Manual

IO-Mailer

Configuration and Programming

Rel. **US/GB 1.2 02/2002 FT**

Models:
57111, 57211
57511, 57611

Contents

2. revised edition

Subject to error and alteration:

Since it is posssible that we make mistakes, you mustn't use any of our statements without verification. Please, inform us of any error or misunderstanding you come about, so we can identify and eliminate it as soon as possible.

Carry out your work on or with W&T products only to the extent that they are described here and after you have completely read and understood the manual or guide. We are not liable for unauthorized repairs or tampering. When in doubt, check first with us or with your dealer.

1. Connectors

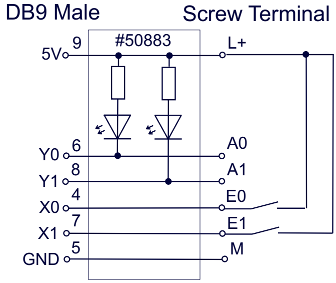
The serial port and the digital in- and ouputs of the IO Mailer use a DB9 female connector having the following pin configuration:

Pin#	Function	
1	nc	---
2	TxD	Output
3	RxD	Input
4	X0	bin. input
5	GND	0V
6	Y0	bin. output
7	X1	bin. input
8	Y1	bin. output
9	VCC	5

The sequence of the serial RxD/TxD lines is thus different from that of the DTE configuration as it would be found for example on the COM port of a PC, and actually corresponds to a DCE configuration. You can safely (i.e. without exceeding permissible input voltages or shorting the outputs) connect a terminal temporarily by using a 1:1 serial cable such as is used for connecting a modem.

The binary inputs X0 and X1 can withstand up to +/-30V, with an switching threshhold between +1V and +4V; open inputs are read as binary 0. Y0 and Y1 are open-collector outputs and can switch max. +30V and up to 100mA. The supply voltage VCC can be used as a pull-up voltage for the binary inputs and in addition supply small output loads up to a total of 20mA.

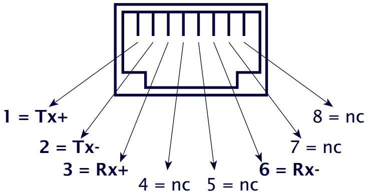
The optional screw terminal adapter P/N 50883 can be used to conveniently wire the in- and outputs.



1.1 Ethernet connection

An IEEE-802.3 compatible connection on a shielded RJ45 male connector is provided on the front panel. Through here you can connect the IO Mailer to a hub or switch. Pin/lead assignments comply with a standard MDI interface (AT&T258), so that you can use here a 1:1 cable having a length of max. 100 meters.

10BaseT, 10MBit/s and autosensing 100BaseTx/10BaseT, 100/10MBit/s are supported:



RJ45 female (per AT&T256)

Tip: Incorporation into a 100BaseTx-network is also possible with the 10BaseT-version by using an autosensing hub or switch. Such an autosensing component automatically sets itself to the transmission speed supported by the terminal device.

The Error LED on the front panel indicates the current link status: If it flashes at 1-2 second intervals, there is a fault in the connection to the hub or the connection is interrupted.

2. Startup

Each Mailer requires an *IP address* which must be unique within your network. A hostname must be assigned to it by DNS. In addition the IO Mailer needs its own *Mail Address*; a corresponding mailbox should also be newly created on your mail server („should“ means for test purposes you can also operate the IO Mailer without it being able to receive mail or even have a valid sender address. But if you want to receive and process mail it will definitely need its own mailbox, and under no circumstances can you use an already existing mailbox!).

To put it another way: To estimate the organizational effort, picture the IO Mailer like a new employee. He may be bringing his own PC, but it still has to be integrated into your network and your mail system.

2.1 Set IP-Address by using DHCP-/BOOTP protocol

Many networks use for centralized and dynamic assigning of IP addresses DHCP (*D*ynamic *H*ost *C*onfiguration *P*rotocol) or BOOTP. Which of the two you use makes no difference to the IO Mailer, since DHCP is simply an upward-compatible expansion of BOOTP. DHCP servers thus also use requests from BOOTP clients. The following parameters can be assigned to the IO Mailer using these protocols:

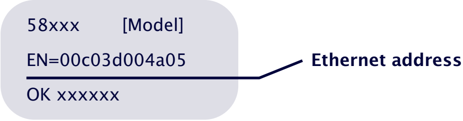
- IP address
- Subnet mask
- Gateway address

It is not possible to transfer other parameters or a lease time option.

Function

To obtain an IP address, the IO Mailer sends a corresponding BOOTP request as a network broadcast after each new start. The reply generated by the DHCP/BOOTP server contains the IP address as well as the subnet mask and gateway address. The IO Mailer immediately adds this information to its non-volatile memory.

Refer to your system administrator for startup of the IO Mailer in DHCP/BOOTP networks. If using DHCP to assign the address, you must note that a reserved IP address is required. To add to the respective address database the administrator will need the Ethernet address of the IO Mailer, which can be found on the sticker on the housing:



58xxx [Model]

EN=00c03d004a05

OK xxxxxx

Once the administrator has made the necessary entries, the IO Mailer automatically obtains the desired IP address after each reset. To ensure availability of the IO Mailer even should the DHCP/BOOTP server fail, the previous IP address is kept if there is no reply.

***Caution:** In DHCP environments the IP address to be assigned must be reserved by a fixed link to the Ethernet address of the IO Mailer. Under Windows NT this is done in the DHCP Manager under menu item* Reservations*. Linux provides the file* .dhcpd.conf *for making the corresponding entry.*

2.2 Set IP Address, Subnet Mask and Gateway without DHCP/BOOTP protocol

2.2.1 IP Address

Connect to the IO Mailer through a serial terminal (interface parameters *9600 baud, 8 data bits, no handshake*) and then connect the Mailier to its power supply. While it‘s starting type the letter „x“ at least three times at the terminal until the prompt „IP no.+<ENTER>:“ appears. Then enter the proposed IP address (four numbers between 0 and 255 separated by decimal points). If you make an erroneous entry the device responds with „*FAIL*“. If this happens, switch the IO Mailer off and start over again.

2.2.2 Gateway and Subnet Mask

This information is needed only if not all the required servers (see section „Network Environment“) are located in the same network segment as the IO Mailer. Ask your network administrator if you‘re not sure. To enter the data, make a Telnet connection with the IO Mailer (TCP Port Number 1111) and select the submenu *„SETUP System -> Setup TCP/IP“*.

2.3 Preparing to Receive Mail

Theoretically all the remaining operating parameters can be configured via e-mail. Some of this information must however already be correctly set before the IO Mailer can receive mail. This is also done through the serial terminal - either in a short question-and-answer dialog (as described below) or using a full configuration file which you load into the IO Mailer through the terminal (more on that in section 4). If your terminal allows sending of prepared text files, this would probably be the simpler solution.

First then connect to the IO Mailer through the terminal (*parameters: 9600 baud, 8 bits, software handshake*). Press the space bar once to display the keyboard menu and then select the option *„setup (by interview)“*. You may specify the servers for SMTP and POP3 either directly as IP addresses or by giving host names; if using the latter, you will also be prompted for the IP address of a DNS server. *„Administrator Mail Address“* means the address from which you want to send the remaining configuration data. Here you can either simply enter a *„*“*, but if so you must correct this in the final configuration (see section on „Access Restrictions“).

To test the configuration send the IO Mailer a mail with *„help“* in the subject line. If everything is correctly set it will pick up the mail within a minute or so and reply. You can observe this process in detail on the monitor; if the settings are wrong this will also tell you some possible causes.

