Ingenieurbüro für Echtzeitprogrammierung

DNS-LIB

DNS unter RTOS-UH

Dok-Rev. 1.1 vom 14.11.2007

Software-Rev. 1.0 vom 13.07.2005

Inhaltsverzeichnis

Revisionsliste:

Rev.	Datum	Na.	Änderung
1.0	13.07.2005	Kr	Erstellung
1.1	18.08.2006	Ko	Überarbeitung

1 Urheberrecht und Haftung

Alle Rechte an diesen Unterlagen liegen bei der IEP GmbH, Langenhagen.

Die Vervielfältigung, auch auszugsweise, ist nur mit unserer ausdrücklichen schriftlichen Genehmigung zulässig.

In Verbindung mit dem Kauf von Software erwirbt der Käufer einfaches, nicht übertragbares Nutzungsrecht. Dieses Recht zur Nutzung bezieht sich ausschließlich darauf, daß dieses Produkt auf oder in Zusammenhang mit jeweils **einem** Computer zu benutzen ist. Das Erstellen einer Kopie ist ausschließlich zu Archivierungszwecken unter Aufsicht des Käufers oder seines Beauftragten zulässig. Der Käufer haftet für Schäden, die sich aus der Verletzung seiner Sorgfaltspflicht ergeben, z.B. bei unautorisiertem Kopieren, unberechtigter Weitergabe der Software usw.. Der Käufer gibt mit dem Erwerb der Software seine Zustimmung zu den genannten Bedingungen. Bei unlizensiertem Kopieren muß vorbehaltlich einer endgültigen juristischen Klärung von Diebstahl ausgegangen werden. Dies gilt ebenso für Dokumentation und Software, die durch Modifikation aus Unterlagen und Programmen von IEP hervorgegangen ist, gleichgültig, ob die Änderungen als geringfügig oder erheblich anzusehen sind.

Eine Haftung seitens IEP für Schäden, die auf den Gebrauch von Software, Hardware oder Benutzung dieses Manuskriptes zurückzuführen sind, wird ausdrücklich ausgeschlossen, auch für den Fall fehlerhafter Software oder irrtümlicher Angaben.

Das Einverständnis des Käufers oder Nutzers für den Haftungsausschluß gilt mit dem Kauf und der Nutzung der Software und dieser Unterlagen als erteilt.

2 Die DNS-Programmierschnittstelle unter RTOS-UH

Der **DNS** (Domain Name Service) dient im Netzwerk/Internet zur Verbindung von Hostnamen und IP-Adressen, er stellt Funktionen zur Verfügung, damit man bei bekannter IP-Adresse den Hostnamen ermitteln kann, oder umgekehrt für einen gegebenen Hostnamen die zugehörige IP-Adresse ermitteln kann.

2.1 Voraussetzungen für die DNS-Programmierschnittstelle

Die DNS-Programmierschnittstelle dient zum Zugriff auf die lokal im System vorhandene `HOSTS`-Datei und DNS-Abfragen an externe DNS-Server.

Für die volle Funktionalität müssen im System folgende Voraussetzungen geschaffen sein:

1. Es muss eine `HOSTS`-Datei lokal auf dem Rechner geben.
2. Es muss eine globale Environmentvariable `HOSTFILE` geben, die den Zugriffspfad auf die `HOSTS`-Datei enthält. Die `HOSTS`-Datei muss mindestens den Eintrag für den lokalen Rechner beinhalten.
3. Es muss eine globale Environmentvariable `DNS_SERVER` geben, die mindestens eine gültige IP-Adresse eines offiziellen DNS-Servers enthält.
4. In der Netzwerkkonfiguration muss eine Route für das Internet bzw. das Netz mit einem DNS-Server eingerichtet sein.

Als Programmiersprache werden PEARL und CREST-C unterstützt.

