KwikNet™

Point-to-Point Protocol

User's Guide

First Printing: August 21, 1998
Last Printing: November 1, 2000

Manual Order Number: PN303-9P

KADAK Products Ltd.
206-1847 West Broadway Avenue
Vancouver, B.C., Canada, V6J 1Y5
Phone: (604) 734-2796
Fax: (604) 734-8114

TECHNICAL SUPPORT

KADAK Products Ltd. is committed to technical support for its software products. Our programs are designed to be easily incorporated in your systems and every effort has been made to eliminate errors.

Engineering Change Notices (ECNs) are provided periodically to repair faults or to improve performance. You will automatically receive these updates for a period of one year. After that period, you may purchase additional updates. Please keep us informed of the primary user in your company to whom these update notices and other pertinent information should be directed.

Should you require direct technical assistance in your use of this KADAK software product, engineering support is available by telephone, fax or e-mail without charge. KADAK reserves the right to charge for technical support services which it deems to be beyond the normal scope of technical support.

We would be pleased to receive your comments and suggestions concerning this product and its documentation. Your feedback helps in the continuing product evolution.

KADAK Products Ltd.
#206 - 1847 West Broadway Avenue
Vancouver, B.C., Canada, V6J 1Y5

Phone: (604) 734-2796
Fax: (604) 734-8114
e-mail: amxtech@kadak.com

DISCLAIMER

KADAK Products Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties or merchantability or fitness for any particular purpose. Further, KADAK Products Ltd. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of KADAK Products Ltd. to notify any person of such revision or changes.

TRADEMARKS

AMX in the stylized form is a registered trademark of KADAK Products Ltd.
KwikNet, AMX, InSight and *KwikLook* are trademarks of KADAK Products Ltd.
UNIX is a registered trademark of AT&T Bell Laboratories.
Microsoft, MS-DOS and Windows are registered trademarks of Microsoft Corporation.
All other trademarked names are the property of their respective owners.

KwikNet Point-to-Point Protocol User's Guide

Table of Contents

This page left blank intentionally.

1. KwikNet PPP Overview

1.1 Introduction

The Point-to-Point Protocol (PPP) is a network protocol used to control the delivery of IP datagrams between two host computers interconnected by a serial link. The KwikNet PPP option adds this protocol to the KwikNet™ TCP/IP Stack, a compact, reliable, high performance TCP/IP stack, well suited for use in embedded networking applications. PPP offers improved reliability and security features not found in SLIP, the Serial Line Internet Protocol included with KwikNet.

The KwikNet PPP option is best used with a real-time operating system (RTOS) such as KADAK's AMX™ Real-Time Multitasking Kernel. However, the KwikNet PPP option can also be used in a single threaded environment without an RTOS. The KwikNet Porting Kit User's Guide describes the use of KwikNet with your choice of RT/OS. Note that throughout this manual, the term RT/OS is used to refer to any operating system, be it a multitasking RTOS or a single threaded OS.

You can readily tailor the KwikNet stack to accommodate your PPP needs by using the KwikNet Configuration Builder, a Windows® utility which makes configuring KwikNet a snap. Your KwikNet stack will only include the PPP features required by your application.

The Point-to-Point Protocol (PPP) is formally defined by the IETF document RFC-1661. That document describes the Link Control Protocol (LCP) used by PPP to negotiate the configuration of the link over which communication will occur.

IETF document RFC-1334 describes the Password Authentication Protocol (PAP) first used by PPP to provide link authentication prior to network use. That protocol has been updated by the Challenge-Handshake Authentication Protocol (CHAP) described by IETF document RFC-1994.

The IP Control Protocol (IPCP), described in IETF document RFC-1332, defines the manner in which IP datagrams are delivered over the link established by PPP. KwikNet also supports the Name Server Address extension to IPCP as defined in RFC-1877.

The KwikNet PPP option is compliant with these specifications. The RFCs should be consulted for any detailed questions concerning the PPP protocol.

Beyond the summary above, this manual makes no attempt to describe the Point-to-Point Protocol (PPP), what it is or how it operates. It is assumed that you have a working knowledge of the PPP protocol as it applies to your needs. Reference materials are provided in Appendix A of the KwikNet TCP/IP Stack User's Guide.