3. Mail Commands

The IO Mailer responds only to mails whose subject line begins with a valid keyword (not case-sensitive). Valid keywords are:

help
Provides a short summary of available commands (non case-sensitive)..

unsubscribe
Allows the recipients of automatically generated mail to delete their own address from the configuration data. This is an uncomplicated solution for cases where a recipient had not agreed to inclusion in the service. This should not be necessary if use of the IO Mailer was carefully prepared in advance.

resume
Reverses the *„unsubscribe“* action.

status
Shows the status of the in- and outputs and count variables..

set
Assigns a value to one or more device variables. Can be used for directly controlling the binary outputs (e.g. *„set y0 = 1“* or *„set y0=0 y1=1“*), but also to reset the internal counters (e.g. *„set n0=0“*).

info
Provides the current device configuration in XML format which is also used for setting the configuration. To show only parts of the configuration you can additionally enter the name of one or more of the eight main branches, e.g.. *„info alarm“* or *„info servers, access“*.

template

Has a similar effect to „*info*“; empty fields in the configuration data are however also output. This is not as clear, but does make it easier to add missing information..

sample

Gives you a sample configuration file with trigger conditions used to create a mail message when X0 or X1 is present.

config

Change device configuration; the configuration data are contained in the body of the mail. For details on the structure of a mail see section 4.

update

Puts the IO Mailer in a special mode for firmware updates. You then have five minutes within which to open a TCP socket to Port 8000 of the IO Mailer and send it a text file with the new firmware. Caution: Until the update has been successfully completed (or until the five minutes have elapsed) the IO Mailer is unavailable for normal function!
For a mail subscriber to be able to use one of these commands he must in most cases have authorization to do so. More on this in the section on access restrictions.

4. Configuration

The IO Mailer is configured by sending it a mail with the subject *„config“*, which contains as the text an XML document with the configuration data. You can also use a serial terminal for configuring; in both cases you use the same XML documents.

The interface paramters are *9600 Baud, 8 bits, software handshake*. Press the space bar on the terminal once to display the keyboard menu. Select the menu item *„configure“* and use the terminal function*„Send text file“* to send the prepared configuration data.

4.1 Why XML?

XML (E*x*tensible *M*arkup *L*anguage) describes language rules used for dividing up the configuration data for the IO Mailer clearly into a tree structure with named branches. A somewhat simpler syntax (such as in the *Ini* files under Windows) would have accomplished the same purpose, but XML documents have a big advantage: There are many help options which make using it easier. For example, you can use Internet Explorer 5 to directly open, view and scroll branch by branch through the tree structure of an XML file as the IO Mailer sends it as a reply to the mail command *„info“*.

4.1.1 Differences from HTML

If you are already familiar with HTML, much of XML will already be recognizable. There are however a few differences, the most important of which are:

- Element and attribute names are case-sensitive.
- Every opening structure element also needs a closing element, i.e., *<name>...</name>*. Free-standing elements are only allowed if they are designated as such by a forward slash in front of the closing arrow bracket:: *<name/>*.
- Attribute values must always be enclosed in quotation marks, e.g.,*<name attr="1"/>*

- Except for the five characters < > & “ ´ (see also table below), there is no coding for special characters. Formulations such as *&auml;* instead of ä are *not* allowed.
- As in the case of HTML, XML also uses for each document a sructure an element pair which encloses the entire document contents. Put another way: The tree structure into which the document is divided originates from just one root. But whereas this root is always *<HTML>* for HTML documents, its name for XML documents is defined depending on the application - for the configuration data of the IO Mailer it is *<io-mail.1>*.

4.1.2 Working with document templates

The IO Mailer can create a rough framework of the XML documents needed for configuration using the mail command *„template“*. All you then need to do is enter your own configuration data. But since you will already have to have configured the mailer for it to be able to even process such a command, the same function is also available on the serial terminal. Select *„full XML template“* in the keyboard menu.

Even if you have never used XML or HTML and are scratching your head over all the arrow brackets, you should still be able to fill out these forms without much difficulty as long as you observe the following formalities:

Special characters

The texts you enter may not include the arrow brackets or three other special characters. Instead use code according to the following table:

<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos

Fill in empty fields

You can enter simple text between associated opening and closing structure elements, and attribute text enclosed by quotation marks. Example:

before:

```
<repeat></repeat>
<action var="" val=""/>
```

after:

```
<repeat>00:00:01</repeat>
<action var="Y1" val="0"/>
```

Long texts

It is not necessary to put the opening and closing structure in the same line, and even the text itself may extend over several lines. Example:

before:

```
<subject></subject>
<message></message>
```

after:

```
<subject>
rather long subject line, at least for our purposes
</subject>
<message>
Somewhat longer paragraph.
Line break,
     left margin
and  size   of the    spaces between words     are
also not critical.
</message>
```

Deleting and copying

Even if each branch appears only once in the original document template: Many can be used over again, and many can even be eliminated completely. The document created by the IO Mailer is formatted such that in most cases you can‘t make a mistake as long as you always delete or copy only entire lines. Example (specifying more than one DNS server):
before:

```
<dns></dns>
```

after:

```
<dns>172.16.232.14</dns>
<dns>172.16.232.15</dns>
```

Another example where the unused elements and attributes are deleted:

before:

```
<trigger op="DELTA" var="N1" val="3" delay="" recover=""/>
<repeat></repeat>
<mailto>test@xyz</mailto>
```

after:

```
<trigger op="DELTA" var="N1" val="3"/>
<mailto>test@xyz</mailto>
```

Note however: There are two important cases where copying or deleting individual lines is not sufficient. First case: The *alarm* branch may contain several *event* branches but only has one in the original document template. How to properly configure with multiple *event* branches is shown in the corresponding examples in the section on „Events and Actions“.

The other case is when the configuration is divided into independent main branches of which you may want to delete only individual ones (more on this in the next section). Each branch begins and ends with a line which contains its opening and closing structure element, for example <access> *and </access>* for the *access* branch. You are allowed both to delete the contents of a branch (the structure elements remain) as well as the complete branch including its structure elements. The effects will be different in each case however. If only the contents of a main branch is deleted, all the associated data from the device configuration is also deleted (you would only want this in exceptional cases). But if you delete the main branch itself, the corresponding section in the device configuration remains unaffected.

Comments

The entire text between *the characters <!-* and -> is ignored. This means you can also quasi-hide entire sections of your document. Not permitted however is embedding comments inside each other. Example:

```
<!-- Hmm... Does that make sense? -->
<!--
<suspect>*</suspect>
-->
```

4.1.3 Division into main branches

The configuration is divided into eight main branches as described in the following sections:

```
<servers>
<pop-account>
<device-id>
<report>
<zone>
<access>
<schedule>
<alarm>
```

These main branches can be configured independently of each other. This means you can for example change the time zone of the Mailer without having to redefine the trigger conditions at the same time. But for each branch which is even mentioned when sending configuration data, the following rule applies: All existing data are deleted before new data are accepted. So if you want say to enter a new DNS server, you also have to know the time and mail server and specify again. But this should not be a major obstacle to making changes, since the current configuration data can always be downloaded from the IO Mailer itself (mail commands *„info“* and *„template“*.

It is therefore advisable to change only the affected branches when making minor changes. In the example above you would thus send a mail having the subject *„info servers“*, edit the XML document you get as a result, enter the new DNS server and then send the mail back with the subject *„config“*. This avoids two possible problems: First of all, IO Mailer operation is needlessly disturbed when certain branches are rewritten even though none of the contents has changed (*report:* the measured value series are deleted, *alarm:* the event processing is reinitialized). And secondly there are branches which some mail subscribers only look at without having permission to change them anyway (*servers, pop-account, access*).

