SII
Seiko Instruments Inc.

Functional Specification (Preliminary)

S-7600A

TCP/IP Network Protocol LSI

Components Marketing Dept.
Marketing Section 2

Tel +81-43-211-1028
Fax +81-43-211-8035

8,Nakase 1-chome, Mihama-ku
Chiba-shi, Chiba 261-8507 Japan

Seiko Instruments Inc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1. General Description

This document contains the functional specification for the S-7600A. S-7600A is the code name for the Seiko Instruments Network Stack chip, which includes the PPP, TCP, IP and UDP protocol. The S-7600A allows various products to be networked giving them the ability to share/sync data between devices.

1.2. S-7600A Feature Set

The S-7600A features include:

- Industry standard protocols. PPP, TCP, IP, UDP are the de facto protocols that are currently used in industry for Internet communication.
- Platform – OS and processor independent
- Highly efficient and portable APIs

1.3. Benefits of S-7600A

The S-7600A benefits include:

- Off loads MIPs allowing systems to operate with low end, low cost processors
- Consumes minimal power, up to 1/100 of competing solution

1.4. Document Organization

The second chapter shows a system level block diagram of the S-7600A followed by a high level view of the core. Chapters 3 through 5 provide detailed explanations of operation, interfacing, and other information about the module blocks that comprise the core.

2. Top Level Architecture

2.1. Overview

This chapter provides a high level description of the S-7600A system. It also provides main register interface descriptions, theory of operations, and verification documentation.

2.1.1. Definitions

●	IP	Internet Protocol
●	PPP	Point-to-Point Protocol
●	TCP	Transmission Control Protocol
●	UDP	User Datagram Protocol
●	API	Application Programming Interface

2.1.2. Applicable Documents

- Hardware Specification for S-7600A

2.2. Functional Block Diagram

Figure 2-1 shows a functional block diagram of the S-7600A. There are blocks of the Network Stack and other functions related to it. The S-7600A has the interface for a host MPU and a Physical layer for various data terminal equipment.

