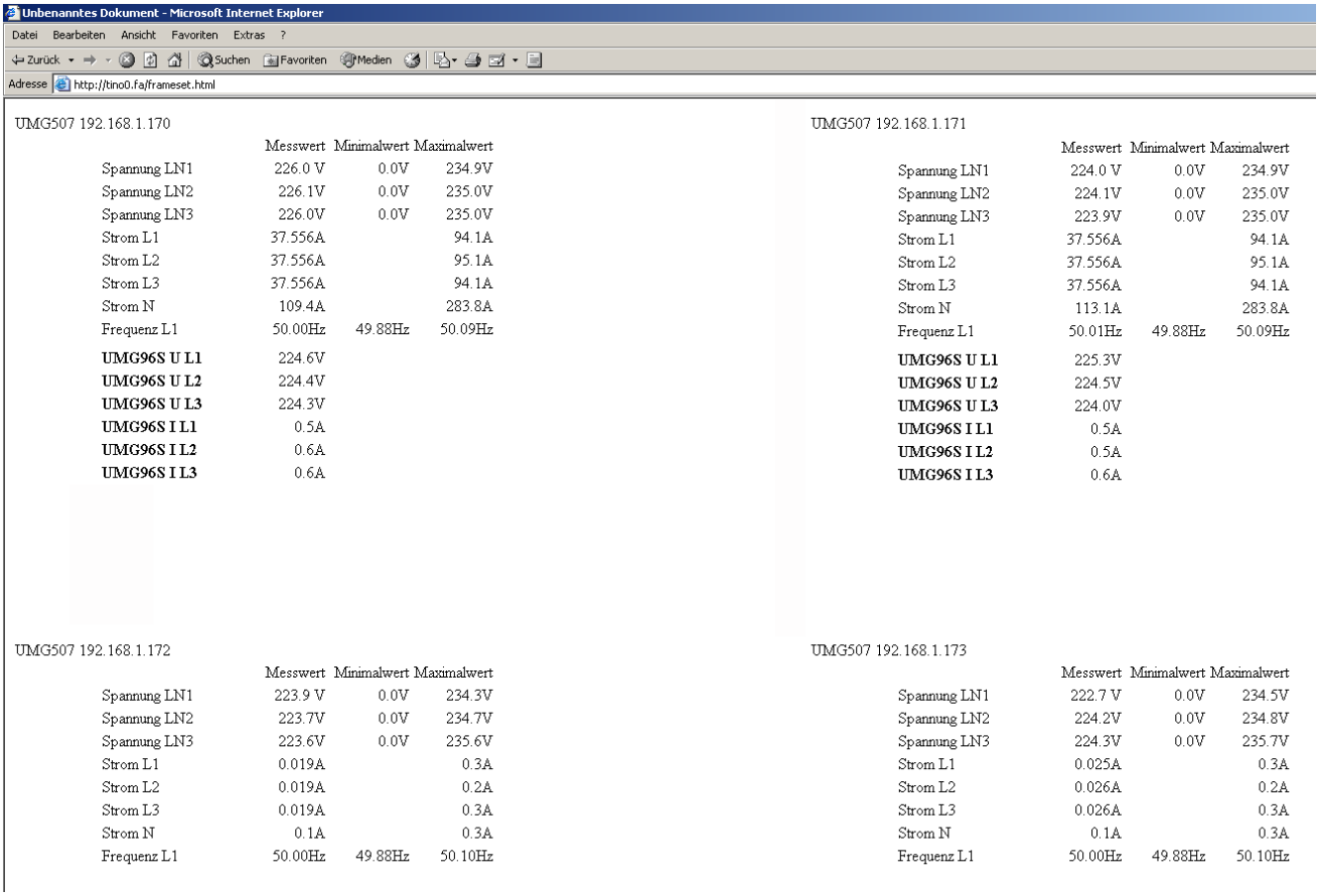
Description	**Symbolic Name**	**HTML command**
Voltage Phase L1	ul1	<rep ul1>
Voltage Phase L1	ul2	<rep ul2>
Voltage Phase L1	ul3	<rep ul3>
Current Phase L1	il1	<rep il1>
Current Phase L1	il2	<rep il2>
Current Phase L1	il3	<rep il3>
Power Phase L1	pl1	<rep pl1>
Power Phase L1	pl2	<rep pl2>
Power Phase L1	pl3	<rep pl3>

Examples for Webserver

In the following chapter some examples for the use of the webserver can be found.