The following examples assume throughout this division into separate branches and, and each exemplifies a change in just one branch. In the case of printed code we are dealing with complete XML documents which would be directly accepted by the IO Mailer but with the unique feature of containing only a single main branch

To avoid misunderstanding, we will indicate here again how several of these examples would have to be combined in order to change several branches at the same time:

```
<io-mail.1>
  <servers>
  ...
  </servers>
  <pop-account>
  ...
  </pop-account>
</io-mail.1>
```

The following example would be the *wrong* method:

```
<io-mail.1>
  <servers>
  ...
  </servers>
</io-mail.1>   <!-- Not this way! Everything after the -->
<io-mail.1>    <!-- first </io-mail.1> is ignored! -->
  <pop-account>
  ...
  </pop-account>
</io-mail.1>
```

To put it another way: The structure element pair which encloses the entire document (in this case *<io-mail.1>...</io-mail.1>*), is allowed of course to only appear once. Otherwise you would be dealing with multiple configuration files - except that you‘re only allowed to send one file at a time to the IO Mailer.

4.1.4 Time formats

There are several places within the configuration data where references to time of day and time spans occur. The format you use here is in all cases (except for the time zone) HH.MM.SS, in other words 04:30:00 for 4:30 in the morning and 00:00:10 for ten seconds. Time spans can alternately be represented as a number of seconds, for example 90 for one and a half minutes or 0.25 for a quarter second.

4.2 Network environment <servers>

The following severs need to be present in your network:

- A DNS-Server. You can get by without one, but then you have to address all the other servers directly by their IP address and cannot use the simpler and easier to use host name.
- An SMTP server and a POP3 server for sending and receiving mail. These services are frequently combined

on one and the same server (in the case of the Microsoft Exchange Server it's even one and the same product).

- A time server conforming with RFC-868. Highly recommended so that the IO Mailer can generate the correct time stamp. There are time servers which are accessible from anywhere in the world through the Internet (such as clock.psu.edu, swisstime.ethz.ch) as well as free software which can be used to configure Windows computers as time servers (more on this in the Appendix). But the easiest method is to have your own Unix or Linux server available. There you only need to ensure that the service indicated as *„time“* is activated in the *„inetd.conf“* file..

Example:

```
<io-mail.1>
  <servers>
    <connect>00:00:05</connect>
    <response>00:01:00</response>
    <terminate>00:00:10</terminate>
    <dns>172.16.232.15</dns>
    <dns>172.16.232.14</dns>
    <smtp>mail.company.com</smtp>
    <pop3>mail.company.com</pop3>
    <time>atropos.company.com</time>
    <time>clock.psu.edu</time>
    <time>time.ien.it</time>
  </servers>
</io-mail.1>
```

Each server type may have several alternate addresses or host names. In such cases the IO Mailer normally attempts to connect to the first listed. Only if this fails does it go through the remaining addresses in order.

It can never hurt to assign more than one DNS or time server. A configuration containing more than one SMTP servers is unusual however, nor does having more than one POP3 servers to choose from generally make sense (due to the way this service works: The POP 3 server represents a collection of mailbox es, and if it fails, their contents are no longer available).

The connect value which you enter here determines how much time the IO Mailer allows for accepting a TCP connection. Unless you specify differently, five seconds is here the default value. Larger values are generally impractical, since it then takes too long to detect failure of a server. Shorter values have even worse consequences, since this may prevent network connections from even being established.

The IO Mailer carries our longer dialogs with its POP3 server within an existing connection. Unfortunately it is normal for heavily loaded servers to require a lot of time for the individual replies.

The *Response* value sets an upper limit for these reponse times. If the specified time is exceeded the IO Mailer disconnects. Unless you specify differently the IO Mailer sets this value to one minute.

The counterpart to the *Connect* value is the *Terminate* value, which specifies the time within which servers must acknowledge establishment of a connection. Ten seconds is considered the standard.

4.3 Mailbox access <pop-account>

The IO Mailer needs this information in order to receive mail. Example::

```
<io-mail.1>
  <pop-account>
    <mailbox>eabox1</mailbox>
    <password>caligula</password>
    <interval>00:01:00</interval>
  </pop-account>
</io-mail.1>
```

***Caution:** The mailbox you specify here must belong solely to the Mailer! This is because it deletes all mails which it has read - even those whose contents it doesn‘t understand. Therefore you must strictly avoid allowing the Mailer to also use the mailbox of a normal network subscriber! Using one mailbox for several Mailers would also not make sense. (Mails to this mailbox would then only reach one of the Mailers; the problem is, you can never tell in advance which one will actually receive the mail.)*

Note also that the IO Mailer does not support passwords - also known as MD5 or challenge-response authentication. The ID for the mail server is thus sent uncoded over the network to the POP3 server for each mailbox query and can be intercepted by other network subscribers with relatively little technical effort. Besides, the password does not have any special position within the configuration data and can be easily retrieved by authorized persons together with the rest of the configuration.

The actual security risk has less to do with the possibility of unauthorized access to the mailbox of the IO Mailer, since mails located there usually contain no great secrets. Rather you should make sure that as few additional rights as possible can be derived from knowledge of the password. In particular you should not „reuse“ the password of a human user.

The polling interval determines how fast the IO Mailer can react to incoming mail commands. To prevent unncecessary network loading you should not be too greedy here. A response time on the order of one minute is generally sufficient.

4.4 Mailer sender address <device-id>

Example:

```
<io-mail.1>
  <device-id>
```

```
    <host>eabox1.company.com</host>
    <address>eabox1@company.com</address>
    <name>IO server in stairway</name>
    <owner>johndoe@company.com</owner> <!-- optional -->
  </device-id>
</io-mail.1>
```

The IO Mailer uses this information for sending mail. It provides its own hostname in order to distinguish itself from the SMTP server. The mail address is noted as the sender on all mails and allows the recipient to reply to the IO Mailer (especially helpful for unsubscribing unwanted notifications). The „Name“ of the Mailer can be any text and is shown at the recipient instead of or together with the sender address. Here you could use a short reference to the function, location or operator of the Mailer.

The *<owner>* field should contain the mail address of a person who is responsible for operation of the IO Mailer. In the first place this address is noted in the headers of all automatically generated mails; anyone having problems who is looking for someone to contact will in all likelihood note this section of a message. In addition the IO Mailer messages concerning special actions to the address indicated here (e.g. that a recipient has deleted himself from the configuration using the *„unsubscribe“* mail command).
In many cases this entry is actually superfluous - especially if you configure the IO Mailer always via e-mail rather than through the serial port. In this case the contact person is automatically whoever last changed the configuration branch *<alarm>*.

4.5 Appearance of the device status <report>

Every mail which is sent as the result of an event (or as a reply to the *„status“* command) contains as the text at minimum a listing of the six device variables. In the simplest case this would look approximately as follows:

```
Thu, 29 Jun 2000 13:52:44
```

```
X0 = 0
X1 = 1
Y0 = 0
Y1 = 0
N0 = 37
N1 = 4
```

Since this doesn‘t mean much to the mail recipient, a comment text can be attached to each variable; in addition the binary variables X0, X1, X0, Y1 can be assigned a text for the state 0 and 1.

In addition, this configuration section can be used to control the recording of measurement series: Up to 256 accumulated values can be stored in a freely definable time-slot pattern for the two counters *N0* and *N1*.