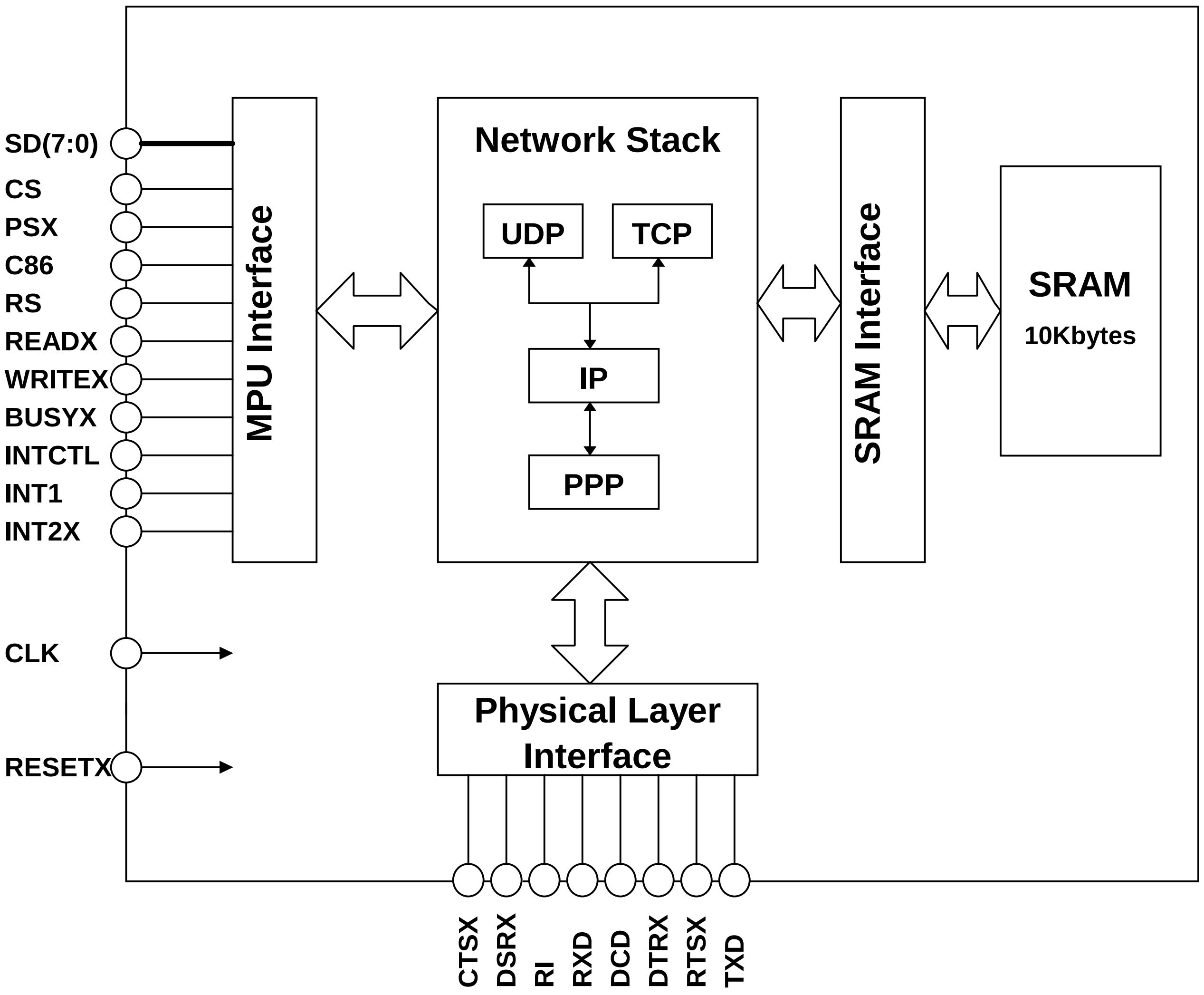


Figure 2-1 Block diagram

The transport and network layers contain:

- Two general sockets that provide connectivity between the application layer and the transport layer.
- The TCP/UDP module that allows for reliable (retransmission) and unreliable (no retransmission) datagram deliveries.
- An IP module that provides connectionless packet delivery.
- The PPP module that provides point-to-point connection link between two peers.

3. TCP/UDP Module

3.1. Overview

TCP is the transmission layer protocol used in Internet communications. It sits between the application layer and a data network layer (IP). TCP is a reliable and connection-oriented protocol that provides error checking, acknowledgment and reordering of received segments. UDP is fast, however is a connection-less and not reliable (does not support retransmission) oriented protocol. TCP/UDP module serves three main purposes:

1. Data Encapsulation – The TCP/UDP layer encapsulates application data by placing a TCP/UDP header in front of it.

2. Error Checking – TCP and UDP uses checksum to check the validity of the data received. It also generates the checksum for entire outbound TCP segment/UDP datagram.

3. Handshaking/Flow Control – TCP establishes reliable handshaking and controls the data flow by using the sequence number and the window advertisement. UDP is connectionless protocol that does not support data flow.

The features of S-7600A's TCP implementation include:

- One pass TCP interpretation.
- TCP checksum.
- Socket slice architecture.
- TCP server mode support
- Support for incoming maximum segment size option
- Automatic source port increment

The features of S-7600A's UDP implementation include:

- One pass UDP interpretation.
- UDP checksum.
- Socket slice architecture.
- UDP daemon mode support
- Automatic source port increment

The TCP/UDP can have multiple socket interfaces as shown in Figure 3-1.