2.1.1 HOSTS-Datei

Die lokale `HOSTS`-Datei sollte mindestens folgenden Aufbau haben:

```
192.168.10.11 MeinRTOS MeinRTOS ; optionaler Kommentar
192.168.10.12 KollegenRTOS ; der Rechner des Kollegen
; dies ist ein Kommentar
;START DNS-ENTRY
;END DNS-ENTRY
<EOF>
```

Die angegebenen IP-Adressen sind beispielhaft und müssen natürlich den lokalen Netzwerkeinstellungen entsprechen.

Die erste Zeile gibt den eigenen Rechnername an, (die Bezeichnung *MeinRTOS* ist hier als Beispiel gewählt), durch den doppelten Namenseintrag wird der eigene Rechnername gekennzeichnet.

Kommentare sind durch ein Semikolon (;) gekennzeichnet, sie gelten immer für den Rest der Zeile.

2.1.1.1 Dynamischer Block in HOSTS-Datei

Besondere Bedeutung haben die Kommentarzeilen:

;START DNS-ENTRY kennzeichnet den Start des dynamischen DNS-Blockes

;END DNS-ENTRY kennzeichnet das Ende des dynamischen DNS-Blockes, und diese Zeile sollte die letzte Zeile in der HOSTS-Datei sein.

In den dynamischen DNS-Block der HOSTS-Datei werden mit Hilfe der im folgenden beschriebenen Funktionen von einem externen DNS-Server bezogene IP-Adressen und Namensbezüge eingetragen. Ein Eintrag ist folgendermaßen aufgebaut:

```
...
;START DNS-ENTRY
212.227.206.144 www.iep.de ; DNS
;END DNS-ENTRY
<EOF>
```

Hier kennzeichnet der Kommentar ;DNS, dass dieser Eintrag dynamisch eingetragen wurde.

2.1.2 Environmentvariable HOSTFILE

Die Environmentvariable HOSTFILE kann folgendermaßen gesetzt werden:

```
ENVSET -G HOSTFILE=/R0/HOSTS
```

Sie gibt an, wo auf dem System die HOSTS-Datei abgelegt ist, der Pfad */R0/* ist hier beispielhaft angegeben, es kann jeder andere Pfad genutzt werden (z.B. */h0/ETC* o. ä.).

2.1.3 Environmentvariable DNS_SERVER

Die Environmentvariable DNS_SERVER wird folgendermaßen gesetzt:

```
ENVSET -G DNS_SERVER=194.25.2.129
```

Sie gibt die IP-Adresse eines externen DNS-Servers an, es kann auch eine Liste von Servern angegeben werden, dann sind die IP-Adressen der einzelnen Server durch Kommata zu trennen.

```
z.B. ENVSET -G DNS_SERVER=194.25.2.129,192.168.10.1,10.2.3.5
```

2.2 Die DNS-Libraries

Für die Sprache PEARL steht eine Library NET2SRx.P90 zur Verfügung, diese kann auch permanent im Eprom oder Flash abgelegt werden.

Für CREST-C sind die Programme mit der NET2LIB.LIB zu linken.

3 Die DNS Funktionen der Library im Überblick

Für den DNS stehen die folgenden Funktionen in der Library zur Verfügung:

Funktion	Bedeutung
RT_gethostname	Abfrage des eigenen Rechnernamens
RT_hostbyname	IP-Adresse eines Rechnernamens auslesen
RT_gethostbyaddr	Rechnernamen aus der IP-Adresse erhalten
RT_inet_addr	IP-Adresse aus IP-Dotted_String erhalten
RT_inet_ntoa	IP-Dotted-String aus IP-Adresse erhalten
RT_hostname_to_ipaddr	Rechnername aus Host-String erhalten
RT_dns_request	DNS-Anfrage für Hostnamen starten
RT_dns_reverse	DNS-Anfrage für IP-Adresse starten
RT_insert_dns_host	Eintragen eines Hosts in den HOSTFILE als DNS
RT_delete_all_dns	Löschen alle DNS Einträge im HOSTFILE

Der Präfix *RT_* gilt für die PEARL90 Funktionen, für die CREST-C Funktionen muss er durch den Präfix *rt_* ersetzt werden.