Example HTML

Display of several UMG507E and UMG96S on a website with automatical actualization.



UMG507 192.168.1.170

	Messwert	Minimalwert	Maximalwert
Spannung LN1	226.0 V	0.0V	234.9V
Spannung LN2	226.1V	0.0V	235.0V
Spannung LN3	226.0V	0.0V	235.0V
Strom L1	37.556A		94.1A
Strom L2	37.556A		95.1A
Strom L3	37.556A		94.1A
Strom N	109.4A		283.8A
Frequenz L1	50.00Hz	49.88Hz	50.09Hz
UMG96S U L1	224.6V		
UMG96S U L2	224.4V		
UMG96S U L3	224.3V		
UMG96S I L1	0.5A		
UMG96S I L2	0.6A		
UMG96S I L3	0.6A		

UMG507 192.168.1.171

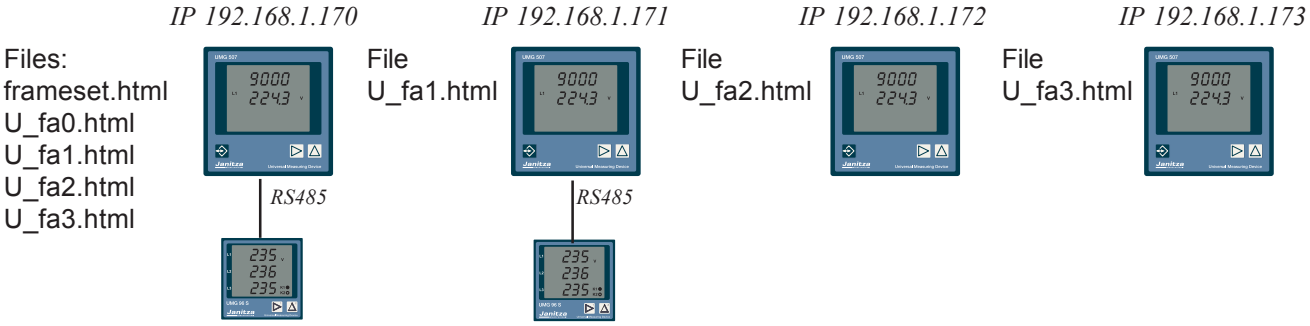
	Messwert	Minimalwert	Maximalwert
Spannung LN1	224.0 V	0.0V	234.9V
Spannung LN2	224.1V	0.0V	235.0V
Spannung LN3	223.9V	0.0V	235.0V
Strom L1	37.556A		94.1A
Strom L2	37.556A		95.1A
Strom L3	37.556A		94.1A
Strom N	113.1A		283.8A
Frequenz L1	50.01Hz	49.88Hz	50.09Hz
UMG96S U L1	225.3V		
UMG96S U L2	224.5V		
UMG96S U L3	224.0V		
UMG96S I L1	0.5A		
UMG96S I L2	0.5A		
UMG96S I L3	0.6A		

UMG507 192.168.1.172

	Messwert	Minimalwert	Maximalwert
Spannung LN1	223.9 V	0.0V	234.3V
Spannung LN2	223.7V	0.0V	234.7V
Spannung LN3	223.6V	0.0V	235.6V
Strom L1	0.019A		0.3A
Strom L2	0.019A		0.2A
Strom L3	0.019A		0.3A
Strom N	0.1A		0.3A
Frequenz L1	50.00Hz	49.88Hz	50.10Hz

UMG507 192.168.1.173

	Messwert	Minimalwert	Maximalwert
Spannung LN1	222.7 V	0.0V	234.5V
Spannung LN2	224.2V	0.0V	234.8V
Spannung LN3	224.3V	0.0V	235.7V
Strom L1	0.025A		0.3A
Strom L2	0.026A		0.2A
Strom L3	0.026A		0.3A
Strom N	0.1A		0.3A
Frequenz L1	50.00Hz	49.88Hz	50.10Hz



Task:
Calling up the website of UMG507 with IP address 192.168.1.170 shall display the measured values of four UMG507 as well as two UMG96S on a HTML page. The actualization shall be carried out each second.