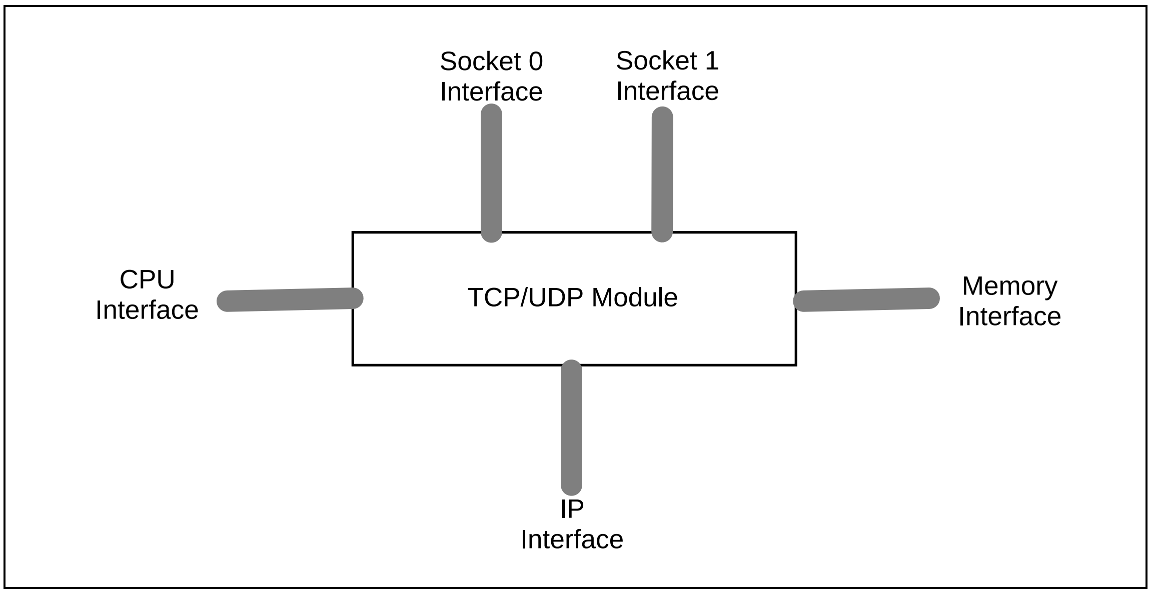


Figure 3-1 TCP/UDP Module Block Diagram

3.1.1. *Definitions*

●	IP	Internet Protocol
●	ICMP	Internet Control Message Protocol
●	TCP	Transmission Control Protocol
●	UDP	User Datagram Protocol

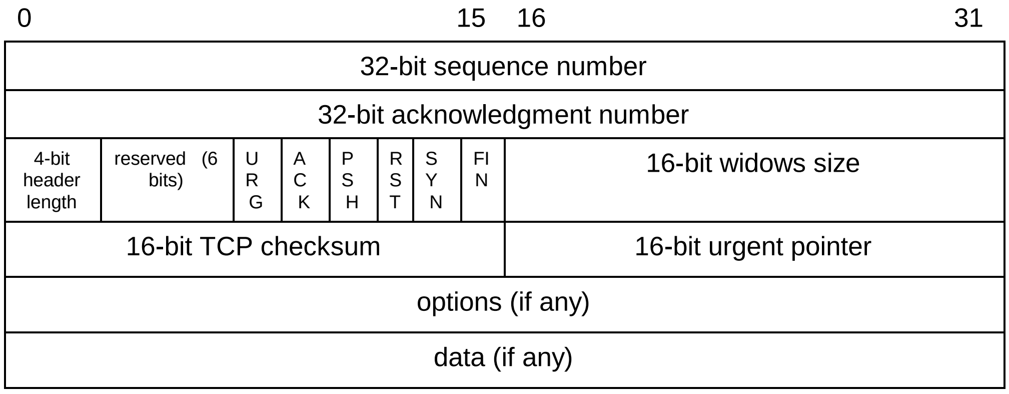
3.1.2. *Applicable Documents*

●	*TCP/IP Volume 1*	Douglas E. Comer 1995
●	*Transmission Control Protocol*	University of Southern California 1981
●	*Requirements for Internet Hosts*	R. Branden, Oct. 1989
●	*NS-1000 Datasheet*	iReady Corporation document

3.2. Functional Description

3.2.1. Segment Description

Figure 3-2 TCP Segment Diagram



3.2.1.1. Source and Destination Port Number

The TCP module uses the source and destination port numbers to determine the actions taken by the TCP receiver. The action list is summarized in the Table 3-1 and Table 3-2.

Table 3-1 TCP Client Receiver Actions

Source Port (remote)	Destination Port (local)	Action
No match	No match	Send RST to source
No match	Match	Send RST to source
Match	No match	Send RST to source
Match	Match	Send segment to application

Table 3-2 TCP Server Receiver Actions

Source Port (remote)	Destination Port (local)	Action
No match	No match	Send RST to source
No match	Match	Makes TCP connection if not connected. If already connected, send RST
Match	No match	Send RST to source
Match	Match	Send segment to application

The TCP hardware does not support concurrent server function by itself. This means that once a TCP connection is made with a client, the socket will not allow new connections with another client until the current connection is terminated. However a concurrent server should be designed with the application level layer using multiple sockets.

3.2.1.2. Sequence Number

This 32-bit field indicates to the TCP receiver whether the incoming segment is to be stored in the TCP receive buffer. If the sequence number matches the receive buffer's next expected sequence number, it stores the data to the buffer. Otherwise, the TCP receiver ignores the incoming segment. This scheme works even if the received sequence number arrives out of order because the remote TCP module guarantees to resend unacknowledged segments.

