Micriμm

Empowering Embedded Systems

μC/SMTPc

V1.01

User's Manual

www.Micrium.com

Disclaimer

Specifications written in this manual are believed to be accurate, but are not guaranteed to be entirely free of error. Specifications in this manual may be changed for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, Micrium assumes no responsibility for any errors or omissions and makes no warranties. Micrium specifically disclaims any implied warranty of fitness for a particular purpose.

Copyright notice

You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of Micrium. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2006 Micriμm, Weston, Florida 33327-1848, U.S.A.

Trademarks

Names mentioned in this manual may be trademarks of their respective companies. Brand and product names are trademarks or registered trademarks of their respective holders.

Registration

Please register the software via email. This way we can make sure you will receive updates or notifications of updates as soon as they become available. For registration please provide the following information:

- Your full name and the name of your supervisor
- Your company name
- Your job title
- Your email address and telephone number
- Company name and address
- Your company's main phone number
- Your company's web site address
- Name and version of the product

Please send this information to: **licensing@micrium.com**

Contact address

Micrium
949 Crestview Circle
Weston, FL 33327-1848
U.S.A.
Phone : +1 954 217 2036
FAX : +1 954 217 2037
WEB : **www.micrium.com**
Email : **support@micrium.com**

Manual versions

If you find any errors in this document, please inform us and we will make the appropriate corrections for future releases.

Manual Version	Date	By	Description
V1.00	2006/02/01	SR	First version.
V1.01	2006/04/25	SR	Code modifications

Table Of Contents

Introduction

SMTP (Simple Mail Transfer Protocol) is a protocol designed to transfer mail reliably and efficiently. When an SMTP client has a message to transmit, it establishes a two-way transmission channel to an SMTP server. The responsibility of an SMTP client is to transfer mail messages to a SMTP server, or report its failure to do so [RFC 2821].

µC/SMTPc is an add-on product to µC/TCP-IP that implements the client SMTP protocol. µC/SMTPc implements part of the following RFC:

RFC 2821	**ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt**
RFC 2822	**ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt**

This document describes how to configure and use the µC/SMTPc module on top of µC/TCP-IP in a µC/OS-II environment. A Cogent CSB337 (ARM9) development platform with IAR compiler is used to demonstrate the typical application of this module, but other platforms and tool chains may be used as well.

Required modules version

The current version of the µC/SMTPc module has been developed and tested using version **1.84** of µC/TCP-IP.

Chapter 2

Directories and Files

The code and documentation of the µC/SMTPc module are organized in a directory structure according to "AN-2002, µC/OS-II Directory Structure". Specifically, the files are found in the following directories:

`\Micrium\Software\uC-SMTPc`
: This is the main directory for µC/SMTPc.

`\Micrium\Software\uC-SMTPc\Doc`
: This directory contains the µC/SMTPc documentation files, including this one.

`\Micrium\Software\uC-SMTPc\Cfg\Template`
: This directory contains a template of µC/SMTPc configuration.

`\Micrium\Software\uC-SMTPc\Source`
: This directory contains the µC/SMTPc source code. This protocol is implemented in four files:

```
smtp-c.c
smtp-c.h
smtp-c_ADT.c
smtp-c_ADT.h
```

Note that the '-c' at the end of `smtp` stands for client and thus contains 'client' side code. `smtp-c.h` is a header file containing client declarations for SMTP. Finally, `smtp-c_ADT.c` and `smtp-c_ADT.h` contain the abstract data types (ADT) used thourough this module.

Chapter 3

Using µC/SMTPc

This chapter provides examples on how to use µC/SMTPc. A Cogent CSB337 (ARM9) running µC/OS-II and µC/TCP-IP was used to demonstrate its application, as illustrated in figure 3-1.

