µC/DHCPc™

The Embedded DHCP Client

User's Manual
V2.08.01

Micriµm

For the Way Engineers Work

Micriμm
1290 Weston Road, Suite 306
Weston, FL 33326
USA

www.Micrium.com

The programs and code examples in this book are presented for instructional value. The programs and examples have been carefully tested, but are not guaranteed to any particular purpose. The publisher does not offer any warranties and does not guarantee the accuracy, adequacy, or completeness of any information herein and is not responsible for any errors and ommissions. The publisher assumes no liability for damages resulting from the use of the information in this book or for any infringement of the intellectual property rights of third parties that would result from the use of this information.

600-uC-DHCPc-002

Micriμm
For the Way Engineers Work

USER'S 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
V2.00	2008/10/17	SR	Released first version
V2.01	2008/11/07	SR	DHCPc API Reference updated
V2.02	2009/01/14	SR	
V2.03	2009/03/12	SR	
V2.04	2009/07/03	SR	
V2.05	2009/08/11	SR	
V2.06	2010/01/04	SR	
V2.07	2010/01/27	ITJ	
V2.08	2010/03/11	SR	
V2.08a	2010/10/15	AA/ITJ	Converted document from Word to FrameMaker
V2.08.01	2011/03/02	SL	Changed font of functions in the API Reference chapter

Table of Contents

Chapter 1 Introduction

DHCP is a protocol designed to enable clients to get IP configuration from a centralized database. This protocol has slightly evolved over the years from the BOOTP protocol initially designed to enable diskless clients to boot from the network. The µC/DHCPc module implements part of the following RFCs:

RFC 2131	ftp://ftp.rfc-editor.org/in-notes/rfc2131.txt
RFC 2132	ftp://ftp.rfc-editor.org/in-notes/rfc2132.txt
RFC 3927	ftp://ftp.rfc-editor.org/in-notes/rfc3927.txt

The first two describe the DHCP mechanism, and the third explains the dynamic configuration of link-local addresses (sometimes referred as Automatic Private IP Addressing, APIPA, or AutoNet in other environments).

This document describes how to configure and use the µC/DHCPc module in a µC/TCP-IP and µC/OS-II or µC/OS-III environment.

Chapter

2

Directories and Files

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

\Micrium\Software\uC-DHCPc-V2
This is the main directory for µC/DHCPc.

\Micrium\Software\uC-DHCPc-V2\Doc
This directory contains all µC/DHCPc documentation files, including this user's manual.

\Micrium\Software\uC-DHCPc-V2\Cfg\Template
This directory contains a template of µC/DHCPc configuration.

\Micrium\Software\uC-DHCPc-V2\Source
This directory contains the µC/DHCPc source code. This protocol is implemented in two OS independent files:

```
dhcp-c.c
dhcp-c.h
```

Note that the '-c' after 'dhcp' stands for client and thus contains 'client' side code.

```
\Micrium\Software\uC-DHCPc-V2\OS\uCOS-II
                                \uCOS-III
```

This is where operating system (OS) dependent code is located. µC/DHCPc is distributed with ports for µC/OS-II and µC/OS-III. Note that it would be possible to use µC/DHCP with other operating systems by developing appropriate dhcp-c_os.* implementation files.

REQUIRED MODULES

µC/DHCPc V2 requires the µC/CPU, µC/LIB, and µC/TCP-IP V2 modules. Please refer to the µC/DHCPc V2 release notes document for required version information.

Chapter

3

Using µC/DHCPc

3-1 µC/DHCPc MODULE USE

µC/DHCPc MODULE INTERNAL

Figure 3-1 below illustrates the internal operations this implementation of the DHCP client performs. This figure explains the various transitions the DHCP client task goes through while managing an interface's lease.