Here is a complete example:

```
<io-mail.1>
    <report>
        <label var=“X0" alias=“light barrier“
               state0=“frei“ state1=“interrupted“/>
        <label var=“X1" alias=“doorbell“
               state0=“losgelassen“ state1=“activated“/>
        <label var=“Y0" alias=“lighting“ state0=“aus“ state1=“on“/>
        <label var=“N0" alias=“visitors“/>
        <history var=“N0" depth=“4" step=“00:00:15"/>
        <history var=“N1" depth=“12" step=“00:00:05"/>
    </report>
</io-mail.1>
```

And a possible output which could be generated:

```
Wed, 2 Aug 2000 17:02:04
X0 = 0  (light barrier free)
X1 = 0 (doorbell button released)
Y0 = 1 (lighting on)
Y1 = 0
N0 = 6 (visitors)
N1 = 10
```

```
17:01:49, N0 = 5 (+1)
17:01:34, N0 = 2 (+3)
17:01:19, N0 = 0 (+2)
17:01:04, N0 = 0
17:02:00, N1 = 9 (+1)
17:01:55, N1 = 9
17:01:50, N1 = 6 (+3)
17:01:45, N1 = 6
17:01:40, N1 = 6
17:01:35, N1 = 4 (+2)
17:01:30, N1 = 0 (+4)
17:01:25, N1 = 0
17:01:20, N1 = 0
17:01:15, N1 = 0
17:01:10, N1 = 0
17:01:05, N1 = 0
```

The numbers in parentheses behind the individual measurement series values indicate the difference from the next most recent value. In the case of the first line of each list this is the difference from the current value, which is not contained in the list itself but rather in the summary of variable which is located at the top of the mail.

4.6 Time zone <zone>

Indicates the time zone where the IO Mailer is located, e.g. +0100 for CET (Central Europe Time), +0200 for CEST (Central Europe Summer Time). As you can see from these examples, the difference in hours and minutes from Greenwich Mean Time GMT is given in two digits each without spaces. Example:

```
<io-mail.1>
  <zone>+0100</zone>
</io-mail.1>
```

Important: The switch between summer and winter time is not made automatically; the user must enter this manually.

The IO Mailer uses the zone information for converting the world time as it receives it from its time server into local time. Specifying an incorrect time zone will cause the time of day in the status report of the Mailer to be offset from the actual local time.

If the world time received from the time server is correct, incorrect time zone information will still allow you to determine when a mail was actually sent. This is because not only the supposed local time is noted in the mail header, but also the time zone which this local time should be associated with. Some mail clients process this information in order to immediately convert the time when a mail was sent into the local time of the recipient.

4.7 Access restrictions <access>

Before the IO Mail executes a mail command it checks to see whether the sender is authorized to perform the action in question. Two authorized groups of persons can be set up for this: Users *(user)* and administrators *(admin)*. Put simply, there must be at least user rights in order to change anything on the IO Mailer, and there must be administrator rights in order to change critical details in the configuration. („Critical“ means changes which would allow you to lock yourself out, i.e. those changes which may not be capable of being reversed via mail but only on site at the serial terminal.) A third authorized group of persons would be the confidential recipient, which however automatically includes anyone who was not expressly excluded as suspect. This looks as follows in detail:

Anyone	help status
Confidential recipient	unsubscribe resume
User	set info template config (with the branches report, alarm, sleep, device_id, zone)
Administrator	update config (with the branches access, servers, pop_account)

A sensible policy would be for example to assign user rights to all employees of your own company and, and to assign additional administrator rights only to a few specifically designated employees. The configuration would look as follows:

```
<io-mail.1>
  <access>
    <user>company.com</user>
    <admin>john.doe@company.com</admin>
    <admin>k.zuse@company.com</admin>
  </access>
</io-mail.1>
```

Entries which contain a „@“ represent simple mail addresses; entries without an „@“ apply to all senders in a complete mail domain. The entry *„company.com“* would here for example allow user rights for *„frank@company.com“* and *„klaus@company.com“* , likewise *„thomas@branch.company.com“*, but not *„stefan@other-company.com“* or *„martin@company.comfinitiv.com“*.

A weakness in all this is however that mail sender addresses can be falsified. But anyone attempting this has a few problems to overcome, for example the command acknowledgements from the IO Mailer will no longer reach him. (Acknowledgements are always sent to the address which was used to check for authorization; *Reply-To* headers are ignorerd.) But this still does not preclude the possibility of such an attack.

If you like you may configure the IO Mailer so that changes can only be made directly through its serial port:

```
<io-mail.1>
  <!-- disallow further authorized access by mail -->
  <access>
    <suspect>*</suspect>
  </access>
</io-mail.1>
```

The „*“ represents a joker which looks at all the addresses and in this case puts all mail subscribers in the lowest authorization level. A clear interpretation of the user group *suspect* would be something like „Recipients who are not even permitted to use the *unsubscribe* command because they would just misuse it“, which admittedly sounds somewhat exaggerated. But the purpose of this example is to prevent sabotage by means of mails having a falsified sender address. The mistrust is not really directed towards the persons who are listed as recipients.

4.8 Time-based control <schedule>

The actions of the IP Mailer can also be controlled to a limited extent as a function of the time of day and day of the week. This is the purpose of two additional binary variables *WE* and *NT*, which represent the status „weekend“ and „night“. In order to use these you must first specify which time periods shall be considered night and weekends.

```
<io-mail.1>
  <schedule>
    <night from="22:00:00" to="06:00:00"/>
    <weekend from="Fr 22:00:00" to="Mo 06:00:00"/>
  </schedule>
</io-mail.1>
```

Note:

The weekday can be abbreviated using German or English conventions, whereby only the first two letters are processed. This means for example. *„Di“*, *„Die“*, *„Tu“* or *„Tue“* would all be valid designations for Tuesday..

As long as the realtime clock on the IO Mailer could not be set (for example because none of the configured time serves was accessible or because none was even specified), time-based control is disabled and the variables *WE* and *NT* have the value 0.

The time-dependent variables are also displayed in the device status - although only if they are actually used and (as shown here) if times for their function have been specified. The device status would then contain two additional lines in the report which could look something like this:

```
<label var="WE" state0="weekday" state1="weekend"/>
<label var="NT" state1="night"/>
```

4.9 The serial port

Compatible with the IO-Mailer are either a serial terminal for configuration and diagnostic purposes, or another peripheral having a serial port which can then send mail or be remote controlled via mail. Depending on which option is used, the serial port on the Mailer can be in either *terminal mode* or *peripheral mode*, with additional configuration data for peripheral mode:

```
<io-mail.1>
  <serial>
    <line baud=“9600" data=“8" stop=“1" parity=“N“ flow=“X“/>
    <escchar>27</escchar>
    <packet mode=“D“ timeout=“0:00:10" chopmsg=“[...]“/>
    <dpacket endchar=“13" filter=“1"/>
  </serial>
</io-mail.1>
```

The serial port parameters are specified with *line*. Available baud rates are 1200, 2400, 4800, 7200, 9600, 14400, 19200, 28800, 38400 or 57600. Data bits can be 7 or 8, and stop bits 1 or 2. N, E and O stand for no, even and odd parity. Flow control may beither N for no handshake, or X for software handshake (XOn/XOff). Hardware handshake is not supported, since the serial port on the Mailer does not have any handshake lines. Keep in mind: The port parameters in terminal mode are, regardless of what you have configured here, always 9600 buad, 8N1, software handshake.

The IO-Mailer starts up by default in peripheral mode. To activate terminal mode instead, enter „x" again as described above under startup until the prompt to enter the IP address appears, then skip this prompt by pressing Enter or Escape. If you find this too clumsy, use *escchar* to define a character which when entered returns the Mailer from peripheral mode to terminal mode. Enter here an ASCII code, e.g. 27 for Escape. Remember, however, that there will be hard to predict and probably unpleasant consequences if the peripheral happens to send this special character for any reason.