When the TCP module makes a connection, the initial sequence number is derived from a random number generator in order to avoid the confusion with earlier connections. After a connection is established, the sender module fills this field with the last acknowledge number received from the TCP peer.

3.2.1.3. Acknowledge Number

The TCP receiver checks this number to disable re-sending data. If the received acknowledge number is larger than the last received number and same or smaller than what was expected, the internal acknowledge number is updated.

The TCP sender fills this field with a valid acknowledge number except for when it first requests a TCP connection as a client. The acknowledge number that will be sent is incremented by the number of data received. It is also incremented in response to SYN, FIN without data because they have virtually one byte worth of data in TCP sequence / acknowledge scheme.

The TCP receiver checks both SEQ and ACK numbers to validate the incoming TCP segments.

3.2.1.4. Header Length

This 4-bit header length field contains an integer that specifies the length of the segment header measured in 32-bit multiples. The TCP receiver state machine uses this field to determine if TCP options exist. Since S-7600A's TCP does not send any TCP options, the TCP sender module always write 01012 to this field.

The TCP receiver checks this field when it receives SYN flag. When this flag is other than 0x50, it assumes that the current TCP segment contains options.

3.2.1.5. Reserved

This 6-bit field is reserved. The TCP receiver module does not check this field. When this field is filled other than zeros, TCP checksum module (actually in IP module) counts that as a part of TCP checksum in order to support misbehaving peers. The TCP sender module always fills this field with zeros.

3.2.1.6. Code

The next 6-bits specifies codes. The codes are summarized in Table 3-3.

Table 3-3 Code Specification

Bit (left to right)	Meaning (if bit set to 1)
URG	Urgent pointer field is valid
ACK	Acknowledgment field is valid
PSH	This segment requests a push
RST	Reset the connection
SYN	Synchronize sequence number
FIN	Sender has reached end of its byte stream

The URG bit, if set, indicates that the incoming TCP segment's urgent pointer is valid. The S-7600A TCP receiver supports UNIX style urgent pointer scheme which treats urgent pointer to be only one byte long. It is capable of queuing up to two occurrences of urgent pointers. The S-7600A TCP never sets URG bit upon send.

The TCP receiver uses ACK flag to check the acknowledge field. If the corresponding acknowledge number is good this number is passed to the sender module which disables resending the old data. If the receiver receives three consecutive ACK segments without any data. The TCP sender sends a TCP segment even when there is no data to be sent. This feature helps the peer to receive the receiver's new window. The TCP sender leaves this bit on except when it sends SYN as a client. If sender has no TCP data to send but it need to acknowledge, ACK is sent without data.

The PSH bit is currently ignored by the TCP receiver. The TCP sender turns on this bit when there is data to be sent (or resent). This tells the application that the received data has to be processed right a way. The TCP sender module turns this bit on for every TCP segment with data field.

The TCP receiver notifies application when RST flag is received. The S-7600A TCP module does not take any actions other than notifying the reception of this flag because the response to this is up to the higher level layer.

The SYN flag is used for the TCP module to initiate a TCP connection. If this flag is received after the connection is established the segment is ignored.

The FIN flag is used to terminate connections. The S-7600A TCP module supports all cases of closing which are client initiated, server initiated and concurrent closing.

3.2.1.7. Windows Size

This 16-bit field is used to advertise how many bytes are reserved for receiving incoming TCP data. After the TCP receiver reads this value, the TCP sender limits the number of byte to be sent if this field indicates smaller number of data than what needs to be sent. The TCP sender also inserts the number of acceptable byte to this field on every TCP segment it sends out.

3.2.1.8. Checksum

This 16-bit field is always checked for incoming TCP segments. The TCP sender always calculates this field before it passes the segment to the IP module.

3.2.1.9. Urgent Pointer

This 16-bit field, upon the reception of URG flag, is used to determine where the urgent pointer resides in the TCP data. This pointer value is translated by TCP module so that the application layer receives this data as a receive buffer offset value. S-7600A TCP module can handle up to two consecutive urgent pointer receptions. The TCP send module never sends urgent pointers.

3.2.1.10. TCP Options

The only option S-7600A module accepts is Maximum Segment Size option only as it receives SYN flag. This is detected by the TCP header field being larger than 0x50. The MSS option is listed in Table 3-4. Any other option received is ignored. The TCP sender never sends option which implies that the MSS for the TCP module is 536 bytes.