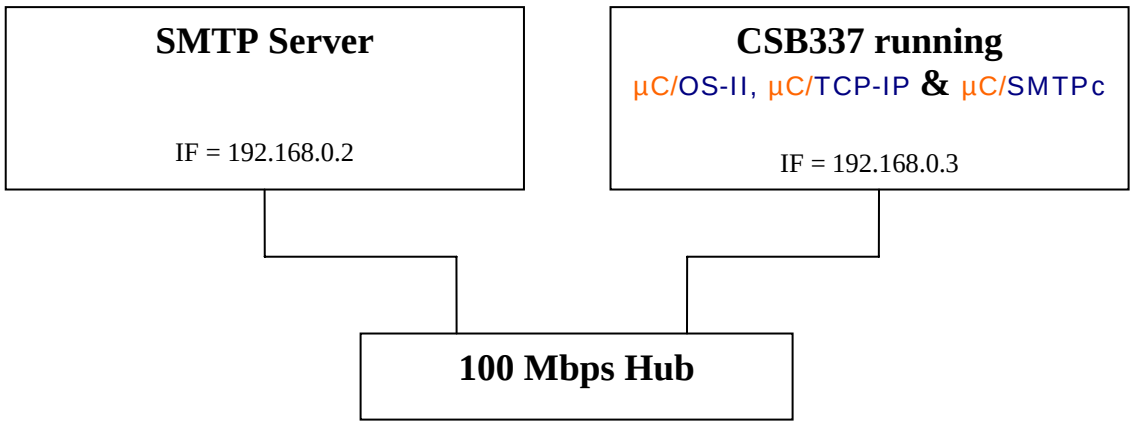

Figure 3-1, Test setup

3.01 Test code

The code in the next section assumes knowledge of µC/OS-II and µC/TCP-IP. Indeed, this section of code only concerns µC/SMTPc and you need to be able to configure the real-time OS and the TCP stack in order to run it.

3.02 Test code, SMTPc_Test()

Listing 3-1 is shown to demonstrate the µC/SMTPc module capabilities. This code sequentially calls every exported functions of this module is order to illustrate its typical use.

Listing 3-1

```
#define  BODY_MSG_LEN           50


void SMTPc_Test (void)
{
    NET_IP_ADDR  ip_server;
    NET_IP_ADDR  client_addr;
    NET_ERR      err_msg;
    NET_SOCK_ID  sock;
    SMTPc_MSG    msg;
    SMTPc_MBOX   from;
    SMTPc_MBOX   to_1;
    SMTPc_MBOX   to_2;
    SMTPc_MBOX   cc_1;
    CPU_CHAR     body_msg[BODY_MSG_LEN];


    ip_server = NetASCII_Str_to_IP("192.168.0.2", &err_msg);                          (1)
    if (err_msg != NET_ASCII_ERR_NONE) {
        APP_DEBUG_TRACE("Error - NetASCII_Str_to_IP: %d\n", err_msg);
        return;
    }

    client_addr = NetASCII_Str_to_IP("192.168.0.3", &err_msg);
    if (err_msg != NET_ASCII_ERR_NONE) {
        APP_DEBUG_TRACE("Error - NetASCII_Str_to_IP: %d\n", err_msg);
        return;
    }

    sock = SMTPc_Connect(ip_server, 0, client_addr, DEF_NO, &err_msg);                (2)
    if (err_msg != SMTPc_ERR_NONE) {
        APP_DEBUG_TRACE("Error - SMTPc_Connect: %d\n", err_msg);
        return;
    }

    SMTPcADT_SetSMTPcMbox(&from, "John Doe",     "John.Doe@foobar.com",  &err_msg);   (3)
    SMTPcADT_SetSMTPcMbox(&to_1, "Jane Doe",     "Jane.Doe@foobar.com",  &err_msg);
    SMTPcADT_SetSMTPcMbox(&to_2, "System admin", "sys_admin@foobar.com", &err_msg);
    SMTPcADT_SetSMTPcMbox(&cc_1, NULL,           "archive@foobar.com",   &err_msg);


    SMTPcADT_InitSMTPcMsg(&msg, &from, NULL, &err_msg);                               (4)
    if (err_msg == SMTPc_ERR_NONE) {

        msg.ToArray[0] = &to_1;                                                       (5)
        msg.ToArray[1] = &to_2;
        msg.CCArray[0] = &cc_1;

        Str_Copy(msg.Subject, "Test message")                                         (6)

        Str_Copy(body_msg, "This is a test, everything is working fine");             (7)
        msg.ContentBodyMsg    = body_msg;
        msg.ContentBodyMsgLen = Str_Len(body_msg);

        SMTPc_SendMsg(sock, &msg, &err_msg);                                          (8)
        if (err_msg != SMTPc_ERR_NONE) {
            APP_DEBUG_TRACE("Error - SMTPc_SendMsg: %d\n", err_msg);
        }
    } else {
        APP_DEBUG_TRACE("Error - SMTPcADT_InitSMTPMail: %d\n", err_msg);
    }

    SMTPc_Disconnect(sock, &err_msg);                                                 (9)
}
```

L3-1(1) Convert the ASCII dotted-decimal notation to a network protocol IPv4 address. `ip_server` is the address of the SMTP server and `client_addr` is the IP address of the device running this module.

L3-1(2) Establish a TCP connection to the SMTP server (session initiation), using the default port (`SMTPc_CFG_IPPORT`, see section 3.03), and initiate client using HELO (non extended).