Serial data that are supposed to trigger events are first subdivided by the Mailer into packets. The way in which this is done is specified by the *packet* and under some conditions the *dpacket* parameters. There are two main modes: Command mode (*mode="C"*), in which the peripheral itself can determine the recipient and contents of every mail to be sent, and data mode (*mode="D"*), in which the output of the device can be analyzed without having to make any special adaptations for working with the IO-Mailer. A packet in *command mode* has the following format:

```
Subject EOL
Recipient address(es)  EOL
Text line EOL
...
Text line EOL
EOL
```

If multiple recipients are specified, commas or semicolons are used as separators, e.g. „ab@example.de, xy@example.de“ (without the quotation marks). EOL (end of line) may be any of the usual end of line characters: CR, LF or CR/LF. The packet end is indicated by two successive line ends.

Serial packets can also be considered as ended if no more serial data arrive over a longer period of time. Unless otherwise specified, the timeframe here is one minute. Enter *timeout="0"* to suppress the use of time limits. As already mentioned, a packet also ends in commmand mode if two line ends come in succession. In data mode, a special packet terminating character can be defined. Here for example you could specify 10 for LF or 13 for CR if each line is supposed to represent a packet. Use *filter="1"* to prevent the packet end character to be stored in the buffer. This is especially important if the serial data (over the placeholder %s) are supposed to be inserted into a mail.

An 8 KB receive buffer is available for the serial data. This makes a buffer overflow relatively improbable. Should that happen, however, attach the string indicated by *chopmsg* at the point where the packet had to be cut off.

4.10 Events and actions <alarm>

This part of the configuration is used to control the actual serviceableness of the IO Mailer. The basic principle is that changes in the inputs can in turn cause changes in the outputs and/or the sending of a mail. Specifically we are talking here about variables, operators, trigger conditions, events, actions and recipient lists. These terms will be described now:

Variables
There are four binary variables which are assigned to the inputs (*X0*, *X1*) and outputs (*Y0, Y1*) and which can assume the states 0 and 1. In addition the IO Mailer uses two counter variables (*N0, N1*) to monitor changes on the inputs. Cycles are counted rather than edges. Put another way:

Pressing and releasing a button on an input increments the associated counter not by two, but rather by one. The rule is that a count is made each time an input returns to the state it was in before the device was turned on. In the above example the counter status would only change when the button is released, regardless of whether this button opens or closes a contact. It should not make much difference anyway in most applications whether in fact the count is changed by a rising or falling edge.

The two binary variables for indicating night and weekend (*NT, WE*) were already covered in the previous section.

Operators

There are six simple operators, each of which compares a variable against a constant value:

EQ	"equal"
NE	"not equal"
GT	"greater than"
LT	"less than"
GE	"greater or equal"
LE	"less or equal"

There is also a seventh *„DELTA“* operator. Its test conditions are met as soon as the associated variable value has changed by a minimum amount. (Here the operator uses a momentary value of its variables as a reference. Exactly when it records these reference values will be described below under „Events“.)

Trigger conditions

A trigger condition essentially combines an operator with its associated comparison objects „variable“ and „constant number“.

In addition you may if necessary specify a *delay* time. This means that the test result of the operator must first be met without interruption over a specified period of time before the trigger condition itself will be met. Likewise you may specify a *recover* time, which has the opposite effect: A trigger condition remains met for a prescribed interval even if the test result of the operator is no longer met. These parameters are intended to prevent a closed event from being registered as a sequence of multiple events due to temporary disturbances (such as bouncing contacts). In practice of course many noise signals are already filtered out as a result of the fact that the IO Mailer input variables are only updated at intervals of 10 ms.

Example: Assume a button is connected to Input *X0*. The first of the following trigger conditions is met as soon as the button was pressed ten times, and the second is met if the button remains depressed for at least a half second:

```
<trigger op="DELTA" var="N0" val="10"/>
<trigger op="EQ" var="X0" val="1" delay="0.5"/>
```

Serial trigger conditions

A serial trigger condition is met as soon as a data packet arrives, and is immediately lifted again. In general, you will also want to require that the packet contain a certain text, whereby the text search can be done optionally with (*case="1"*) or, to ignore capitalization, (*case="0"*). In the examples below, the first condition would be met by the text „End", but not by „Legend", and the second would be met both by „Alarm" and by „Fire alarm"..

```
  <dtrigger substring="End" case="1"/>
  <dtrigger substring="alarm" case="0"/>
```

Actions

An „action" in the context of the IO Mailer is the assignment of a value to one of its variables. (Sending a mail does not therefore fall under this heading.) The usual way of doing this would be by setting output variables and flags (Y0, Y1, M0, M1), and by sending serial data (assignment to TX), but if you really want to, you can also change the counter status (N0, N1). References to the time of date variables and inputs are not allowed.

Recipient lists

Recipient lists may be assigned to a particular event or apply globally. A mail is sent when an event occurs which has at least one recipient address associated with it. In case of globally valid addresses every event thus generates a mail.

Events

An event consists of trigger conditions, actions and a list of mail recipients. In case of multiple trigger conditions all must be met simultaneously in order to initiate the event (implicit AND-operation).

As soon as an event occurs, the associated *actions* are immediately carried out (with a response time of around 10 ms), any *DELTA* operators associated with the trigger conditions are reset (by instructing them to use a new reference value), and a mail request is entered which is not deleted until a corresponding mail is actually generated. Even if sending of the mail can be delayed considerably for various reasons, it is still ensured that no mails are forgotten simply because for example the associated event (e.g. pressing a button or key) was ended too soon. Mail requests do not accumulate: If for example an event occurs multiple times before the opportunity to send presents itself, still only one mail is generated.

All this happens as a rule only when an event newly occurs, but it may be repeated at regular intervals by using an additional *repeat* parameter as long as the event lasts (i.e. as long as its trigger conditions are met). Note however that events which contain a *DELTA* operator by definition can never persist and therefore cannot be repeated.

4.10.1 Simple IO Examples

In this first example two events are defined which occur when a signal is applied to Input *X0* or *X1*. The mails which are generated differ in their subject lines, but the recipient and the message text are the same in both instances. The message text „." represents a special situation. The entire text following an inidividual period is cut off when the mail is sent - including the status report which is normally appended to every mail (cf.

description of the configuration branch *report*). This may be desirable in some cases so as not to confuse the recipient with unnecessary details.

```
<io-mail.1>
  <alarm>
    <event>
       <trigger op="EQ" var="X0" val="1"/>
       <subject>Please refill snack vending machine</subject>
    </event>
    <event>
       <trigger op="EQ" var="X1" val="1"/>
       <subject>Please refill cigarette vending machine</subject>
    </event>
    <!– the following applies to both events –>
    <mailto>john.doe@company.com</mailto>
    <message>See building superintendent for keys.</message>
    <message>.</message>
  </alarm>
</io-mail.1>
```

As already mentioned, multiple trigger conditions can be linked with each other. This would be relevant particularly in the context of the time-based variables *(WE, NT)*, whose changes are almost certainly not interesting in and of themselves.

```
<io-mail.1>
  <alarm>
    <event>
      <!– Both of the following conditions must be met –>
      <!– at the same time (implicit AND-operation): –>
      <trigger op="EQ" var="X0" val="1"/>
      <trigger op="EQ" var="WE" val="1"/>
      <subject>Fault in the heating system</subject>
      <message>And of all times on a weekend...</message>
    </event>
    <mailto>john.doe@company.com</mailto>
  </alarm>
</io-mail.1>
```