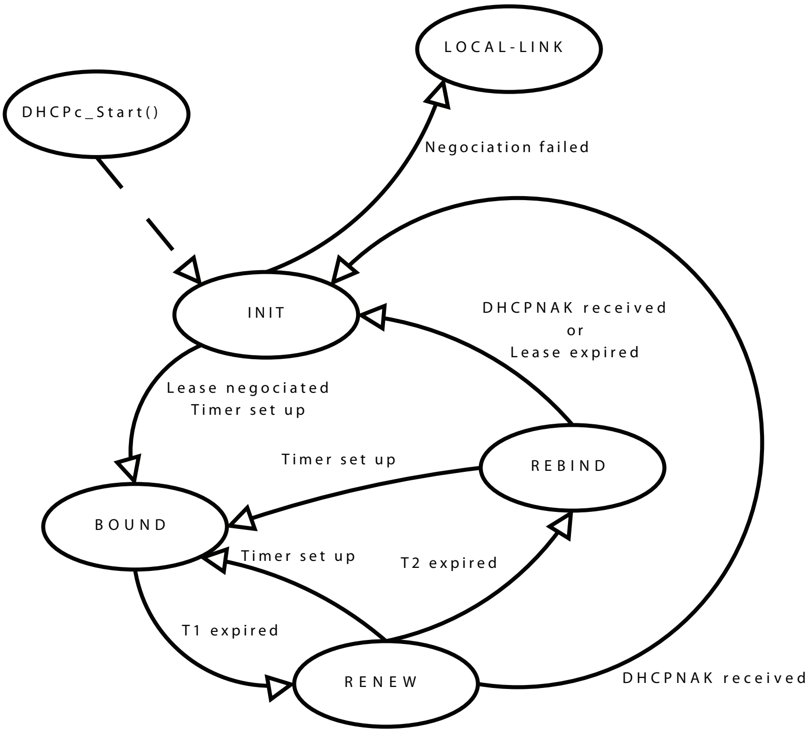

Figure 3-1 **µC/DHCPc internal**

DHCP BROADCAST AND UNICAST MESSAGES

In order to work with DHCP the TCP/IP software should accept and forward to the IP layer any IP packets deliveredto the target’s hardware address even before the IP address is configured. However, some TCP/IP stacks, such as µC/TCP-IP, are not able to do achieve this.

Therefore, when µC/DHCPc is used with µC/TCP-IP, the `DHCPc_CFG_BROADCAST_BIT` configuration parameter must be set to `DEF_ENABLED` to have the module work properly.

3-2 µC/DHCPc CONFIGURATION

The µC/DHCPc module has to be configured according to your specific needs. A template configuration file (`dhcp-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. Here is the list of the values and description of each of the configuration variable. However, keep in mind that future releases of this module might include more configuration options.

3-2-1 MODULE CONFIGURATION

```
#define  DHCPc_CFG_IP_PORT_SERVER                           67
#define  DHCPc_CFG_IP_PORT_CLIENT                           68
```

Define respectively the UDP port µC/DHCPc will send requests to and the one used to receive replies. Those default values are the ones specified in RFC #2131.

```
#define  DHCPc_CFG_MAX_RX_TIMEOUT_MS                      5000
```

Value for the socket receive timeout on each reception attempt.

```
#define  DHCPc_CFG_BROADCAST_BIT_EN                DEF_ENABLED
```

Whether or not to instruct the server to use broadcast when sending replies. This must be set to `DEF_ENABLED` when used with µC/TCP-IP.

```
#define  DHCPc_CFG_MAX_NBR_IF                                2
```

Defines the maximum number of interface that will be managed by the µC/DHCPc module at any time.

```
#define  DHCPc_CFG_ADDR_VALIDATE_EN               DEF_ENABLED
```

Whether or not the DHCP client should perform a final check prior to use this address in order to make sure it is not being used by another host on the network. As suggested in RFC 2131, this check uses an ARP broadcast. Note that you will need µC/TCP-IP V1.88 or higher in order to use this functionnality.

```
#define  DHCPc_CFG_DYN_LOCAL_LINK_ADDR_EN         DEF_ENABLED
```

Whether or not the dynamic link-local configuration mechanism is used in the case where a DHCP server could not be contacted.

```
#define  DHCPc_CFG_LOCAL_LINK_MAX_RETRY                    3
```

Maximum number of link-local configuration retries when the selected address is already in use on the network.

```
#define  DHCPc_CFG_ARG_CHK_EXT_EN                 DEF_DISABLED
#define  DHCPc_CFG_ARG_CHK_DBG_EN                 DEF_DISABLED
#define  DHCPc_DBG_CFG_MEM_CLR_EN                 DEF_DISABLED
```

These defines determine whether the argument check feature is enabled or not, as well as the memory clearfeature. This is a convenient faature while debugging, and should be set to `DEF_DISABLED` for production code.