Beispiel:

Die Funktion RT_dns_request aus PEARL90 wird zu rt_dns_request in CREST-C.

3.1 RT_gethostname

Voraussetzungen: HOSTS-Datei, HOSTFILE Environmentvariable gesetzt.

Die Funktion **RT_gethostname** liefert in einen bereitgestellten Puffer den eigenen Rechnernamen zurück, falls dieser in der HOSTS-Datei vereinbart ist. Der Puffer muss eine ausreichende Größe haben, damit der Rechnername eingefügt werden kann, sonst wird der Name abgeschnitten. Die maximal in den Puffer einfüllbaren Zeichen werden durch den Parameter maxlen festgelegt. Als Rückgabewert wird die Länge des eingesetzten Namen zurückgegeben (> 0 wenn erfolgreich).

3.1.1 RT_gethostname Anwendung in PEARL90

```
SPC RT_gethostname ENTRY( name CHAR(1) IDENT,
                          maxlen FIXED(15) /* Laenge Buffer*/
                        ) RETURNS( FIXED(15) ) GLOBAL ;
...
DCL name CHAR(80) ;
DCL len  FIXED(15);
...
len = RT_gethostname( name.CHAR(1), 80 ) ;
IF len GT 0 THEN
   /* eigener Rechnername ist in „name“ gesetzt */
ELSE
   /* Abfrage fehlgeschlagen */
```

3.1.2 rt_gethostname Anwendung in CREST-C

```
int rt_gethostname( char *name, int maxlen ) ;
...
char name[80] ;
int  len      ;
...
len = rt_gethostname( name, sizeof( name ) ) ;
if ( len > 0 )
   /* eigener Rechnername ist in „name“ gesetzt */
else
   /* Abfrage fehlgeschlagen */
```

3.2 RT_hostbyname

Voraussetzungen: HOSTS-Datei, HOSTFILE Environmentvariable gesetzt.

Die Funktion **RT_hostbyname** liefert zu einem Rechnernamen die IP-Adresse zurück, falls dieser in der HOSTS-Datei vereinbart ist. Als Rückgabewert wird die gefundene IP-Adresse zurückgegeben (!= 0 wenn erfolgreich).

3.2.1 RT_hostbyname Anwendung in PEARL90

```
SPC RT_hostbyname ENTRY( name CHAR(1) IDENT
                       ) RETURNS( FIXED(31) ) GLOBAL ;
...
DCL name  CHAR(80) ;
DCL ipadr FIXED(31);
...
name  = 'KollegenRTOS' ;
ipadr = RT_gethostname( name.CHAR(1) ) ;
IF ipadr NE 0(31) THEN
   /* ipadr enthält die IP-Adresse des Rechners */
ELSE
   /* Name nicht bekannt */
```

3.2.2 rt_hostbyname Anwendung in CREST-C

```
unsigned long rt_hostbyname( char *name ) ;
...
char name[80] ;
unsigned long ipadr ;
...
name  = "KollegenRTOS" ;
ipadr = rt_gethostname( name ) ;
if ( ipadr )
   /* ipadr enthält die IP-Adresse des Rechners */
else
   /* Name nicht bekannt */
```

3.3 RT_gethostbyaddr

Voraussetzungen: HOSTS-Datei, HOSTFILE Environmentvariable gesetzt.

Die Funktion **RT_gethostbyaddr** liefert zu einer angegebenen IP-Adresse den Hostnamen zurück, falls dieser in der HOSTS-Datei vorhanden ist. Bei PEARL90 wird als Rückgabewert die gefundene Länge des eingesetzten Namens zurückgegeben (> 0 wenn erfolgreich).

Bei CREST-C wird ein char* auf den gefundenen Namen zurückgeben.