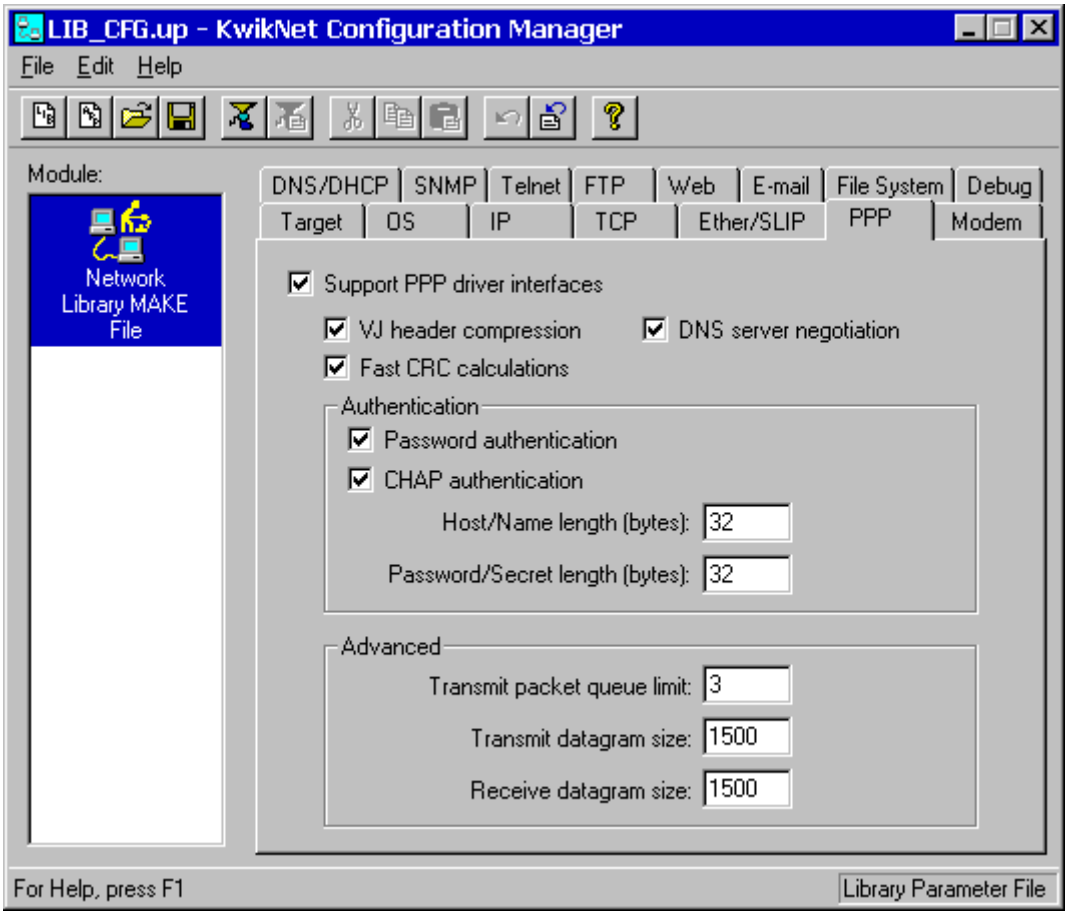
The purpose of this manual is to provide the system designer and applications programmer with the information required to properly configure and implement a networking system using the KwikNet TCP/IP Stack and PPP. It is assumed that you are familiar with the architecture of the target processor.

KwikNet and its options are available in C source format to ensure that regardless of your development environment, your ability to use and support KwikNet is uninhibited.

The C programming language, commonly used in real-time systems, is used throughout this manual to illustrate the features of KwikNet and its PPP option.

1.2 KwikNet PPP Library Configuration

You can readily tailor the KwikNet stack to accommodate your PPP needs by using the KwikNet Configuration Builder to edit your KwikNet Library Parameter File. The KwikNet PPP Library parameters are edited on the PPP property page. The layout of the window is shown below.



Support PPP

If you have one or more PPP networks with serial UART device drivers, check this box to force generation of the PPP Network Driver in a KwikNet PPP Library. Otherwise, leave this box unchecked.

PPP Library Parameters (continued)

VJ Header Compression

The KwikNet PPP option supports Van Jacobson (VJ) Compression, a technique for compressing TCP and IP headers, thereby improving transfer times over serial links if large numbers of small packets are being transferred. Check this box to enable VJ compression. Otherwise, leave this box unchecked.

Fast CRC Calculations

Check this box to enable faster CRC calculation. Although performance is improved, an additional 512 bytes of memory are used for a lookup table. To avoid this memory penalty, leave this box unchecked.

DNS Server Negotiation

Check this box to enable KwikNet to negotiate the DNS servers, if any, to be accessible on each PPP network interface. If you have not included a DNS client in your configuration, KwikNet will be unable to accept DNS servers proposed by its PPP peers but will still be able to present DNS server IP addresses for use by its peers. To preclude DNS server negotiation, leave this box unchecked.

Password Authentication

Check this box if any of your PPP networks must support the Password Authentication Protocol (PAP) to negotiate authorized use of the serial link. Otherwise, leave this box unchecked.

CHAP Authentication

Check this box if any of your PPP networks must support the Challenge-Handshake Authentication Protocol (CHAP) to negotiate authorized use of the serial link. Otherwise, leave this box unchecked.