Solution:
This can be solved by a corresponding Frameset. Please create a Framset with four seperate Frames on the UMG507 with IP address 192.168.1.170. The changeover to the other devices is carried out with the META TAG command **„meta http-equiv="refresh" content="1; url=http://192.168.1.171/U_fa1.html“**

Programming of the Frameset for device 192.168.1.170
File name: Frameset.html

```
<html>
<head>
<title>UMG507E  IP 192.168.1.170</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<frameset rows="50%,50%" cols="*" framespacing="0"" frameborder="no" border="0">
  <frameset rows="*" cols="50%,50%">
    <frame src="U_fa0.html" name="topFrame" scrolling="auto" >
    <frame src="U_fa1.html" name="topFrame" >
  </frameset>
  <frameset rows="*" cols="50%,50%">
    <frame src="U_fa2.html" name="topFrame" >
    <frame src="U_fa3.html" name="topFrame" >
  </frameset>
</frameset>
<noframes>
<body>
Your browser does not support Frames
</body>
</noframes>
</html>
```

Example of a Framesite for device IP 192.168.1.170
File name: U_fa0.html

```
<html>
<head>
<title>UMG507E  IP 192.168.1.170</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="refresh" content="1; url=U_fa0.html">
</head>
<body bgcolor="#FFFFFF">
<table width="332" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="332" height="19" valign="top">UMG507 192.168.1.170</td>
  </tr>
</table>
<table border="0">
<colgroup> <col width="80"> <col width="140"> <col width="80" align=right><col width="80" align=right><col
width="80" align=right> </colgroup>
<tr><td></td><td></td><td> measured value </td><td> Minimum value</td><td>Maximum value</td></tr>
<tr><td></td><td>voltage LN1</td><td> <rep ul1>V</td><td> <rep ul1min>V</td><td><rep ul1max>V</td></
tr>
.....
<tr><td></td><td>  </td><td>  </td></tr>
<tr><td></td><td><b>UMG96S U L1</b></td><td><rep darray_1>V</td></tr>
<tr><td></td><td><b>UMG96S U L2</b></td><td><rep darray_2>V</td></tr>
.....
```

With the META Refresh command, the site is loaded automatically within the interval of a second. The measured values of the UMG96S are bound in by the symbolic names of the data arrays (Address 9000 - 9126) (see address list).

Example of the Framesite for device IP 192.168.1.171
File name: U_fa1.html
(the files U_fa2.html, U_fa3.html have identical structure)

```
<html>
<head>
<title>UMG507E  IP 192.168.1.171</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="refresh" content="1; url=http://192.168.1.171/U_fa1.html">
</head>
<body bgcolor="#FFFFFF">
<table width="332" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="332" height="19" valign="top">UMG507 192.168.1.171</td>
  </tr>
  <tr>
    <td height="2"></td>
  </tr>
</table>
<table border="0">
<colgroup> <col width="80"> <col width="140"> <col width="80" align=right><col width="80" align=right><col
width="80" align=right> </colgroup>

<tr><td></td><td></td><td> measured value </td><td> minimum value</td><td>maximum value</td></tr>

<tr><td></td><td>voltage LN1</td><td> <rep ul1>    V</td><td> <rep ul1min>V</td><td><rep ul1max>V</td></
tr>
<tr><td></td><td>voltage LN2</td><td><rep ul2>V</td><td><rep ul2min>V</td><td><rep ul2max>V</td></tr>
<tr><td></td><td>voltage LN3</td><td><rep ul3>V</td><td><rep ul3min>V</td><td><rep ul3max>V</td></tr>
<tr><td></td><td>current L1</td><td><rep il1,3>A</td>   <td></td>          <td><rep il1max>A</td></tr>
<tr><td></td><td>current L2</td><td><rep il2,3>A</td>   <td></td>          <td><rep il2max>A</td></tr>
<tr><td></td><td>current L3</td><td><rep il3,3>A</td>   <td></td>          <td><rep il3max>A</td></tr>
<tr><td></td><td>current N</td><td><rep is>A</td>       <td></td>          <td><rep ismax>A</td></tr>
<tr><td></td><td>frequency L1</td><td><rep fl1,2>Hz</td><td><rep fl1min,2>Hz</td><td><rep fl1max,2>Hz</td></
tr>
<tr><td></td><td> </td><td> </td></tr>
<tr><td></td><td> </td><td> </td></tr>
<tr><td></td><td><b>UMG96S U L1</b></td><td><rep darray_1>V</td></tr>
<tr><td></td><td><b>UMG96S U L2</b></td><td><rep darray_2>V</td></tr>
<tr><td></td><td><b>UMG96S U L3</b></td><td><rep darray_3>V</td></tr>
<tr><td></td><td><b>UMG96S I L1</b></td><td><rep darray_4>A</td></tr>
<tr><td></td><td><b>UMG96S I L2</b></td><td><rep darray_5>A</td></tr>
<tr><td></td><td><b>UMG96S I L3</b></td><td><rep darray_6>A</td></tr>
</table>
</body>
</html>
```