The next example introduces somewhat more complicated trigger conditions. Imagine that Output *Y0* controls an indicator lamp. The first two events work together to cause this indicator lamp to flash (0.3 sec. on, 0.2 sec. off) as long as there is a signal on Input *X0*. The third associated event makes sure that the lamp goes out when the input signal is no longer present on *X0*. The fourth event would function independently of the other two. It sends messages as long as the signal is present on *X0* - initially after one second, then at 10-minute intervals.

```
<io-mail.1>
  <alarm>
    <event>
      <trigger op="EQ" var="X0" val="1"/>
      <trigger op="EQ" var="Y0" val="0" delay="0.2"/>
      <action var="Y0" val="1"/>
    </event>
    <event>
      <trigger op="EQ" var="X0" val="1"/>
      <trigger op="EQ" var="Y0" val="1" delay="0.3"/>
      <action var="Y0" val="0"/>
    </event>
    <event>
      <trigger op="EQ" var="X0" val="0"/>
      <action var="Y0" val="0"/>
    </event>
    <event>
      <trigger op="EQ" var="X0" val="1" delay="1"/>
      <repeat>00:10:00</repeat>
      <mailto>john.doe@company.com</mailto>
      <subject>Please empty containers</subject>
      <message>.</message>
    </event>
  </alarm>
</io-mail.1>
```

You can also cause the IO Mailer to send a turn-on message, such as for checking whether and when momentary interruptions in the power supply occur. For this you would formulate an event without any trigger condition. Such an event then only occurs once (because it gives no trigger condition which would not be met) and then never again (precisely because it *persists*, which precludes it happening again).

```
<io-mail.1>
  <alarm>
    <event>
      <mailto>john.doe@company.com</mailto>
      <subject>Start IO Server</subject>
    </event>
    ...
  </alarm>
</io-mail.1>
```

You can also use a slight variation of this technique to send status reports at regular intervals. In this example the *repeat* parameter causes the event to repeat at intervals of one hour as long as its trigger conditions are met (which by its nature they are).

```
<io-mail.1>
  <alarm>
    <event>
      <mailto>john.doe@company.com</mailto>
      <subject>Nothing particular happening</subject>
      <message>Just another hour gone by</message>
      <repeat>01:00:00</repeat>
    </event>
  </alarm>
</io-mail.1>
```

4.10.2 Examples for use of the serial port

For the following examples to work, the serial port must be in peripheral mode (not terminal mode), and for the last two you must also be in data mode (not command mode), i.e. <packet mode=“D“>, not „C“.

The first example illustrates the sending of serial data. X1 and X0 should in this example always be connected through a light barrier or a turnstile to the entry and exit door of a room so that in the variable N1 the persons entering the room can be counted and in N0 the persons leaving the room. DN (the difference between N1 and N0) then contains the number of persons who are actually in the room.

Using the following configuration, a message is sent as soon as there exactly ten persons in the room, and then again when another person enters. The third evevt causes the counters to be reset if the „number of persons“ becomes negative, which is obviously not a reasonable result..

```
<io-mail.1>
  <alarm>
    <event>
      <trigger op=“EQ“ var=“DN“ val=“10"/>
      <action var=“TX“ val=“there are now exactly ten.\r\n“/>
    </event>
    <event>
      <trigger op=“DELTA“ var=“N1" val=“1"/>
      <trigger op=“GT“ var=“DN“ val=“10"/>
      <action var=“TX“ val=“Here comes another one! „/>
      <action var=“TX“ val=“It’s getting crowded in here!\r\n“/>
    </event>
    <event>
      <trigger op=“LT“ var=“DN“ val=“0"/>
      <action var=“N0" val=“0"/>
      <action var=“N1" val=“0"/>
      <action var=“TX“ val=“Oops, counted wrong...\r\n“/>
    </event>
  </alarm>
</io-mail.1>
```

Take special note of the second event, which contains two actions with assignments to TX. This is in fact allowed, and causes the texts to be sent one after the other in the order they are found in the configuration data. Longer texts can be divided in this way into up to 15 sections (such as lines). Note also: As opposd to mail texts, serial outputs give you full control over each individual character. The line break in particular has to be explicitly indicated, in this case using the control characters „\r\n“ for CR, LF.

The next example actually consists of two parts. The first event contains a serial trigger condition which sets the flag M0 as soon as a packet with the text „Alarm“ arrives, and a key on input X0 allows M0 to then be reset. Just imagine that output Y0 controls an indicator light. Now assuming the two first events trigger Y0 directly instead of M0, you would be done already at this point. Instead, the three other events cause this lamp to flash (0.3 sec. on, 0.2 sec. off) as long as M0 is set.

```
<io-mail.1>
  <alarm>
    <!– M0 set/reset–>
    <event>
      <dtrigger substring=“alarm“ case=“0"/>
      <action var=“M0" val=“1"/>
    </event>
    <event>
      <trigger op=“EQ“ var=“X0" val=“1"/>
      <action var=“M0" val=“0"/>
    </event>
    <!– Y0 flashes as long as M0 is set –>
    <event>
      <trigger op=“EQ“ var=“M0" val=“1"/>
      <trigger op=“EQ“ var=“Y0" val=“0" delay=“0.2"/>
      <action var=“Y0" val=“1"/>
    </event>
    <event>
      <trigger op=“EQ“ var=“M0" val=“1"/>
      <trigger op=“EQ“ var=“Y0" val=“1" delay=“0.3"/>
      <action var=“Y0" val=“0"/>
    </event>
    <!– Y0 off when M0=0 –>
    <event>
      <trigger op=“EQ“ var=“M0" val=“0"/>
      <action var=“Y0" val=“0"/>
    </event>
  </alarm>
</io-mail.1>
```

The last example shows the linking of several serial triggers with each other and with a regular trigger condition. The *message* text does contain some peculiarities: When the mail is sent, the place holder %s is replaced with the contents of the data packet that generated the event, and the control character \c forces a line break. As was already mentioned above, a single period in the last line can prevent the device status from being appended. What is new is that the period doesn‘t have to be a message in itself if it is preceded by \n as shown here.

```
<io-mail.1>
  <alarm>
    <event>
      <!– three implicit AND-operated trigger conditions: –>
      <dtrigger substring=“alarm“/>
      <dtrigger substring=“ 51 „/>
      <trigger op=“EQ“ var=“NT“ val=“0"/>
      <mailto>j.doe@company.de</mailto>
      <subject>Alarm messzge from Station 51</subject>
      <message>The message text is:\n%s\n.</message>
    </event>
  </alarm>
</io-mail.1>
```

The final thing to note about the function of the <dtrigger> is that they trigger an event either immediately or never. This means that the two searched for texts must appear in one and the same packet in order to cause anything to happen. Hits appearing in multiple packets do not count, regardless of how short the time between the packets might be. And packets that contain the same two texts but which arrive during the night (i.e., if the variable NT is not equal to 0) likewise have no effect. Especially not, as one might assume, that a mail is sent as soon as night ends.

5 Restrictions

5.1 What doesn‘t work

- It is not possible to compare two variables with each other as a trigger condition, rather only one variable compared with a constant value. Only a comparison between the counters N0 and N1 is (indirectly) possible, namely for the contents of DN.
- As long as the serial port is in command mode, <packet mode=“C“/>, any serial trigger conditions specified by <dtrigger/> have no effect.
- The placeholder %s has an effect only in the *message* text of a mail. It is not possible to insert the received serial packet into the subject line of the mail or into serial outputs.

5.2 Limit values

Some of these numbers have already been covered elsewhere in the manual, so we will list them here only for overview purposes.