L3-1(3) Prepare the `SMTPc_MBOX` structure (copy names and addresses for the various recipients and sender).

L3-1(4) Initialize the `SMTPc_MSG` structure (the message) and set the sender (mandatory "from" field and optional "sender").

L3-1(5) Once the message has been initialized, the recipients have to be inserted into the `SMTPc_MSG` structure. It is important to start at the first position of the arrays (index "`0`") and not to insert the recipients sequentially. When sending a message, this module presumes no more recipient is present when the first NULL value is encountered in a given `SMTPc_MSG` array.

The length of the various arrays are user configured (see section 3.03).

L3-1(6) Copy the subject of the message into the `SMTPc_MSG` structure. The length of the subject array is also user defined (see section 3.03).

L3-1(7) Set the pointer to the message body in the `SMTPc_MSG` structure, as well as the length of the data to be sent.

Note here that **it is the application's responsibility to respect the 1000 character line limit** imposed by the RFC 2821. In order to do so, each line of the message body MUST be terminated by the sequence `<CRLF>` (ASCII characters 13 and 10). You might want to use the defined constant `CRLF` for this purpose (see `smtp-d.h` for more details).

L3-1(8) Send the message to the SMTP server. If something goes wrong, the value of `err_msg` is set accordingly, the mail transfer is automatically aborted, and the server is reset (it is then waiting for another call to `SMTPc_SendMsg`).

L3-1(9) Close the session between the client and the SMTP server, and terminate the TCP connection.

3.03 µC/SMTPc module configuration

The µC/SMTPc module has to be configured according to your specific needs. A template configuration file (smtp-c_cfg.h) is included in the module package (see Chapter 2, Directories and Files), and this configuration should be copied into your app_cfg.h file. Note that future releases of this module may include more configuration options.

Here are the customizable variables:

```
#define  SMTPc_CFG_IPPORT                25
```

This value sets the default port to use when calling SMTPc_Connect() without specifying any particular port. Standard listening port for SMTP servers is 25.

```
#define  SMTPc_CFG_MBOX_NAME_DISP_LEN    50
```

This value corresponds to the maximum length of the displayed name associated with a mailbox, including '\0'. This length MUST be smaller than 600 in order to respect the Internet Message size limit (see RFC 2821 for more details on this limit). This length has a direct impact on the SMTPc_MBOX structure size, and as a rule of thumb, it should be as small as possible in order to preserve memory.

```
#define  SMTPc_CFG_MSG_SUBJECT_LEN       50
```

This sets the maximum length of the string containing the mail subject, including '\0'. This value MUST be smaller than 900, for the same reason cited above.

```
#define  SMTPc_CFG_MSG_MAX_TO             5
#define  SMTPc_CFG_MSG_MAX_CC             5
#define  SMTPc_CFG_MSG_MAX_BCC            5
#define  SMTPc_CFG_MSG_MAX_ATTACH         5
```

Size of the various arrays inside the SMTPc_MSG structure. For instance, a size of '5' for SMTPc_CFG_MSG_MAX_TO would allow the a message be built and sent to a maximum of 5 recipients in the “To: “ header field of the mail object. Since pointer are used in the structure, having arrays bigger than needed does not influence greatly the memory usage of the application.

3.04 µC/ SMTPc module limitations

This SMTP client implements a part of RFC 2821; not all commands have been implemented (see smtp-c.h for more details). For instance, the current release does not support the extended EHLO client initiation (only the traditionnal HELO is used).

As mentioned in section 3.02 (L3-1(7)), the µC/SMTPc module does not prevent excessive line length (line longer than 1000 characters, including <CRLF>) in the body content of a message. It is the application responsibility to make sure the message sent (member ContentBodyMsg of structure SMTPc_MSG) conform to this limitation.

When sending messages, the name of the mailbox's owner (field `NameDisp` of structure `SMTPc_MBOX`) is not transmitted.

MIME encoding is not currently supported; messages must then contain US-ASCII (7 bits) characters only.

Finally, sending attachments is not currently offered to the client application of this module.

3.05 µC/SMTPc memory requirements

µC/SMTPc only has one global variable in RAM: `SMTPc_Comm_Buf`. The amount of memory needed by this variable corresponds to the value of `SMTPc_COMM_BUF_LEN`. See the file `smtp-c.h` for more details.

As for the calling task's stack size, it should be larger than 104 bytes; a smaller stack could introduce run-time problems.