While calling up the Frameset the page „U_fa1.html“ of UMG507 with IP address 192.168.1.170 is loaded. Due to the automatic actualization with command **„url“** the side of the device with IP address 192.168.1.171 is loaded automatically and actualized each second. The pages U_fa2.html and U_fa3.html have the identical structure except the URL.

Important note: To enable the parser of the UMG507 to replace the symbolic names by the actual measured values, those may be saved with the file ending „html“. The obsolete file ending „htm“ is not supported anymore. The file names may have 8 digits at maximum.

Example Macromedia Flash

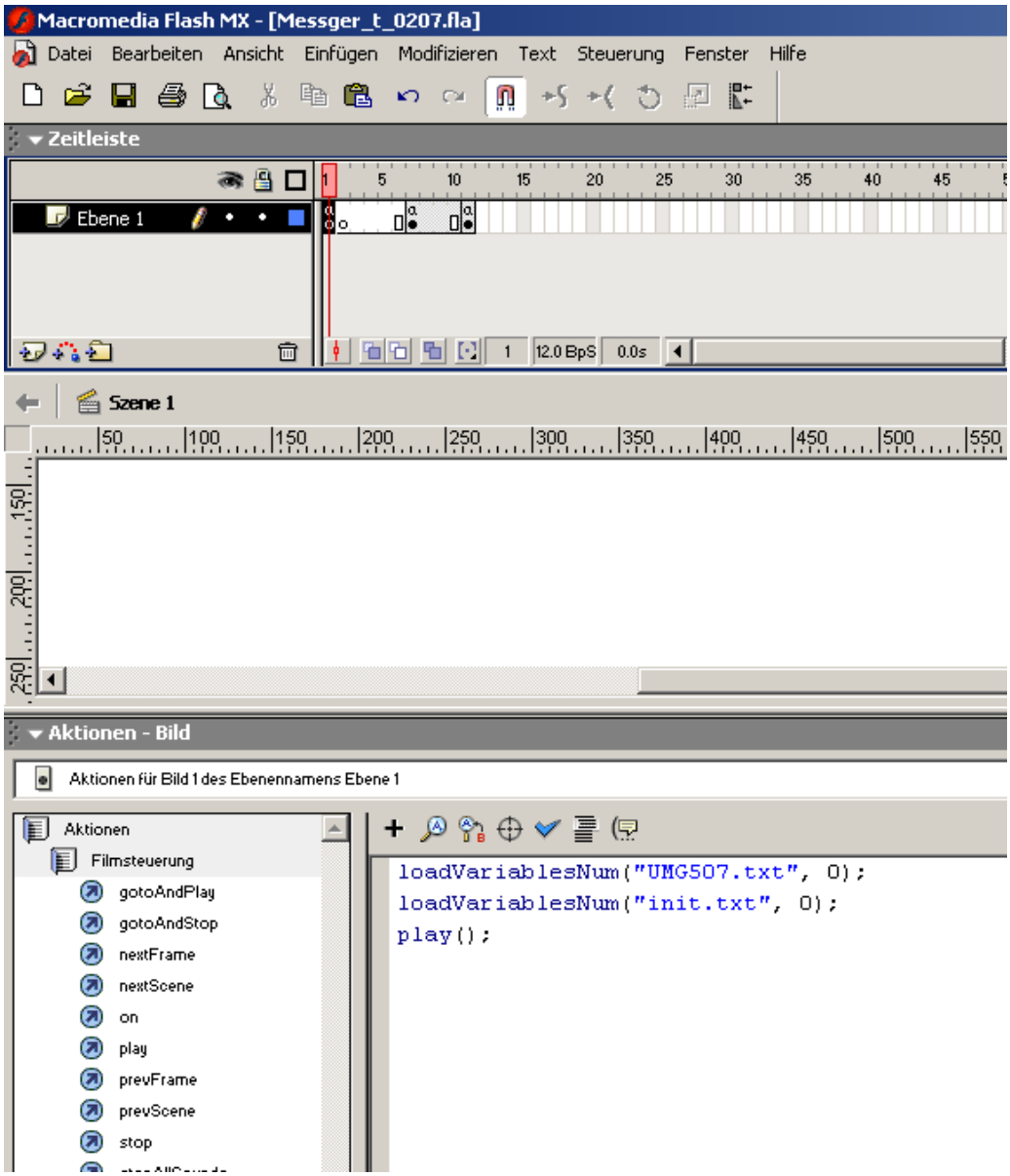
With Macromedia Flash very sophisticated user surfaces can be created. There are almost no limits for the graphical arrangement. The UMG507E/EP understands Flash files of the versions MX and 2004. More obsolete versions are not supported anymore.