Host/Name Length

This parameter serves two purposes. During PAP negotiation, it specifies the length (in bytes) of the longest host identification string (peer-ID) expected in any PAP authentication packet which will be received or sent on the PPP network. During CHAP negotiation, it specifies the length (in bytes) of the longest name expected in any CHAP authentication packet which will be received or sent on the PPP network. A value of 32 is typical.

PPP Library Parameters (continued)

Password/Secret Length

This parameter serves two purposes. During PAP negotiation, it specifies the length (in bytes) of the longest password expected in any PAP authentication packet which will be received or sent on the PPP network. During CHAP negotiation, it specifies the length (in bytes) of the longest CHAP secret expected in any CHAP authentication packet which will be received or sent on the PPP network. A value of 32 is typical.

Transmit Packet Queue Limit

The KwikNet PPP Network Driver queues packets for transmission until they can be accepted for transmission by the serial device driver. This parameter defines the maximum number of packets which the network driver will queue.

Transmit Datagram Size

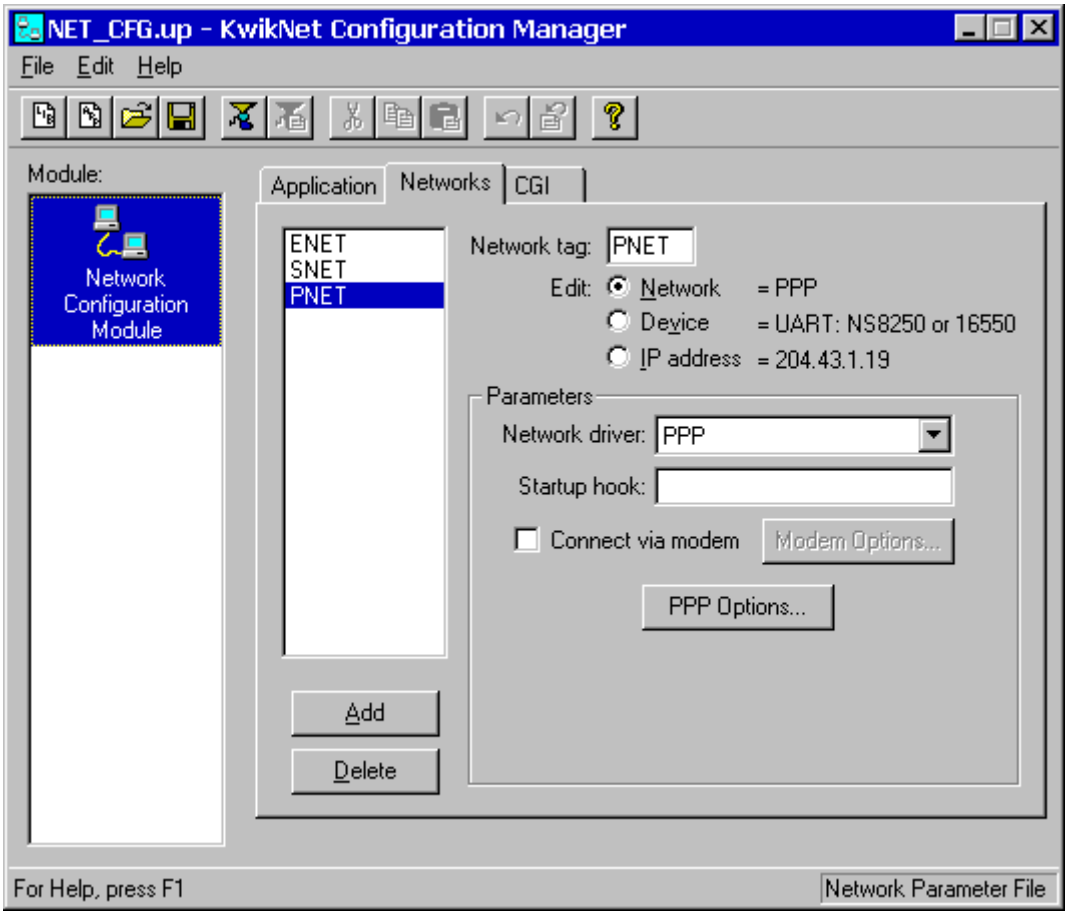
This parameter defines the size (in bytes) of the longest datagram which will be transmitted on a PPP network. The PPP network driver will not negotiate a transmit datagram size larger than this value. A typical value is 1500 bytes.

Receive Datagram Size

This parameter defines the size (in bytes) of the longest datagram which will be accepted from a PPP network. The PPP network driver will not negotiate a receive datagram size larger than this value. A typical value is 1500 bytes.

1.3 KwikNet PPP Network Definition

You must predefine each PPP network which your application must support. A separate definition is required for each such network. The total number of networks must not exceed the maximum number of networks which your KwikNet Library Parameter File allows. Each PPP network is defined using the Networks property page. The layout of the window is shown below.