3-2-2 OPERATING SYSTEM CONFIGURATION

The following configuration constants relate to the µC/DHCPc OS port. For many OSs, the µC/DHCPc task priority and stack size will need to be explicitly configured for the particular OS (consult the specific OS's documentation for more information).

For µC/OS-II and µC/OS-III, the following macros must be configured within `app_cfg.h`:

```
#define  DHCPc_OS_CFG_TASK_PRIO                           13
#define  DHCPc_OS_CFG_TMR_TASK_PRIO                       14
```

Values of the priority for the two µC/DHCPc tasks. The values assigned depend upon the software architecture of your system, and on the importance of this module's response time relative to other tasks.

```
#define  DHCPc_OS_CFG_TASK_STK_SIZE                          512
#define  DHCPc_OS_CFG_TMR_TASK_STK_SIZE                      512
```

Values of the stack sizes, in number of stack-sized words, for the two µC/DHCPc tasks. These default values should be sufficient for most environments, but you should check this on your system for acceptable reliability or performance.

3-3 INTERFACE WITH RTOS

µC/DHCPc requires the presence of a Real Time Operating System (RTOS). As mentioned in Chapter 2, µC/DHCPc is delivered with ports for µC/OS-II and µC/OS-III, but it is possible for µC/DHCPc to be used with another RTOS by providing appropriate implementations of dhcp-c_os.*.

3-4 µC/DHCPc EXAMPLE CODE

The file app.c contains an example of application code, and was written to illustrate the capabilities of the µC/DHCPc module. That code simply initializes µC/OS-II or µC/OS-III, µC/TCP-IP and µC/DHCPc, and creates a few tasks and other kernel objects that will give the user information about the state of the system.

Some sections of the source code have been removed or modified to help focus on the µC/DHCPc module use.