- In the present firmware version, a maximum of 15 events, 25 trigger conditions and 40 actions can be defined.
- The largest possible values for response delay and recovery time for a trigger condition is 00:10:55, or just under eleven minutes.
- Message texts are allowed to be no more than 254 character long (corresponding to a mail of approx. 3 - 4 lines). The contents of a mail may consist of multiple *mesage* blocks, but these texts are not seamlessly appended to each other, rather each one is represented as a separate paragraph. In addition, only 8 kB of memory is available for all the configuration data together.
- The serial outputs for an event may consist of max. 15 assignments to TX.
- The serial input buffer is 8 kB in size. A single serial packet exceeding this size is therefore by necessity truncated.

- The XML document with the configuration data is allowed to be no more than 20 kB long, regardless of whether it is submitted via mail or locally through the serial port. Comments are not counted, so you don't necessarily have to skimp here. In addition, however, there is a limitation of 64 kB total size for mails, and larger ones are deleted without reading. This limit is rather aribrarily set, but it is still necessary since the IO-Mailer could otherwise become blocked for long periods of time by huge file attachments, whether maliciously or by mistake.

5.3 Possible problems

- Since it is often possible to reach a mail recipient under multiple variations of one and the same address (e.g.„*john.doe@mail.company.com*“ as an equivalent for „*john.doe@company.com*“), it can happen that not all recipients are able to use the „*unsubscribe*“ command. This will fail for recipients who are entered in the configuration under a different address than the one their own mail software recognizes as a sender address..
- The XML document with the configuration data may not be larger than 20 KB; it doesn't matter whether it is uploaded by mail or locally through the serial port. But since comments are not counted, you do not necessarily have to limit yourself in this regard. Of course there is also a 64 KB overall limit for mails; larger mails are deleted without being read. This limit is necessary because otherwise the IO Mailer could be blocked for longer periods of time by enormous file attachments which are sent either maliciously or inadvertently.
- In contrast to the XML specification the IO Mailer does not understand the 16-bit character sets UTF-8 and UTF-16, but rather only the 8-bit character set ISO-8859-1 and its subset 7-bit US-ASCII. Since ISO-8859-1 is the usual character set in Western Europe under Windows and Unix, this limitation is normally not a problem as long as you use a simple text editor for editing the configuration data. But if you use a special XML editor you should be sure you don't

save the data as UTF-16. Whenever possible use ISO-8859-1, and resort to UTF-8 only in emergencies.

- The IO Mailer does not understand MIME-encoded base64 but rather only quoted-printable and, of course, simple text. A decent mail software program should have no reason to ever encode the configuration data as base64. But if this should happen the configuration data will be rejected with the error message „*start tag <io-mail.1> not found*“. In this situation try to get around your mail software by sending the configuration data not as a file attachment but rather by inserting it directly into the body of the mail text.

6. Appendix: Internet addresses

6.1 XML

General information re: XML (specifications, questions and answers):

http://www.w3.org/TR/REC-xml
http://msdn.microsoft.com/xml/tutorial/default.asp
http://www.ucc.ie/xml/
http://msdn.microsoft.com/workshop/xml/general/xmlfaq.asp

Free download: The Microsoft XML Validator (requires Internet Explorer 5). What you basically get is an HTML page containing JavaScript excerpts. This allows you to read-in the name of an XML file and forward it to the built-in routines of Explorer for checking. Very helpful when you want to check XML files for proper format before sending them to the IO Mailer.

http://msdn.microsoft.com/downloads/samples/internet/xml/xml_validator/default.asp

Free download: The Microsoft XML Notepad (also requires Internet Explorer 5). A nice toy, though in no way required for editing configuration data for the IO Mailer, which can be done just as well using a normal text editor.

http://msdn.microsoft.com/xml/notepad/intro.asp

6.2 Timeservers

You can find a directory of time servers accessible from anywhere in the world (strictly speaning NTP servers) at http://www.eecis.udel.edu/~mills/ntp/servers.htm.

NTP (*N*etwork *T*ime *P*rotocol) is a cumbersome protocol for the most precise possible clock synchronization in extended networks (see RFG-1305), not to be confused with the RFC-868 time service which the IO Mailer requires. In practice most NTP servers include the simple time stamp service. Note however that you should use only servers in this list which are expressly designated „Open access“.

Free download: Time server for Windows computers. A local solution like this makes more sense for most purposes than resorting to an external time server .

http://www.arachnoid.com/abouttime/index.html

As already mentioned above you should have no need for such software if you have your own Unix or Linux server. These already have an RFC-868 time server built-in; you just need to make sure that it is also activated („*time*“ service in the „*inetd.conf*“ file).Technische Daten Typen 57111, 57211

7 Firmware Update

The IO-Mailer is divided internally into two function areas TCP/IP-Ethernet and SMTP/POP3-Client.

Each of the two function areas has its own self-contained firmware which can be updated independently.

7.1 Updating the TCP/IP-Ethernet area

7.1.1 Under Windows 9x/NT/2000

Required is a PC running under Windows with a network connection and an activated TCP/IP stack. You will need 3 files for the update process, all of which can be obtained from the Web site **http://www.wut.de** for downloading.

- The executable update tool for sending the firmware to the IO-Mailer.
- The file with the new firmware to be sent to the IO-Mailer.
- The configuration file (*.cfg), which sets the parameters in the TCP/IP stack as they need to be for the IO-Mailer.

The update process is described in sequential steps below. Please note any additional notes which may be found on our Web pages.

1. Start a Telnet session on the configuration port of the Com-Server.

   ```
   telnet [IP address] 1111
   ```

   Open *SETUP: System ® Flash Update ® Net Update,* and confirm with *„y“*. The network connection is then closed by the IO-Mailer and the corresponding Telnet message must be confirmed with *„OK“*. The green status LED indicates that you are now in update mode.
2. Start the update tool. Open *CS programming ® Flash* to get to the input screen for updating new firmware.
3. Enter the IP address of the IO-Mailer in the corresponding fields as well as the name of the firmware

file and the configuration file. In the option field *„Output“* activate only the item *„Firmware“*.

4. Click on the *Start* button. The update will take several seconds. It is not finished until a message window indicates the end of the update process.
5. Check in the IO-Mailer configuration file to verify that the new operating software has been installed. The new version number for the firmware must be displayed in *INFO System ® SOFTW Date/Rev*.

If the previous version is still displayed, the file with the new operating software is corrupted. Please contact your dealer in this event.

7.1.2 Under UNIX

Required is a computer with a network connection and a TCP/IP stack which provides the network protocols Telnet and TFTP.

To update using TFTP protocol, you can upon request get the firmware from W&T (info@wut.de) split up into 4 files which are sent to the IO-Mailer in sequence.

1. Use Telnet to start the remote configuration tool of the IO-Mailer.

   ```
   telnet [IP address] 1111
   ```

   Open *SETUP: System ® Flash Update ® Net Update*, and confirm with *„y“*. The network connection is then closed by the IO-Mailer and the corresponding Telnet message must be confirmed with *„OK“*. The green status LED indicates that you are now in update mode.
2. Use the TFTP command to send the first file in binary mode to the IO-Mailer. While the data are being sent over the network, the Status LED will be on. Then the IO-Mailer goes into programming mode and the Error LED comes on. This process may take several seconds. Wait until the Error LED goes out and the Status LED comes back on.
   Repeat this process for all four files.
3. The IO-Mailer recognizes when all the files have been transmitted and automatically restarts itself. If the green Status LED comes on again after all the files have been

sent, repeat step 2 in full. The duplicate transmission of a file does not result in an update error. The IO-Mailer waits as long as it takes for all the necessary files to be sent.