Edit: Device

You must select the Edit: Device radio button to define the network's device driver parameters. You must select a serial device driver for use by the PPP network driver. This process is described in Chapter 2.4 of the KwikNet TCP/IP Stack User's Guide.

Edit: IP Address

You must select the Edit: IP Address radio button to define the network's IP address. This process is described in Chapter 2.4 of the KwikNet TCP/IP Stack User's Guide. This IP address will be used for the PPP network interface unless otherwise negotiated according to your PPP option settings.

PPP Network Definition (continued)

Tag

Each network must have a unique 4-character network tag. This parameter defines that tag. Although KwikNet does not restrict the content of the tag in a network description, the Configuration Manager only supports 4 ASCII characters as a tag.

Edit: Network

You must select the Edit: Network radio button to define the network parameters.

Network Driver

You must choose PPP from the pull down list to attach the KwikNet PPP Network Driver to your PPP network.

Startup Hook

This parameter provides the name of an application function which will be called when the network driver is being initialized. This function can modify the network's configuration parameters and IP address information before the network has been fully initialized. If your application does not require a startup hook for this network, leave this field empty. The PPP network driver startup hook is described in Chapter 1.4.

Modem Connection

The PPP network driver supports remote connections using the KwikNet Modem Driver. Check this box to attach the Modem Driver to this network. Otherwise, leave this box unchecked. The KwikNet Modem Driver is described in Chapter 1.7 of the KwikNet Device Driver Technical Reference Manual.

Note that the network still requires a device driver even if the Modem Driver is used.

Modem Options

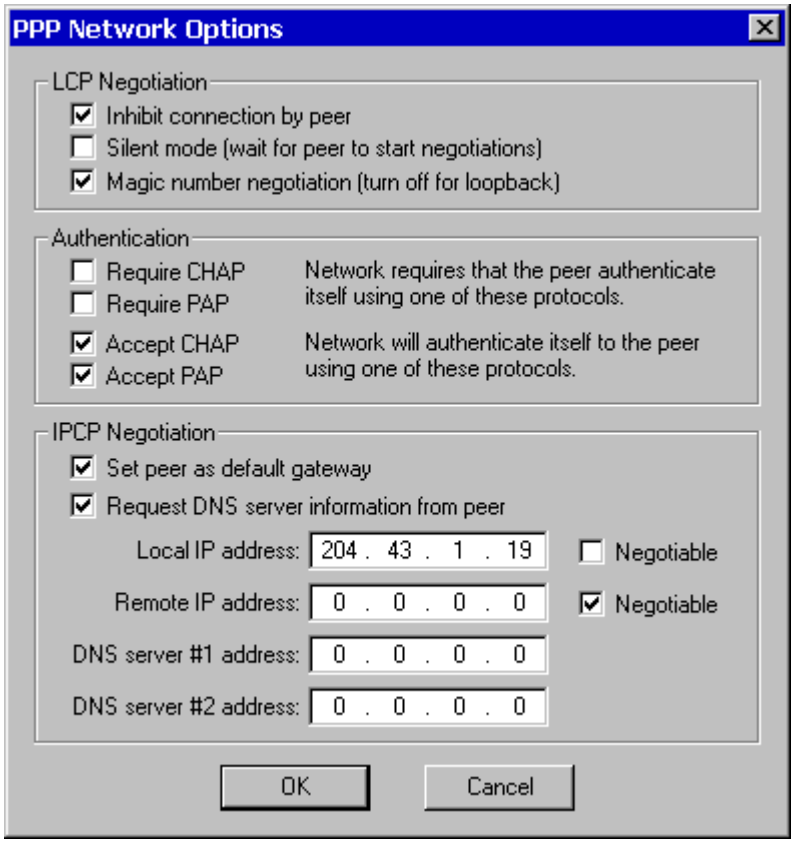
If you have attached the Modem Driver to this network, then click the Modem Options... button to open the Modem Options Dialog. Within this dialog you can configure the modem to meet your requirements. The modem dialog is described in Chapter 2.4 of the KwikNet TCP/IP Stack User's Guide.

PPP Options

Click on the PPP Options... button to open the PPP Options Dialog. Within this dialog you can adjust LCP, authentication and IPCP parameters to meet your PPP requirements. Note that these PPP option settings are for one particular PPP network interface. Different PPP networks can have different option settings.

PPP Options

For each KwikNet network which supports PPP, you must define the PPP parameters which govern its use. These PPP parameters are edited using the PPP Dialog Box entered via the PPP Options... button on the Networks definition property page. The layout of the dialog box is shown below.



LCP Negotiation Options

Inhibit Connection by Peer

Check this box if the network's peer is not allowed to initiate a PPP connection. If this box is checked, only the local network will be able to initiate the establishment of a PPP connection with its peer. In most systems you should leave this box unchecked so that either peer can start the PPP connection process.

PPP Options (continued)

Silent Mode

Check this box to force the local network to operate in silent mode. Otherwise, leave this box unchecked.