3.3.1 RT_gethostbyaddr Anwendung in PEARL90

```
SPC RT_gethostbyaddr ENTRY( addr FIXED(31) ,
                            name CHAR(1) IDENT ,
                            maxlen FIXED(15)
                          ) RETURNS( FIXED(15) ) GLOBAL ;
...
DCL name  CHAR(80)  ;
DCL ipadr FIXED(31) ;
DCL len   FIXED(15) ;
...
ipadr = TOFIXED('C0A80A0C'B4); /* 192.168.10.12 */
len = RT_gethostname( ipadr, name.CHAR(1), 80 ) ;
IF len GT 0 THEN
   /* name enthält den Namen des Rechners */
ELSE
   /* IP-Adresse nicht bekannt */
```

3.3.2 rt_gethostbyaddr Anwendung in CREST-C

Abweichend von der PEARL-Prozedur ist hier der Rückgabewert der Funktion ein Pointer auf den Namens-String, er ist gleich NULL wenn die IP-Adresse nicht bekannt ist.

```
char* rt_hostbyname( unsigned long ipadr,
                     char *name, int maxlen ) ;
...
char           name[80] ;
unsigned long  ipadr    ;
char          *p        ;
...
ipadr = 0xC0A80A0C ;
p = rt_gethostname( ipadr, name, sizeof(name) ) ;
if ( p )
   /* p enthält den Pointer auf den Namen des Rechners, somit
```

```
        * ist p = &name[0]
        */
    else
       /* ipadr nicht bekannt */
```

3.4 RT_inet_addr

Voraussetzungen: keine

Die Funktion **RT_inet_addr** liefert zu einem angegebenen IP-dotted-String die IP-Adresse zurück (> 0 wenn erfolgreich).

3.4.1 RT_inet_addr Anwendung in PEARL90

```
SPC RT_inet_addr ENTRY( ipstr CHAR(1) IDENT
                      ) RETURNS( FIXED(31) ) GLOBAL ;
...
DCL ipstr  CHAR(15) ;
DCL ipadr FIXED(31) ;
...
ipstr = '192.168.10.13' ;
ipadr = RT_inet_addr( ipstr.CHAR(1) ) ;
IF ipadr NE 0(31) THEN
   /* ipadr enthält die IP-Adresse */
ELSE
   /* IP-Adresse nicht wandelbar */
```

3.4.2 rt_inet_addr Anwendung in CREST-C

```
Unsigned long rt_inet_addr( char *ipstr ) ;
...
char ipstr[16] ;
unsigned long ipadr ;
...
ipstr = "192.168.10.13" ;
ipadr = rt_inet_addr( ipstr ) ;
if ( ipadr )
   /* ipadr enthält die IP-Adresse */
else
   /* ipadr nicht wandelbar */
```

3.5 RT_inet_ntoa

Voraussetzungen: keine

Die Funktion **RT_inet_ntoa** liefert zu einer angegebenen IP-Adresse den IP-Dotted-String zurück. Bei PEARL90 wird als Rückgabewert die gefundene Länge des eingesetzten IP-Strings zurückgegeben (> 0 wenn erfolgreich).

Bei CREST-C wird ein char* auf den gewandelten IP-String zurückgeben (!= NULL wenn erfolgreich).

3.5.1 RT_inet_ntoa Anwendung in PEARL90

```
SPC RT_inet_ntoa ENTRY( ipadr FIXED(31),
                        ipstr CHAR(1) IDENT,
                        maxlen FIXED(15)
                      ) RETURNS( FIXED(15) ) GLOBAL ;
...
DCL ipstr CHAR(16)  ;
DCL ipadr FIXED(31) ;
DCL len   FIXED(15) ;
...
ipadr = TOFIXED('C0A80A0D'B4) ;
len = RT_inet_addr( ipadr, ipstr.CHAR(1), 16 ) ;
IF len GT 0 THEN
   /* ipstr enthält den IP-String 192.168.10.13 */
ELSE
   /* buffer zu klein */
```

3.5.2 rt_inet_ntoa Anwendung in CREST-C

```
char * rt_inet_ntoa( unsigned long ipadr,
                     char *ipstr,
                     int maxlen ) ;
...
char           ipstr[16] ;
unsigned long  ipadr     ;
char          *p         ;
...
ipadr = 0xC0A80A0D ;
p = rt_inet_ntoa( ipadr, ipstr, sizeof( ipstr ) ) ;
if ( p )
   /* ipstr enthält den IP-String „192.168.10.13“ */
else
   /* buffer zu klein */
```

3.6 RT_hostname_to_ipaddr

Voraussetzungen: HOSTS-Datei, HOSTFILE und DNS_SERVER Environmentvariable gesetzt.