Chapter 4

µC/SMTPc API Reference

This chapter provides a reference to the µC/SMTPc API. Each of the user-accessible services is presented in alphabetical order. The following information is provided for each of those services:

- A brief description
- The function prototype
- The filename of the source code
- A description of the arguments passed to the function
- A description of the returned value(s)
- Specific notes and warnings on using the service

SMTPc_Connect()

```
void SMTPc_Connect(NET_IP_ADDR  ip_server,
                   CPU_INT16U   port,
                   NET_IP_ADDR  client_addr,
                   CPU_BOOLEAN  init_extended,
                   NET_ERR     *perr);
```

File	Called from
SMTP-C.C	Application

SMTPc_Connect() establishes a TCP connection to the SMTP server and initiate the SMTP session.

Arguments

ip_server	IP address of the SMTP server to contact.	
port	TCP port to use. If "0", SMTPc_DFLT_PORT is used.	
client_addr	Address literal helping identifying the client system.	
init_extended	Whether of not to attempt an extended session initialization (EHLO).	
perr	Pointer to a variable that will hold the return error code from this function, which can be any of the following:	
	SMTPc_ERR_NONE	No error, TCP connection established.
	SMTPc_ERR_SOCK_OPEN_FAILED	Error opening socket.
	SMTPc_ERR_SOCK_CONN_FAILED	Error connecting to server.
	SMTPc_ERR_RX_FAILED	Error receiving server reply.
	SMTPc_ERR_REP	Error with reply.

Returned Values

Socket descriptor if no error; -1 otherwise.

Notes/Warnings

1. If anything goes wrong while trying to connect to the server, the socket isclosed by calling NetSock_Close(). Hence, all data structures are returned to their original state in case of a failure to established the TCP connection. If the failure occurs when initiating the session, the application is responsible of the appropriate action(s) to be taken.
2. The server will send a 220 "Service ready" reply when the connection is completed. The SMTP protocol allows a server to formally reject a transaction while still allowing the initial connection by responding with a 554 "Transaction failed" reply.
3. In the current implementation, the extended session initialization (using EHLO) is not supported. The session is hence established using HELO independently of the init_extended argument value.

SMTPc_Disconnect()

```
void SMTPc_Disconnect(NET_SOCK_ID  sock,
                      NET_ERR     *perr);
```

File	Called from
SMTP-C.C	Application

SMTPc_Disconnect() closes the connection between the client and the server.

Arguments

`sock` Socket ID returned by `SMTPc_Connect()`.

`perr` Pointer to a variable that will hold the return error code from this function, which can be any of the following:

`SMTPc_ERR_NONE` No error.

Returned Values

void.

Notes/Warnings

1. The receiver (client) MUST NOT intentionally close the transmission channel until it receives and replies to a QUIT command.
2. The receiver of the QUIT command MUST send an OK reply, and then close the transmission channel.

SMTPc_SendMsg()

```
void SMTPc_SendMsg(NET_SOCK_ID  sock,
                   SMTPc_MSG   *msg,
                   NET_ERR     *perr);
```

File	Called from
SMTP-C.C	Application

SMTPc_SendMsg() sends a message (an instance of the SMTPc_MSG structure) to the SMTP server.

Arguments

sock	Socket ID returned by SMTPc_Connect().	
msg	SMTPc_MSG structure encapsulating the message to send.	
perr	Pointer to a variable that will hold the return error code from this function, which can be any of the following:	
	SMTPc_ERR_NONE	No error.
	SMTPc_ERR_NULL_ARG	Mandatory argument(s) missing.
	SMTPc_ERR_RX_FAILED	Error receiving server reply.
	SMTPc_ERR_REP	Error with reply.
	SMTPc_ERR_TX_FAILED	Error querying server.
	SMTPc_ERR_LINE_TOO_LONG	Line limit exceeded.

Returned Values

void.

Notes/Warnings

1. The function SMTPcADT_InitSMTPcMsg() has to be called before being able to send a message.
2. The message has to have at least one receiver, either "To", "CC", or "BCC".

Appendix A

µC/SMTPc Licensing Policy

You need to obtain an 'Object Code Distribution License' to embed µC/SMTPc in a product that is sold with the intent to make a profit. Each 'different' product (i.e. your product) requires its own license but, the license allows you to distribute an unlimited number of units for the life of your product. Please indicate the processor type(s) (i.e. ARM7, ARM9, MCF5272, MicroBlaze, Nios II, PPC, etc.) that you intend to use.

For licensing details, contact us at:

Micrium
949 Crestview Circle
Weston, FL 33327-1848
U.S.A.

Phone : +1 954 217 2036
FAX : +1 954 217 2037

WEB : **www.micrium.com**
Email : **licensing@micrium.com**

Appendix B

References

µC/OS-II, The Real-Time Kernel, 2nd Edition
Jean J. Labrosse
CMP Books, 2002
ISBN 1-57820-103-9