When in silent mode, the local network will not initiate LCP negotiation with its peer. Instead, the network will await the initiation of the LCP negotiation sequence by the peer.

Note that only one of the peers on a PPP network should operate in silent mode. If both peers operate silently, then LCP negotiations will never start and a PPP connection will never be established.

Magic Number Negotiation

Check this box if magic numbers are to be used during LCP negotiation. Otherwise, leave this box unchecked. Be sure to leave this box unchecked if you are using a loopback device driver.

Authentication Options

Require CHAP or PAP

If this network must authenticate its peer using CHAP or PAP, check the box to indicate that CHAP or PAP is required. If both boxes are checked, the network will first try to authenticate its peer using CHAP. If CHAP is not supported by the peer, PAP will be used. If no peer authentication is required, leave both boxes unchecked.

Accept CHAP or PAP

If this network supports authentication by its peer, check the box to indicate that CHAP or PAP authentication requests will be accepted. If both boxes are checked, the network will accept authentication requests made using either CHAP or PAP. If the network does not support requests from its peer for authentication, leave both boxes unchecked.

IPCP Negotiation Options

Default Gateway

The default gateway for each PPP network is usually provided in the network's IP address definition accessed via the Networks property page. Check this box if the IP address of network's peer is to be used as the default gateway for the network instead of the value provided in the network's definition. Otherwise, leave this box unchecked.

Request DNS Server

Check this box if you will accept one or two DNS server IP addresses from the network's peer for use by your DNS client. Otherwise, leave this box unchecked. If you check this box, be sure to enable PPP DNS server negotiation and include the KwikNet DNS client.

PPP Options (continued)

Local IP Address

The IP address for each PPP network is usually provided in the network's IP address definition accessed via the Networks property page. For convenience, the network IP address is repeated on the PPP Options dialog.

PPP allows the local IP address to be negotiated with the network's peer. If you wish to allow such negotiation, check the box labeled Negotiable.

If the local IP address is 0.0.0.0, then the local IP address must be negotiated with the peer, whether or not the Negotiable box is checked. In this case, the peer must provide the local IP address if the network is to be operable.

If a local IP address is provided, it will be proposed as the network's IP address in the initial stages of any negotiation. However, if the Negotiable box is checked and an alternate IP address is suggested by the peer, it will be accepted for local use.

If a local IP address is provided and the Negotiable box is unchecked, then no negotiation of the local IP address will be permitted. The IP address which you defined will be used as the local IP address.

Once the local IP address has been set and a link established, the IP address is still subject to change via DHCP if permitted by your network definition.

Remote IP Address

PPP allows the IP address of the network's peer to be negotiated with the peer, if permitted by the peer. If no such negotiation is required, set the remote IP address to 0.0.0.0 in the space provided and leave the Negotiable box unchecked.

If a remote IP address is provided, it will be presented for acceptance by the peer in the initial stages of any negotiation. However, if the Negotiable box is checked, the peer can reject the proposal and the IP address of the peer will be as defined by the peer.

If a remote IP address is provided and the Negotiable box is unchecked, then the peer must accept your proposed IP address if the network is to become operable. The IP address which you provide will unconditionally be used as the peer IP address.

If the peer asks to negotiate its IP address and the defined remote IP address is 0.0.0.0, then the local IP address plus one will be proposed for use by the peer.

DNS Server #1 and #2 IP Address

If you have enabled PPP DNS server negotiation, KwikNet can send the IP address of one or two DNS servers to the network's peer for use by that peer. You can enter the DNS server IP address in either of the spaces provided. An IP address of 0.0.0.0 will not be presented to the peer during DNS negotiation.

1.4 PPP Startup Hook

When the KwikNet TCP/IP Stack begins execution, each PPP network driver initializes the networks for which it is responsible. The PPP network driver receives a pointer to an application network startup hook, if one exists. The hook is a function which the driver can call to give the application an opportunity to alter the driver's operating characteristics. The startup hook can dynamically adjust network parameters to adapt to the need of the particular application.

The name of the startup hook function, if any, is provided by you in the network definition on the Networks property page of your Network Parameter File.

Each PPP network definition in your Network Parameter File can specify a PPP network startup hook for that network.

Every PPP network startup hook receives a pointer to a network parameter structure which contains network parameters ready for use by the PPP network driver. It is these parameters which can be modified by the startup hook.