Listing 3-1 **Example code**

```
#define  APP_CFG_DHCP_NBR_IF_CFGD

static  CPU_BOOLEAN  AppInit_DHCPc (void)
{
    CPU_INT08U      nbr_if_started;
    NET_IF_NBR      if_nbr_cur;
    DHCPc_OPT_CODE  req_param[DHCPc_CFG_PARAM_REQ_TBL_SIZE];
    CPU_INT08U      req_param_qty;
    CPU_INT08U      nbr_if_init;
    CPU_BOOLEAN     if_dhcp_init_tbl[APP_CFG_DHCP_NBR_IF_CFGD];
    CPU_INT08U      if_done_ix;
    DHCPc_STATUS    status;
    DHCPc_ERR       err;


    err = DHCPc_Init();                                                                       (1)
    if (err == DHCPc_ERR_NONE) {
        printf("DHCP client successfully initialized\n\r");
     } else {
        printf("DHCP client initialization failed\n\r");
        return (DEF_FAIL);
    }

    req_param[0]   = (DHCPc_OPT_CODE)DHCP_OPT_DOMAIN_NAME_SERVER;                             (2)
    req_param_qty  =  1;
    nbr_if_started =  0;
    if_nbr_cur     =  NET_IF_NBR_BASE_CFGD;

    while (nbr_if_started < APP_CFG_DHCP_NBR_IF_CFGD) {
        printf("Starting DHCP on interface %d... ", if_nbr_cur);
        DHCPc_Start((NET_IF_NBR      ) if_nbr_cur,                                            (3)
                    (DHCPc_OPT_CODE *)&req_param[0],
                    (CPU_INT08U      ) req_param_qty,
                    (DHCPc_ERR      *)&err);
        if (err == DHCPc_ERR_NONE) {
            printf("OK\n\r");

        } else {
            printf("FAILED\n\r");
            return (DEF_FAIL);
        }
        nbr_if_started++;
        if_nbr_cur++;
    }

    while (nbr_if_init < APP_CFG_DHCP_NBR_IF_CFGD) {
        OSTimeDlyHMSM(0, 0, 0, 100);

        if_done_ix = 0;
        if_nbr_cur = NET_IF_NBR_BASE_CFGD;

        while (if_done_ix < APP_CFG_DHCP_NBR_IF_CFGD) {
            if (if_dhcp_init_tbl[if_done_ix] != DEF_YES) {

                status = DHCPc_ChkStatus(NET_IF_NBR_BASE_CFGD, &err);                         (4)
                switch (status) {
                    case DHCP_STATUS_CFG_IN_PROGRESS:                                         (5)
                         break;

                    case DHCP_STATUS_CFGD:                                                    (6)
                         printf("IF %d configured\n\r", if_nbr_cur);
                         if_dhcp_init_tbl[if_done_ix] = DEF_YES;
                         nbr_if_init++;
                         break;

                    case DHCP_STATUS_CFGD_NO_TMR:                                             (7)
                         printf("IF %d configured (no timer set)\n\r", if_nbr_cur);
                         if_dhcp_init_tbl[if_done_ix] = DEF_YES;
                         nbr_if_init++;
                         break;

                    case DHCP_STATUS_CFGD_LOCAL_LINK:                                         (8)
                         printf("IF %d configured (link-local address)\n\r", if_nbr_cur);
                         if_dhcp_init_tbl[if_done_ix] = DEF_YES;
                         nbr_if_init++;
                         break;
                    case DHCP_STATUS_FAIL:                                                     (9)
                         printf("IF %d configuration failed\n\r", if_nbr_cur);
                         if_dhcp_init_tbl[if_done_ix] = DEF_YES;
                         nbr_if_init++;
                         break;

                    default:
                         break;
                }
            }

            if_done_ix++;
            if_nbr_cur++;
        }
    }

    return (DEF_OK);
}
```

L3-1(1) Initialize the DHCP client. If the process is successful, the DHCP client's tasks are started, and its various data structures are initialized.

L3-1(2) Request additional parameters from the DHCP server. Note that the server will not necessarily transmit those parameters.

L3-1(3) Start the DHCP management of the interfaces. Note that the interface is not configured yet upon returning from this function.

L3-1(4) Once the DHCP management of an interface has been started, the application may want to check the status of the lease negotiation in order to determine whether or not the interface has been properly configured.

L3-1(5) Status DHCP_STATUS_CFG_IN_PROGRESS means that the negotiation is still underway.

L3-1(6) Status DHCP_STATUS_CFGD indicates that the DHCP negotiation is done and that the interface is properly configured.

L3-1(7) Status DHCP_STATUS_CFGD_NO_TMR specifies that the DHCP negotiation is done and that the interface is properly configured, but no timer has been set for renewing the lease. The effect of this is that the lease is going to be permanent, even though the server might have set a time limit for it.

L3-1(8) Status DHCP_STATUS_CFGD_LOCAL_LINK means that the DHCP negotiation was not successful, and that a link-local address has been attributed to the interface. It is important to note that the DHCP client will not try to negotiate a lease with a server at this point.

L3-1(9) Status DHCP_STATUS_FAIL denotes a negotiation error. At this point, the application should call the DHCPc_Stop() function and decide what to do next.

Chapter

4

µC/DHCPc API

This chapter provides a reference to the µC/DHCPc 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 values
- Specific notes and warnings on using the service

4-1 DHCPc_Init()

Initializes the DHCP client.

FILES

dhcp-c.h/dhcp-c.c

PROTOTYPE

```
DHCPc_ERR  DHCPc_Init (void);
```

ARGUMENTS

None.

RETURNED VALUES

DHCPc_ERR_NONE, if no errors;

Specific initialization error code, otherwise.