Die Funktion **RT_hostname_to_ipaddr** liefert zu einen angegebenen Hostnamen oder IP-Dotted-String die IP-Adresse zurück.

Dabei unterstützt die Funktion verschiedene Modi, gesteuert durch die Eingangsvariable dns:

dns=0

Es wird versucht den Namen als IP-Dotted-String zu wandeln, wenn die Notation nicht wandelbar ist, wird sie als Hostname interpretiert und nach diesem in der HOSTS-Datei gesucht.

dns=1

Zusätzlich wird eine DNS-Anfrage an den externen DNS-Server gestellt.

dns=2

Falls die IP-Adresse bei der DNS-Anfrage gefunden wurde, wird sie im HOSTFILE eingetragen.

Im Fall dns != 0 wird für die DNS-Abfrage das angegebene dns_timeout [in msec] berücksichtigt.

Folgende Suchreihenfolge wird für den angegebenen Namen eingehalten:

1. Versuch der Wandlung eines IP-Dotted-String
2. Suche des Namen in der HOSTS-Datei
3. DNS-Server Anfrage

Falls einer dieser Schritte erfolgreich war, werden die Nachfolgenden nicht mehr ausgeführt.

Falls die Environmentvariable DNS_SERVER nicht gesetzt ist, so sind die Modi dns=1 oder 2 ohne Wirkung.

Als Rückgabewert wird die gewandelte IP-Adresse zurückgegeben (> 0 wenn erfolgreich).

3.6.1 RT_hostname_to_ipaddr Anwendung in PEARL90

```
SPC RT_inet_ntoa ENTRY( name CHAR(1) IDENT,
                        dns  FIXED(15),
                        dns_timeout FIXED(31)
                      ) RETURNS( FIXED(31) ) GLOBAL ;
...
DCL name  CHAR(80)  ;
DCL ipadr FIXED(31) ;
DCL dns   FIXED(15) INIT(2); /* Mode = 2 */
DCL dns_t FIXED(31) INIT(2000(31)) ; /* Timeout 2 Sec */
...
name = ‘www.iep.de‘ ;
ipadr = RT_hostname_to_ipaddr( name.CHAR(1), dns, dns_t ) ;
IF ipadr NE 0(31) THEN
   /*
    * ipadr enthält die IP-Adresse von www.iep.de und in der
    * HOSTS-Datei ist ein entsprechender Eintrag vorhanden.
    */
ELSE
   /* nicht wandelbar */
```

3.6.2 rt_hostname_to_ipaddr Anwendung in CREST-C

```
unsgined long rt_hostname_to_ipaddr( char *name,
                                     int dns, int dns_timeout ) ;
...
char *name ;
unsigned long ipadr ;
int dns_m = 1 ; /* keinen Eintrag in HOSTS-Datei machen */
int dns_t = 10000 ; /* 10 Sekunden Timeout */
...
name = “www.iep.de“ ;
ipadr = rt_hostname_to_ipaddr( name, dns_m, dns_t ) ;
if ( ipadr )
   /* ipadr enthält die IP-Adr von www.iep.de */
else
   /* nicht wandelbar */
```

3.7 RT_dns_request

Voraussetzungen: DNS_SERVER Environmentvariable gesetzt.

Die Funktion RT_dns_request liefert zu einen angegebenen Hostnamen die IP-Adresse zurück.