Table 3-4 MSS Option

Byte 0	Byte 1	Byte 2	Byte 3
Option Code	Length in bytes	MSB	LSB
0 x 02	0 x 04	MSB	LSB

3.2.1.11. TCP Data

The TCP data is passed to the application layer. Because TCP data is stream data, the applications do not see the break in the data stream. The application keeps the TCP sending and receiving buffer from overflow. If the receive buffer overflows, the incoming TCP data is ignored but the TCP module still maintains TCP connections appropriately.

3.2.2. TCP State Transition

The TCP module, Figure 3-3, supports all TCP state as described in RFC 798.

Figure 3-3 TCP State Diagram

```
                              +---------+ ---------\      active OPEN
                              |  CLOSED |            \    -----------
                              +---------+<---------\   \   create TCB
                                |     ^              \   \  snd SYN
                   passive OPEN |     |   CLOSE        \   \
                   ------------ |     | ----------       \   \
                    create TCB  |     | delete TCB         \   \
                                V     |                      \   \
                              +---------+            CLOSE    |    \
                              |  LISTEN |          ---------- |     |
                              +---------+          delete TCB |     |
                   rcv SYN      |     |     SEND              |     |
                  -----------   |     |    -------            |     V
 +---------+      snd SYN,ACK  /       \   snd SYN          +---------+
 |         |<-----------------           ------------------>|         |
 |   SYN   |                    rcv SYN                     |   SYN   |
 |   RCVD  |<-----------------------------------------------|   SENT  |
 |         |                    snd ACK                     |         |
 |         |------------------           -------------------|         |
 +---------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +---------+
   |           --------------   |     |   -----------
   |                  x         |     |     snd ACK
   |                            V     V
   |  CLOSE                   +---------+
   | -------                  |  ESTAB  |
   | snd FIN                  +---------+
   |                   CLOSE    |     |    rcv FIN
   V                  -------   |     |    -------
 +---------+          snd FIN  /       \   snd ACK          +---------+
 |  FIN    |<-----------------           ------------------>|  CLOSE  |
 | WAIT-1  |------------------                              |   WAIT  |
 +---------+          rcv FIN  \                            +---------+
   | rcv ACK of FIN   -------   |                            CLOSE  |
   | --------------   snd ACK   |                           ------- |
   V        x                   V                           snd FIN V
 +---------+                  +---------+                   +---------+
 |FINWAIT-2|                  | CLOSING |                   | LAST-ACK|
 +---------+                  +---------+                   +---------+
   |                rcv ACK of FIN |                 rcv ACK of FIN |
   |  rcv FIN       -------------- |    Timeout=2MSL -------------- |
   |  -------              x       V    ------------        x       V
    \ snd ACK                 +---------+delete TCB         +---------+
     ------------------------>|TIME WAIT|------------------>| CLOSED  |
                              +---------+                   +---------+
```

Because the TCP hardware supports all states if an application needs to reuse the socket shortly after it sends connection close request, it needs to reset the socket first. This forces the TCP State to be in a Closed State even if the previous TCP connection is not successfully closed. Because the hardware socket number is limited, this method is OK. The side effect of this scheme is that it leaves half-closed socket on the peer's side for a period of time.

The TCP hardware supports the server mode. When it is in the Listen State, the TCP receiver disables the remote socket-matching feature. The TCP hardware supports the half -closed state whether the half-close is initiated from local end or the remote end.

3.2.3. *UDP Datagram Description*

Figure 3-4UDP Segment Diagram

0 15	16 31
16-bit source port number	16-bit destination port number
16-bit UDP length	16-bit UDP checksum
data (if any)	

3.2.3.1. *Source and Destination Port Number*

The source and the destination port numbers are used by the UDP module to determine UDP receiver module's action. The action list is summarized in Table 3-5.

Table 3-5 UDP Receiver Actions

Source Port (remote)	Destination Port (local)	Action
No Match	No Match	Ignore the Datagram
No Match	Match	Send Datagram to Application
Match	No Match	Ignore the Datagram
Match	Match	Send Datagram to Application.

The UDP receiver does not care the source port number upon datagram receptions since it is connectionless protocol. However, the UDP receiver has a way to pass the source port and IP address information to the application level which is useful for writing UDP-based daemons. The S-7600A UDP module and IPMP module do not support "Port Unreacheable" feature so that mis-delivered UDP datagram is merely discarded.