Example: SCO UNIX
Enter the following commands after the corresponding prompt:

```
# tftp
tftp> connect [ip_number|host_name]
tftp>  binary
tftp>  put C4r1_1.4_1 [remote filename]
       (remote filename = any letter)
```

Now wait until the green Status LED comes on again. Then use the command *put* to send the remaining three files and quit the TFTP connection.

```
tftp>  quit
#
```

4. Check in the IO-Mailer configuration file to verify that the new operating software has been installed. The new version number for the firmware must be displayed in *INFO System ® SOFTW Date/Rev*.

If the previous version is still displayed, the file with the new operating software is corrupted. Please contact your dealer in this event.

7.2 Updating the SMTP/POP3-Client area

For the update process you will need an update file (*.mhx), which you can download from **http://www.wut.de**.

7.2.1 Updating over the network

1. Open the mhx-file in an editor.
2. Copy the complete contents to the clipboard (*Edit ® Select All ® Edit ® Copy*).

3. Now you must place the IO-Mailer in update mode. There are two ways to do this:
   - Sending an e-mail to the IO-Mailer with „Update“ as the subject.
   - Connecting the serial port of the IO-Mailer to a COM port on your PC.
   - Open a terminal program with 9600 baud, 8 bits, no Parity, no Handshake.
   - Place the IO-Mailer in terminal mode by typing several small „x“ characters while turning on the power, until IP+Enter appears in the terminal pro gram. Press the Enter key and then the # key. The following will appear:

     ```
     MB90F562 bootloader v1.6 W&T 06/2001
     Invoked by software, ESC to cancel
     Waiting (Port 1)...
     ```

4. Start a Telenet session to the IO-Mailer

   ```
   telnet [IP address]
   ```

5. Now insert the contents of the clipboard under Telnet (*Edit* ® *Insert*)

   The following will appear in order:

   ```
   Programming...
   48074
   Done!
   ```

After loading the new firmware, the IO-Mailer automatically ends the Telnet session.

7.2.2 Updating through the serial port

1. Open the mhx-file in an editor.
2. Copy the complete contents to the clipboard (*Edit* ® *Select All* ® *Edit* ® *Copy*).
3. Now you must place the IO-Mailer in update mode. Connect the serial port on the IO-Mailer to a COM port on your PC.

Open a terminal program with 9600 baud, 8 bits, not parity, handshake XON/XOFF.
Place the IO-Mailer in terminal mode by typing several small „x“ characters while turning on the power, until IP+Enter appears in the terminal program. Press the Enter key and then the # key.

Press the Enter key and then the # key.

The following appears:

```
MB90F562 bootloader v1.6 W&T 06/2001
Invoked by software, ESC to cancel
Waiting (Port 0)...
```

4. Now insert the contents of the clipboard under Telnet (*Edit* ® *Insert*)

   The following will appear in order:

```
Programming...
48074
Done!
```

The firmware update is now completed.

Technical data 57111, 57211

	57111	57211
Supply voltage	5V by wall plug AC/DC adaptor 230V/50Hz	
Current draw	typi. 270mA max. 330mA	typ. 520mA max. 630mA
Configuration Interface	RS232 DCE-pinout on DB9F 9600Bd, 8 Bit, no Parity (only RxD und Txd supported)	
Digital inputs	2 inputs, voltage resistent up to +/-30V CMOS-compatible trigger 32 Bit counter (max. +/-2147483647)	
Digital outputs	2 outputs, open collector max. 30V/100mA	
Network	Ethernet 10BaseT	Ethernet 100BaseTx/10BaseT
Mail functions	Configuration Alarm, State Seting outputs	
Mail protocols	POP3, SMTP	
Housing	Plastic compact housing, 105x75x22mm for DIN rail mounting	
Weight	approx. 200g	
Operating ambient	0..+50°C strung 0..+60°C single	
Storage ambiente	-40..+70°C	

subject to errors and alternations
Wiesemann & Theis GmbH

Technical data Typen 57511, 57611

	57511	57611
Supply voltage	12–24V AC or DC through screw terminal	
Current draw	typ. 155mA @24V max. 205mA @24V	typ. 175mA @24 max. 225mA @24V
Configuration Interface	RS232 DCE-pinout on DB9F 9600Bd, 8 Bit, no Parity (only RxD und Txd supported)	
Digital inputs	2 inputs, voltage resistent up to +/-30V CMOS-compatible trigger 32 Bit counter (max. +/-2147483647)	
Digital outputs	2 outputs, open collector max. 30V/100mA	
Network	Ethernet 10BaseT	Ethernet 100BaseTx/10BaseT
Mail functions	Configuration Alarm, State Seting outputs	
Mail protocols	POP3, SMTP	
Housing	Plastic compact housing, 105x75x22mm for DIN rail mounting	
Weight	approx. 200g	
Operating ambient	0..+50°C strung 0..+60°C single	
Storage ambiente	-40..+70°C	

subject to errors and alternations
Wiesemann & Theis GmbH

EC Declaration of Conformity

In accordance with Article 10.1 of the Directive 89/336/EWG

Wiesemann & Theis GmbH, Wuppertal hereby declares that the products

W&T IO-Mailer	Model 57111, Model 57211
W&T IO-Mailer	Model 57511, Model 57611

to which this declaration refers are in compliance with the following standards and normative documents:

1. EN 55022-B (1998)
   Product-specific emissions standard
   for Communications Technology

2. EN 50082-2 (1996) Noise Immunity
   2.1. EN 61000-4-2 ESD
   2.2. EN 61000-4-3 EMF Electromagnetic Fields Immunity
   2.3. EN 61000-4-4 Burst
   2.4 EN 61000-4-5 Surge
   2.5. EN 61000-4-6 Immunity to Conducted Disturbances,
   Induced by RF
   2.6. EN 61000-4-8 Power Frequency Magnetic Field
   Immunity (not applicable)
   2.7. EN 61000-4-11 Voltage Dips, Short Interruptions, and
   Voltage Variations Immunity

3. EN 60950 (1997)
   Product-specific Low Power Directive for Communications Technology

Wuppertal, September 13, 2001

[redacted]

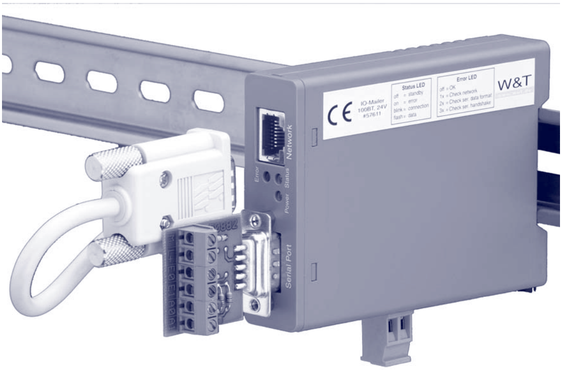
Klaus D. Meyer
Sales Manager

Dipl.-Ing. Rüdiger Theis
CEO

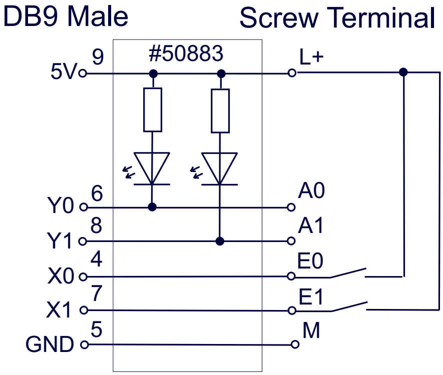
Appendix: Optional accessories

Screw terminal adapter part number 50883

A screw terminal adapter model 50883 is available for easily connecting digital signals.



Wiring example:


DB9 Male
Screw Terminal
#50883
5V
9
Y0
6
Y1
8
X0
4
X1
7
GND
5
L+
A0
A1
E0
E1
M