Falls die Environmentvariable DNS_SERVER nicht gesetzt ist, so liefert die Funktion immer eine 0 zurück. Die Funktion fragt grundsätzlich bei dem DNS-Server nach dem Hostnamen und berücksichtig eventuell vorhandene Einträge in der HOSTS-Datei nicht.

Als Rückgabewert wird die gewandelte IP-Adresse zurückgegeben (> 0 wenn erfolgreich).

3.7.1 RT_dns_request Anwendung in PEARL90

```
SPC RT_dns_request ENTRY( name CHAR(1) IDENT,
                          dns_timeout FIXED(31)
                        ) RETURNS( FIXED(31) ) GLOBAL ;
...
DCL name  CHAR(80)  ;
DCL ipadr FIXED(31) ;
DCL dns_t FIXED(31) INIT(2000(31)) ; /* Timeout 2 Sec */
...
name = ‘www.iep.de‘ ;
ipadr = RT_dns_request( name.CHAR(1), dns_t ) ;
IF ipadr NE 0(31) THEN
   /* ipadr enthält die IP-Adresse von www.iep.de */
ELSE
   /* nicht gefunden */
```

3.7.2 rt_dns_request Anwendung in CREST-C

```
unsgined long rt_dns_request( char *name,
                              int dns_timeout ) ;
...
char          *name  ;
unsigned long  ipadr ;
int            dns_t = 10000 ; /* 10 Sekunden Timeout */
...
name = “www.iep.de“ ;
ipadr = rt_dns_request( name, dns_t ) ;
if ( ipadr )
   /* ipadr enthält die IP-Adr von www.iep.de */
else
   /* nicht wandelbar */
```

3.8 RT_dns_reverse

Voraussetzungen: DNS_SERVER Environmentvariable gesetzt.

Die Funktion **RT_dns_reverse** liefert zu einer angegebenen IP-Adresse den von einem DNS-Server gelieferten Hostnamen zurück.

Falls die Environmentvariable DNS_SERVER nicht gesetzt ist, so liefert die Funktion immer eine 0 zurück. Die Funktion fragt grundsätzlich bei dem DNS-Server nach der IP-Adresse und berücksichtigt eventuell vorhandene Einträge in der HOSTS-Datei nicht.

Es sollten nur Anfragen mit dieser Funktion gemacht werden, wenn die IP-Adresse auch aus dem Internet oder einem DNS-Server stammt.

Als Rückgabewert wird die Länge des gelieferten Hostname zurückgegeben(> 0 wenn erfolgreich).

3.8.1 RT_dns_reverse Anwendung in PEARL90

```
SPC RT_dns_reverse ENTRY( ipadr FIXED(31),
                          name CHAR(1) IDENT,
                          maxlen FIXED(15),
                          dns_timeout FIXED(31)
                        ) RETURNS( FIXED(15) ) GLOBAL ;
...
DCL name  CHAR(80)  ;
DCL ipadr FIXED(31) ;
DCL dns_t FIXED(31) INIT(2000(31)) ; /* Timeout 2 Sec */
...
ipadr = TOFIXED('D4E3CE90'B4) ;
len = RT_dns_reverse( ipadr, name.CHAR(1), 80, dns_t ) ;
IF len GT 0 THEN
   /* name enthält den Hostnamen www.iep.de */
ELSE
   /* nicht gefunden */
```

3.8.2 rt_dns_request Anwendung in CREST-C

```
int rt_dns_request( unsigned long ipadr,
                    char *name,
                    int maxlen,
                    int dns_timeout ) ;
...
char          name[80] ;
unsigned long ipadr    ;
int           dns_t = 10000 ; /* 10 Sekunden Timeout */
int           len      ;
...
```

```
ipadr = 0xD4E3Ce90 ;
len = rt_dns_reverse( ipadr, name, sizeof( name ), dns_t );
if ( len > 0 )
   /* name enthält den Hostnamen www.iep.de */
else
   /* nicht wandelbar */
```

3.9 RT_insert_dns_host

Voraussetzungen: HOSTS-Datei, HOSTFILE Environmentvariable gesetzt.