Flash offers the opportunity to access external files during runtime and to show the content of the files within the actual Flash Player Film. The UMG507E/EP uses this possibility and replaces the parameter commands similar to a HTML pages in an „ini“ file by the actual measured data and writes them into a Txt file. This Txt file can be read by Flash during runtime.

the INI file must have the following structure for determination of the measured values:

&Var_0=<rep ul1>
&Var_2=<rep ul2>
&Var_4=<rep ul3>
&Var_6=<rep il1>
.....

&Var_0 is the internal Flash variable. The actual voltage value is assigned to this variable by the parameter **<rep ul1>**.



By the command **„loadVariablesNum("UMG507.txt", 0)“** the variables are read during film start and can be processed within the Flash Films. The Txt file is actualized each second by UMG507 and actualized cyclically within the film by Flash.

It is important, that the internet browser does not access temoral sites. This setting can usually be checked in menu „temporal internet sites“ within your internet browser.

Example JAVA

In the following example you can find the source text including the description of a JAVA MODBUS TCP/IP Clients. The Client was written and tested for the UMG507E/EP. On the CD-ROM PSWbasic/professional you can find the matching CLASS file.

```
/*
 * MOD_TCPClient.java
 *
 * Created on 30. Juni 2003, 08:30
 */

package de.janitza.mod_tcp;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;

import java.net.InetSocketAddress;
import java.net.Socket;

class TransactionIDWrong extends Exception{};

/** This class handles the communication with an device over MOD/TCP.
 * It only provides access to registers. So we are roughly Class 0 conform.
 * Typical usage:
 *    MOD_TCPClient modClient = new MOD_TCPClient();
 *    modClient.setUnitID(128);
 *    modClient.connect("192.168.2.214", 502);
 *    modClient.getRegisterAsFloat(0); // Voltage L1 on an UMG507
 *    // Possibly more calls here...
 *    modClient.close();
 * </code>
 */
public class MOD_TCPClient {

    /** Creates a new instance of MOD_TCPClient */
    public MOD_TCPClient() throws IOException {
        m_Socket = new Socket();
        m_Address = null;
        m_UnitID = 0;
        m_TransactionID = 0;
    }

    /** Connects to a MOD/TCP device.
     * @param host The host to connect to.
     * @param port The port to connect to. The specification says its always 502.
     * @throws IOException if an connection could not be estabilshed.
     */
    public void connect(String host, int port) throws IOException{
        m_Address = new InetSocketAddress(host, port);
        connect();
    }
```

```
/** Reconnects to a previosly connected device
     * @see #connect(String, int)
     * @throws IOException if an connection could not be estabilshed.
     */
    public void connect() throws IOException{
        if(m_Socket.isConnected()) {
            close();
        }
        m_Socket = new Socket();
        m_Socket.connect(m_Address);
        OutputStream os = m_Socket.getOutputStream();
        m_BufOutStream = new BufferedOutputStream(os);

        InputStream is = m_Socket.getInputStream();
        m_BufInStream = new BufferedInputStream(is);
    }

    /** Closes the socket and frees used resources.
     * @throws IOException if an connection could not be closed.
     */
    public void close() throws IOException{
        m_Socket.close();
        m_Socket = null;
    }

    /** Sets the unit identifier.
     * @param id The unit identifier.
     */
    public void setUnitID(byte id) {
        m_UnitID = id;
    }

    /** Gets the unit identifier.
     * @return The unit identifier previously set by setUnitID(int).
     */
    public byte getUnitID() {
        return m_UnitID;
    }

    /** Returns a given number of words of the specified register.
     * The words are returnd as bytes in a byte array.
     * @param refNr The register number.
     * @param wordCount the number of words(2 bytes) that should be requested.
     * @return An array of bytes with the size specified in the answer of the device.
     * @throws IOException if the connection is not open.
     */
    public byte[] getRegisterAsBytes(short refNr, short wordCount) throws IOException{