REQUIRED CONFIGURATION

None.

NOTES / WARNINGS

None.

EXAMPLE USAGE

```
DHCPc_ERR  err;


err = DHCPc_Init();
if (err == DHCPc_ERR_NONE) {
    printf("Init successful\n\r");
} else {
    printf("Init error\n\r");
}
```

4-2 DHCPc_Start()

Starts the DHCP address configuration and management on the specified interface.

FILES

dhcp-c.h/dhcp-c.c

PROTOTYPE

```
void  DHCPc_Start  (NET_IF_NBR       if_nbr,
                   DHCPc_OPT_CODE  *preq_param_tbl,
                   CPU_INT08U       req_param_tbl_qty,
                   DHCPc_ERR       *perr);
```

ARGUMENTS

if_nbr Interface number to start DHCP configuration and management.

preq_param_tbl Pointer to table of requested DHCP parameters.

req_param_tbl_qty Pointer to buffer that will receive the option value.

perr Pointer to variable that will receive the return error code from this function:

```
DHCPc_ERR_NONE
DHCPc_ERR_INIT_INCOMPLETE
DHCPc_ERR_IF_INVALID
DHCPc_ERR_PARAM_REQ_TBL_SIZE
DHCPc_ERR_MSG_Q
DHCPc_OS_ERR_LOCK
DHCPc_ERR_IF_INFO_IF_USED
DHCPc_ERR_INVALID_HW_ADDR
DHCPc_ERR_IF_INFO_NONE_AVAIL
DHCPc_ERR_COMM_NONE_AVAIL
```

RETURNED VALUES

None.

REQUIRED CONFIGURATION

None.

NOTES / WARNINGS

None.

EXAMPLE USAGE

```
DHCPc_OPT_CODE  req_param[DHCPc_CFG_PARAM_REQ_TBL_SIZE];
CPU_INT08U      req_param_qty;
DHCPc_ERR       err;

req_param[0]  = (DHCPc_OPT_CODE)DHCP_OPT_DOMAIN_NAME_SERVER;
req_param_qty =  1;

DHCPc_Start((NET_IF_NBR      ) ET_IF_NBR_BASE_CFGD,
            (DHCPc_OPT_CODE *)&req_param[0],
            (CPU_INT08U      ) req_param_qty,
            (DHCPc_ERR      *)&err);
if (err == DHCPc_ERR_NONE) {
    printf("Interface DHCP management     successfully started\n\r");
} else {
    printf("Interface DHCP management NOT successfully started\n\r");
}
```

4-3 DHCPc_Stop()

Stops the DHCP address configuration and management on the specified interface.

FILES

dhcp-c.h/dhcp-c.c

PROTOTYPE

```
void  DHCPc_Stop (NET_IF_NBR   if_nbr,
                  DHCPc_ERR   *perr);
```

ARGUMENTS

if_nbr   Interface number to stop the DHCP configuration and management.

perr   Pointer to variable that will receive the return error code from this function:

DHCPc_ERR_NONE
DHCPc_ERR_INIT_INCOMPLETE
DHCPc_ERR_IF_INFO_IF_NOT_USED
DHCPc_ERR_MSG_Q
DHCPc_OS_ERR_LOCK
DHCPc_ERR_COMM_NONE_AVAIL

RETURNED VALUES

None

REQUIRED CONFIGURATION

None.

NOTES / WARNINGS

None.

EXAMPLE USAGE

```
DHCPc_ERR   err;

DHCPc_Stop(NET_IF_NBR_BASE_CFGD, &err);
if (err == DHCPc_ERR_NONE) {
    printf("Interface DHCP management     successfully stopped\n\r");
} else {
    printf("Interface DHCP management NOT successfully stopped\n\r");
}
```

4-4 DHCPc_ChkStatus()

Checks an interface’s DHCP status and last error.