Every network parameter structure begins with a structure containing common network address information. The structure *knx_nainfo* is defined in file *KN_API.H* as follows:

```
struct knx_nainfo {
   struct in_addr xna_ipaddr;    /* Network IP address            */
   struct in_addr xna_snmask;    /* Network subnet mask           */
   struct in_addr xna_defgate;   /* Network default gateway       */
   short          xna_dhcp;      /* Use DHCP (0/1 = no/yes)       */
   short          xna_rsv1;      /* Reserved for alignment        */
   };
```

Purpose **PPP Network Startup Hook**

Context ■ KwikNet Task □ Application Task

Setup Structure definitions are in file *KN_API.H*.

```
#include "KN_LIB.H"
int net_hook(KN_TYTAG nettag, void *paramp);
```

The startup hook function calling method is equivalent to:

```
struct knx_ppp_prep params;
KN_TYTAG    tag;
result = net_hook(tag, &params);
```

Description *Nettag* is the 4-character tag assigned to the network of interest.

Paramp is a pointer to a network parameter structure. This structure contains network parameters ready for use by the network driver. These parameters can be modified by your startup hook. The structure *knx_ppp_prep* is defined in file *KN_API.H* as follows:

```
struct knx_ppp_prep {
                                /* Network address information */
  struct knx_nainfo xppp_nainfo;
  unsigned long  xppp_lcpopt;   /* LCP configuration options  */
  unsigned long  xppp_ipcpopt;  /* IPCP configuration options */
  struct in_addr xppp_ipremote; /* Default remote IP address  */
  struct in_addr xppp_ipdns1;   /* Primary DNS IP address     */
  struct in_addr xppp_ipdns2;   /* Secondary DNS IP address   */
  };
```

Every network parameter structure begins with a structure containing common network address information. The structure *knx_nainfo* has been described in the introduction of this chapter.

Structure member *xppp_lcpopt* specifies the LCP negotiation options which were defined for the network in your Network Parameter File. This parameter is the logical OR of any of the following bit masks which are defined in header file *KN_API.H*.

KN_PPP_PASSIVE	Wait for peer if peer not yet ready for negotiation.
KN_PPP_RESTART	Renegotiate when physical link is established.
KN_PPP_SILENT	Wait for peer to initiate negotiation.
KN_PPP_MAGIC	Use magic numbers during LCP negotiation.
KN_PPP_RCHAP	Require CHAP to authenticate peer.
KN_PPP_RPAP	Require PAP to authenticate peer.
KN_PPP_OCHAP	Offer CHAP when authenticated by peer.
KN_PPP_OPAP	Offer PAP when authenticated by peer.

...more

Description ...continued

Structure member *xppp_ipcpopt* specifies the IPCP negotiation options which were defined for the network in your Network Parameter File. This parameter is the logical OR of any of the following bit masks which are defined in header file *KN_API.H*.

KN_PPP_DEFROUTE	Use peer IP address for default gateway.
KN_PPP_NEGLOCAL	Local IP address is negotiable.
KN_PPP_NEGREMOTE	Remote IP address is negotiable.
KN_PPP_REQDNS	Request DNS server addresses.
KN_PPP_SUPDNS	Supply DNS server addresses.

Structure member *xppp_ipremote* defines the default remote IP address to be used when negotiating the peer's IP address.

Structure members *xppp_ipdns1* and *xppp_ipdns2* provide the DNS server IP addresses to be supplied to the PPP network peer when mask bit *KN_PPP_SUPDNS* in option parameter *xppp_ipcpopt* is set. An IP address of *0* will not be sent to the peer.

Returns If successful, the value *0* is returned.
Otherwise, the value *-1* is returned.

Example

```
#include "KN_LIB.H"

int net_hook(KN_TYTAG nettag, void *paramp)
{
   struct knx_ppp_prep *pp =
          (struct knx_ppp_prep *)paramp;

   /* Do not allow DHCP to adjust the local IP address     */
   pp->xppp_nainfo.xna_dhcp = 0;

   /* Use magic numbers during LCP negotiation             */
   pp->xppp_lcpopt |= KN_PPP_MAGIC;

   /* Do not use PAP during any LCP negotiation            */
   pp->xppp_lcpopt &= ~(KN_PPP_RPAP | KN_PPP_OPAP);

   return (0);
   }
```

1.5 PPP Authentication Parameters

If any of your PPP networks require PAP or CHAP authentication, you must provide the KwikNet PPP network driver with access to the PAP and CHAP authentication parameters needed to support these protocols. These parameters are the heart of your PPP security system. The maintenance and safekeeping of these parameters is your responsibility.

The KwikNet PPP authentication parameters for each PPP network are located in array *kn_ppp_auth[]* in module *PPP_AUTH.C* located in the KwikNet *PPP* installation directory. The authentication parameters are specified in a *knx_ppp_admin* structure which is defined in module *PPP_AUTH.C*.

```
struct knx_ppp_admin {              /* PPP administration structure */
   KN_TYTAG         xad_tag;        /* PPP network tag              */
   struct knx_authn *xad_lpname;    /* PAP local name               */
   struct knx_authn *xad_lppwd;     /* PAP local password           */
   struct knx_authn *xad_ppname;    /* PAP peer name                */
   struct knx_authn *xad_pppwd;     /* PAP peer password            */
   struct knx_authn *xad_lcname;    /* CHAP local name              */
   struct knx_authn *xad_lcsecret;  /* CHAP local secret            */
   struct knx_authn *xad_pcname;    /* CHAP challenge name          */
   struct knx_authn *xad_pcsecret;  /* CHAP peer secret             */
   };
```

The network tag is provided to identify the PPP network to which the authentication parameters apply.