Die Funktion **RT_insert_dns_host** trägt eine Kombination aus angegebener IP-Adresse und Hostnamen in die HOSTS-Datei als dynamischen DNS-Eintrag ein.

Falls die Environmentvariable HOSTFILE nicht gesetzt ist, so liefert die Funktion immer eine 0 zurück. Als Rückgabewert wird der erfolgreiche Eintrag in der HOSTS-Datei gemeldet (> 0 wenn erfolgreich).

3.9.1 RT_insert_dns_host Anwendung in PEARL90

```
SPC RT_insert_dns_host ENTRY( name CHAR(1) IDENT,
                              ipadr FIXED(31)
                            ) RETURNS( FIXED(15) ) GLOBAL ;
...
DCL name  CHAR(80)  ;
DCL ipadr FIXED(31) ;
DCL stat  FIXED(15) ;
...
ipadr = TOFIXED('D4E3CE90'B4) ;
name  = 'www.iep.de' ;
stat = RT_insert_dns_host( name.CHAR(1), ipadr ) ;
IF stat GT 0 THEN
   /* Die HOSTS-Datei enthält eine neue Zeile nach der
    * Zeile ;START DNS-ENTRY mit folgendem Inhalt
    * 212.227.206.144 www.iep.de ; DNS
    */
ELSE
   /* HOSTS-Datei nicht gefunden oder kein dynamischer
    * Block eingerichtet (siehe 2.1.1.1)
    */
```

3.9.2 rt_insert_dns_host Anwendung in CREST-C

```
int rt_insert_dns_host( unsigned long ipadr,
                        char *name ) ;
...
char          *name  ;
unsigned long  ipadr ;
int            stat  ;
...
ipadr = 0xD4E3CE90 ;
name = "www.iep.de" ;
```

```
stat = rt_insert_dns_host( ipadr, name) ;
if ( stat > 0 )
   /*
    * Die HOSTS-Datei enthält eine neue Zeile nach
    * der Zeile ;START DNS-ENTRY mit folgendem Inhalt
    * 212.227.206.144 www.iep.de ; DNS
    */
else
   /*
    * HOSTS-Datei nicht gefunden oder kein dynamischer
    * Block eingerichtet (siehe 2.1.1.1)
    */
```

3.10 RT_delete_dns_all

Voraussetzungen: HOSTS-Datei, HOSTFILE Environmentvariable gesetzt.

Die Funktion **RT_delete_dns_all** löscht alle dynamischen DNS-Einträge aus der HOSTS-Datei. Falls die Environmentvariable HOSTFILE nicht gesetzt ist, so liefert die Funktion immer eine 0 zurück. Als Rückgabewert wird das erfolgreiche Löschen in der HOSTS-Datei gemeldet (> 0 wenn erfolgreich).

3.10.1 RT_delete_dns_all Anwendung in PEARL90

```
SPC RT_delete_dns_all ENTRY RETURNS( FIXED(15) ) GLOBAL ;
...
DCL stat  FIXED(15) ;
...
stat = RT_delete_dns_all ;
IF stat GT 0 THEN
   /*
    * Die HOSTS-Datei hat keine Zeile mehr im dynamischen
    * Block, alle eingetragenen Rechner sind gelöscht
    */
ELSE
   /*
    * HOSTS-Datei nicht gefunden oder kein dynamischer
    * Block eingerichtet (siehe 2.1.1.1)
    */
```

3.10.2 rt_delete_dns_all Anwendung in CREST-C

```
int rt_delete_dns_all( void ) ;
...
int stat ;
...
stat = rt_delete_dns_all() ;
if ( stat > 0 )
   /*
    * Die HOSTS-Datei hat keine Zeile mehr im dynamischen
    * Block, alle eingetragenen Rechner sind gelöscht
    */
else
   /*
    * HOSTS-Datei nicht gefunden oder kein dynamischer
    * Block eingerichtet (siehe 2.1.1.1)
    */
```