FILES

dhcp-c.h/dhcp-c.c

PROTOTYPE

```
void DHCPc_Stop (NET_IF_NBR   if_nbr,
                 FDHCPc_ERR  *perr_last);
```

ARGUMENTS

if_nbr Interface number to check status.

perr_last Pointer to variable that will receive the last return error code for this interface:

```
DHCPc_ERR_NONE
DHCPc_ERR_INIT_INCOMPLETE
DHCPc_ERR_IF_NOT_MANAGED
```

RETURNED VALUES

DHCP status for the interface.

REQUIRED CONFIGURATION

None.

NOTES / WARNINGS

None.

EXAMPLE USAGE

```
DHCPc_STATUS  status;
DHCPc_ERR     err;

status = DHPCc_ChkStatus(NET_IF_NBR_BASE_CFGD, &err);
switch (status) {
    case DHCP_STATUS_CFGD:
    case DHCP_STATUS_NO_TMR:
    case DHCP_STATUS_LOCAL_LINK:
         printf("Interface configured\n\r");
         break;

    default:
         printf("Interface NOT configured\n\r");
         break;
}
```

4-5 DHCPc_GetOptVal()

Gets the value of a specific DHCP option for a given interface.

FILES

dhcp-c.h/dhcp-c.c

PROTOTYPE

```
void  DHCPc_GetOptVal  (NET_IF_NBR        if_nbr,
                        DHCPc_OPT_CODE    opt_code,
                        CPU_INT08U       *pval_buf,
                        CPU_INT16U       *pval_buf_len,
                        DHCPc_ERR        *perr);
```

ARGUMENTS

if_nbr Interface number to get option value.

opt_code Option code to get value.

pval_buf Pointer to buffer that will receive the option value.

pval_buf_len Pointer to a variable to … :

Pass the size of the buffer, in octets, pointed to by pval_buf;

Return the actual length of the option, if no errors;
Return an undefined value, otherwise.

perr Pointer to variable that will receive the return error code from this function:

DHCPc_ERR_NONE
DHCPc_ERR_NULL_PTR
DHCPc_ERR_INIT_INCOMPLETE
DHCPc_ERR_IF_NOT_MANAGED
DHCPc_ERR_IF_NOT_CFG
DHCPc_ERR_IF_OPT_NONE

DHCPc_ERR_OPT_BUF_SIZE
DHCPc_OS_ERR_LOCK

RETURNED VALUES

None.

REQUIRED CONFIGURATION

None.

NOTES / WARNINGS

None.

EXAMPLE USAGE

```
#define  DHCP_TIME_OFFSET_VAL_BUF_LEN   4

CPU_INT08U  time_offset_val_buf[DHCP_TIME_OFFSET_VAL_BUF_LEN];
CPU_INT16U  offset_val_buf_len;
DHCPc_ERR   err;

offset_val_buf_len = DHCP_TIME_OFFSET_VAL_BUF_LEN;
DHCPc_GetOptVal((NET_IF_NBR    ) NET_IF_NBR_BASE_CFGD,
                (DHCPc_OPT_CODE) DHCP_OPT_TIME_OFFSET,
                (CPU_INT08U   *)&time_offset_val_buf[0],
                (CPU_INT16U   *)&offset_val_buf_len,
                (DHCPc_ERR    *)&err);
if (err == DHCPc_ERR_NONE) {
    printf("Time offset successful retrieved\n\r");
} else {
    printf("Error retrieving DHCP time offset\n\r");
}
```

Appendix

A

µC/DHCPc Licensing Policy

You need to obtain an “Object Code Distribution License” to embed µC/DHCPc in a product that is sold with the intent to make a profit. Each individual 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
1290 Weston Road, Suite 306
Weston, FL 33326
USA

Phone:	+1 954 217 2036
Fax:	+1 954 217 2037
E-mail:	Licensing@Micrium.com
Web:	www.Micrium.com

Appendix

B

References

Labrosse, Jean J. *µC/OS-III, The Real-Time Kernel*, Micriµm Press, 2009.

Labrosse, Jean J. *MicroC/OS-II: The Real Time Kernel*. 2nd edition. Newnes, 2002.

Légaré, Christian. *µC/TCP-IP Protocol Stack*. Micriµm Press, 2010.