Each PAP name and password and CHAP name and secret is provided in a *knx_authn* structure which is defined in module *PPP_AUTH.C*. The structure provides the length of the authentication parameter and a pointer to the character array which constitutes the parameter.

```
struct knx_authn {                  /* PPP authentication parameter */
   short int   xad_length;          /* Length of parameter          */
   short int   xad_rsv1;            /* Reserved for alignment       */
   const char  *xad_value;          /* Pointer to parameter value   */
   };
```

File *PPP_AUTH.C* provides the authentication parameters for one user on a single PPP network. You must edit this module to define the authentication parameters which apply to each of your PPP networks.

The KwikNet PPP network driver accesses these authentication parameters by calling the KwikNet PPP administration function *kn_ppp_admin()* within module *PPP_AUTH.C*. You must review this function and, if necessary, alter it to meet your specific requirements. A description of this function is presented later in this chapter.

Security Issues

Since the PPP administration function *kn_ppp_admin()* is the funnel through which all PPP authentication parameters flow, you may wish to enhance this function to include security features to prevent unauthorized access to this critical information. For example, your authentication parameters might be derived by reading the magnetic strip on an authorization card.

If your network supports both local and peer authentication, you should use different authentication parameters for the local network and its peer. If one set of parameters is used to authenticate both, then the security provided by PPP is trivial to crack.

If your network supports both CHAP and PAP authentication, you should use different authentication parameters for CHAP and PAP. Since PAP sends the password as unencrypted text, a "rogue peer" could negotiate PAP to obtain the password and then use the password as the secret for subsequent CHAP negotiations with a more secure authenticator.

For more information on PPP security issues, refer to pages 66-68 in Carlson's book *PPP Design and Debugging* (see Appendix A of the KwikNet TCP/IP Stack User's Guide).

Purpose **Provide PPP Authentication Services**

Context ■ KwikNet Task □ Application Task

Setup Prototype and definitions are in file *KN_API.H*.
Structure definitions are in file *PPP_AUTH.C*.

```
#include "KN_LIB.H"
int kn_ppp_admin(KN_TYTAG nettag, int opcode,
            const char *instr, int inlen,
            char *outbuf, int buflen);
```

Description *Nettag* is the 4-character tag assigned to the network of interest.

Opcode is one of the following operation codes:

KN_PPPAUTH_LPNAME	PAP: get local name
KN_PPPAUTH_LPPWD	PAP: get local password
KN_PPPAUTH_PPPWD	PAP: get peer password
KN_PPPAUTH_PPCHECK	PAP: authenticate peer
KN_PPPAUTH_LCNAME	CHAP: get local name
KN_PPPAUTH_LCSECRET	CHAP: get local secret
KN_PPPAUTH_PCNAME	CHAP: get challenge name
KN_PPPAUTH_PCSECRET	CHAP: get peer secret

Instr is a pointer to a character array needed to process the request specified by *opcode*. If no such parameter is required, *instr* will be *NULL*.

Inlen is the length of the character array, if any, referenced by parameter *instr*.

Outbuf is a pointer to a character array used to process the request specified by *opcode*. For most operations, *outbuf* is a pointer to storage for the authentication parameter being returned to the caller. If *opcode* is *KN_PPPAUTH_PPCHECK*, *outbuf* is a pointer to an additional input parameter.

Buflen is the length of the character array referenced by parameter *outbuf*.

...more

Description ...continued

Interpretation of parameters *instr* and *outbuf* for each value of *opcode* is as follows.

For operation code ***KN_PPPAUTH_LPNAME***, *instr* is unused. The local PAP name is returned in the buffer at **outbuf* for delivery to the peer.

For operation code ***KN_PPPAUTH_LPPWD***, *instr* is unused. The local PAP password is returned in the buffer at **outbuf* for delivery to the peer.

For operation code ***KN_PPPAUTH_PPPWD***, *instr* is a pointer to the PAP name provided by the peer for authentication. The peer PAP password is returned in the buffer at **outbuf* for eventual comparison with the password provided by the peer.

For operation code ***KN_PPPAUTH_PPCHECK***, *instr* is a pointer to the PAP name and *outbuf* is a pointer to the PAP password provided by the peer for authentication. These parameters must be checked for validity and the result of the check must be returned to the caller. The input parameters at **instr* and **outbuf* must not be modified.

For operation code ***KN_PPPAUTH_LCNAME***, *instr* is a pointer to the CHAP name provided by the peer to identify itself. The local CHAP name is returned in the buffer at **outbuf* for delivery to the peer.