        byte[] sendBuf = new byte[5];
        byte[] result;
        byte[] recvBuf;
```

```
sendBuf[0] = 0x03;
        sendBuf[1] = (byte)((refNr & 0xff00) >> 8);
        sendBuf[2] = (byte)(refNr & 0xff);
        sendBuf[3] = (byte)((wordCount & 0xff00) >> 8);
        sendBuf[4] = (byte)(wordCount & 0xff);
       sendBytes(sendBuf);
        recvBuf = recvBytes();
        result = new byte[recvBuf[1]];
        for(int i = 0; i < recvBuf[1]; ++i) {
            result[i] = recvBuf[i + 2];
       }
       return result;
   }

   /** Returns a register as float.
    * Two words are requested. Since MOD/TCP is Bigendian we convert the value
    * into littleendian floats.
    * @param refNr The register to read from.
    * @return The float representing the register.
    * @throws IOException if the connection is not open.
    * @see #getRegisterAsBytes(short, short)
    */
   public float getRegisterAsFloat(short refNr) throws IOException {
       return bitsToFloat(getRegisterAsBytes(refNr, (short)2));
   }

         /** Converts Bigendian bytes to a float.
          * @param bytes The bigendian bytes.
          * @return The littleendian float.
          */
   private static float bitsToFloat(byte[] bytes) {
       float result = 0;
                int floatBits = bytes[3] & 0xff;
                floatBits += (bytes[2] & 0xff) << 8;
                floatBits += (bytes[1] & 0xff) << 16;
                floatBits += (bytes[0] & 0xff) << 24;
                result = Float.intBitsToFloat(floatBits);
       if((result < 1E-5) && (result > -1E-5))
           result = 0;
       return result;
   }


         /** Converts Bigendian bytes to a short.
          * @param bytes The bigendian bytes.
          * @return The littleendian short.
          */
   public static short bitsToShort(byte[] bytes) {
       short result = (short)(bytes[1] & 0xff);
       result += (short)((bytes[0] & 0xff) << 8);
       return result;
   }
   /** For internal use. Sends one datagram to a MOD/TCP device.
    */
   private void sendBytes(byte[] buf) throws IOException{
       byte[] sendBuf = new byte[7];
```

```
        sendBuf[0] = (byte)((m_TransactionID & 0xff00) >> 8);
        sendBuf[1] = (byte)(m_TransactionID & 0xff);
        sendBuf[2] = 0;
        sendBuf[3] = 0;
        sendBuf[4] = 0;
        sendBuf[5] = (byte)(buf.length + 1 & 0xff);
        sendBuf[6] = (byte)(m_UnitID & 0xff);
        m_BufOutStream.write(sendBuf);
        m_BufOutStream.write(buf);
        m_BufOutStream.flush();
    }

    /** For internal use. Receives one datagram.
     */
    private byte[] recvBytes() throws IOException {
        byte[] result;
        byte[] recvBuf = new byte[7];
        m_BufInStream.read(recvBuf);
        if( ! (m_TransactionID == bitsToShort(recvBuf))) {
//            throw new TransactionIDWrong();
        }
        ++m_TransactionID;
        result = new byte[recvBuf[5] -1];
        m_BufInStream.read(result);
        return result;
    }

    /** The socket used to connect to the device */
    private Socket m_Socket;
    /** The streams connected to the socket */
    private BufferedInputStream m_BufInStream;
    /** The streams connected to the socket */
    private BufferedOutputStream m_BufOutStream;
    /** The address to connect to */
    private InetSocketAddress m_Address;
    /** The unit identifier */
    private byte m_UnitID;
    /** The actual transaction ID */
    private short m_TransactionID;

        public static void main(String[] args) {
                try {
                        if(args.length < 1) {
                                System.err.println("Usage: java de.janitza.mod_tcp.MOD_TCPClient <host>");
                                System.exit(1);
                        }
                        MOD_TCPClient modClient = new MOD_TCPClient();
                        modClient.setUnitID((byte)128);
                        modClient.connect(args[0], 502);
                        System.err.println(modClient.getRegisterAsFloat((short)0)); // Voltage L1 on an UMG507
                        // Possibly more calls here...
                        modClient.close();
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }
}
```