For operation code ***KN_PPPAUTH_LCSECRET***, *instr* is a pointer to the CHAP name provided by the peer to identify itself. The local CHAP secret is returned in the buffer at **outbuf*. It will be used to encrypt the response to the peer's challenge.

For operation code ***KN_PPPAUTH_PCNAME***, *instr* is unused. The challenge CHAP name is returned in the buffer at **outbuf* for delivery to the peer in the authentication challenge.

For operation code ***KN_PPPAUTH_PCSECRET***, *instr* is a pointer to the CHAP name provided by the peer to identify itself. The peer CHAP secret is returned in the buffer at **outbuf*. It will be used to encrypt the challenge which was sent to the peer for comparison with the challenge response from the peer.

...more

...continued

Returns

For operations which return an authentication parameter in the buffer at **outbuf*, an integer *n* is returned. The value of *n* reflects the success or failure of the request.

n >= 0	Length of authentication parameter stored at **outbuf*
n = -1	Authentication parameter is not available
n = -2	Insufficient storage at **outbuf* for authentication parameter

For operation code *KN_PPPAUTH_PPCHECK*, an integer *n* is returned which reflects the success or failure of the validity check requested.

n = 1	The peer's PAP name and password are valid.
n = 0	The peer's PAP name and password are NOT valid.
n = -1	Peer PAP name and password validation is not supported by this function. The PPP network driver will use operation code *KN_PPPAUTH_PPPWD* to fetch the password and then do the comparison itself.

1.6 Adding PPP to Your Application

Before you add the PPP protocol to your application, you must have a working KwikNet IP stack (with or without TCP) operating with your RT/OS and your serial device driver. It is imperative that you start with a tested TCP/IP stack with functioning device drivers before you add PPP. If these components are not operational, the KwikNet PPP option cannot operate correctly.

KwikNet PPP Library

Begin by deciding which PPP features must be supported. Review the PPP property page described in Chapter 1.2. In particular, omit support for VJ compression, fast CRC calculation and authentication unless you actually have such a need. The memory savings are significant.

If any of your PPP networks must support PAP or CHAP authentication, edit file *PPP_AUTH.C* in KwikNet directory *PPP* as described in Chapter 1.5 to provide access to your administration function *kn_ppp_admin()*.

Armed with your PPP feature list, use the KwikNet Configuration Manager to edit your application's KwikNet Library Parameter File to include the PPP protocol. Then rebuild your KwikNet Libraries. A new PPP Library, *KNnnnPPP.A*, will be produced along with your IP Library. Of course a TCP Library will also be generated if you have included the TCP protocol in your configuration. The library extension may be *.A* or *.LIB* or some other extension dictated by the toolset which you are using.

KwikNet Network Configuration Module

Use the KwikNet Configuration Manager to edit your application's KwikNet Network Parameter File to define each of your PPP network interfaces. Then rebuild your KwikNet Network Configuration Module.

Reconstructing Your KwikNet Application

If you have not already done so, add your network serial device drivers to your application. Start by reviewing your KwikNet board driver module *KN_BOARD.C*. Unless you have modified the board driver, it supports four network interrupt sources. If necessary, increase the number of supported network interrrupt sources to allow support for your network serial device drivers. If you are porting KwikNet for use with your choice of RT/OS, your KwikNet OS Interface Module *KN_OSIF.C* must be updated to support these additional network interrupt sources. If changes are made, be sure to rebuild your KwikNet Libraries and compile board driver module *KN_BOARD.C*.

Since your KwikNet Network Parameter File now includes your PPP network definitions, do not forget to rebuild and compile your KwikNet Network Configuration Module.

Your application link and/or locate specification files must be updated to add the KwikNet PPP Library file *KNnnnPPP.A* prior to the KwikNet IP Library. The object modules for your network serial device drivers must also be included in your link specification together with your other application object modules.

With these changes in place, you can link and create an updated KwikNet application with PPP support included.

AMX Considerations

When reconstructing a KwikNet application which uses the AMX Real-Time Multitasking Kernel, adapt the procedure just described to include the following considerations.

You may have to edit your AMX User Parameter File to increase the size of the AMX Interrupt Stack and the KwikNet Task stack to meet the needs of the PPP protocol. You must then rebuild and compile your AMX System Configuration Module.

No changes to your AMX Target Configuration Module are required to support PPP provided that your network serial device drivers are already part of your application.

If the KwikNet board driver module *KN_BOARD.C* is altered to increase the number of supported network interrrupt sources, your AMX Target Parameter File must be edited to add the corresponding ISP definitions. If changes are made, be sure to rebuild and compile your AMX Target Configuration Module and compile board driver module *KN_BOARD.C*. When using KwikNet with AMX, there is no need to edit the KwikNet OS Interface Module *KN_OSIF.C*.

This page left blank intentionally.