The UDP sender gets the destination information and uses this number to fill the destination field.

3.2.3.2. *UDP Length*

This 16-bit field indicates to the UDP receiver the size of the UDP data. The UDP module extracts the UDP data length based on this field and only receives the correct amount of data. If the IP module indicates different number of UDP data, the UDP data disregard the UDP datagram.

3.2.3.3. *UDP Checksum*

The IP module calculates UDP checksum if the field is not zero. The IP module disables checksum checking if it is zero since the UDP checksum is optional. The S-7600A UDP sender always utilizes the UDP checksum feature.

3.2.3.4. UDP Data

The UDP data is passed to the application layer. Because UDP data is not stream data, the applications sometimes have to know which block each bulk of data belong to. The S-7600A hardware solves this in two different modes.

The first mode allows the application to read the length of current data. This length information is queued up to four blocks. If an application chooses to use this option, there is no way to know the IP addresses and ports of remote senders.

The second mode allows the application to know the remote sender's IP and port numbers. It is done by embedding this information at the head of the data block in addition to the data length of the current data block. UDP server type applications need to use this mode.

4. IP Module

4.1. Overview

Internet Protocol (IP) is the network layer protocol used in Internet communications. It sits between the transport layer (usually TCP or UDP) and a data link layer (PPP). The protocol contains sub-protocol: Internet Control Message Protocol (ICMP).

The IP layer has three main functions:

1. Data Encapsulation – The IP layer encapsulates TCP and UDP segments sent to it by prepending an IP header in front of it.

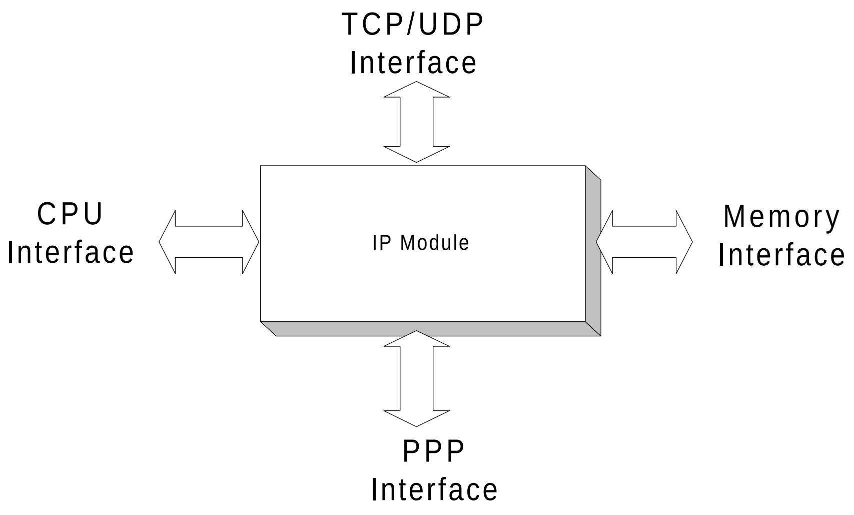
2. Transport Layer Decoding – Upon reception of an IP datagram, the IP layer examines the protocol field and spools the incoming datagram to the proper transport layer.

3. Control and Error Message Reporting – The IP protocol is responsible for reporting error conditions such as undefined ports, protocols, and IP addresses via the ICMP sub-protocol.

The features of S-7600A's IP implementation include the acceptance of broadcast messages, and ICMP echo and redirection codes.

The IP module interfaces with the TCP/UDP Module, Memory Interface, PPP Module, and the CPU as shown in Figure 4-1.

Figure 4-1 IP Module Block Diagram



4.1.1.

Definitions

FCS	Frame Check Sequence
IP	Internet Protocol
ICMP	Internet Control Message Protocol
LCP	Link Control Protocol
MRU	Maximum Receive Unit
NCP	Network Control Protocol
PPP	Point-to-Point Protocol
TCP	Transmission Control Protocol
TOS	Type of Service
UDP	User Datagram Protocol

4.1.2. Applicable Documents

Internet Control Message Protocol	J. Postel, Sept. 1981
Internet Protocol	J. Postel (ed.), Sept. 1981
Customer Identifier Generation	iReady Technical Specification

4.2. Functional Description

4.2.1. IP Frame Description

The default uncompressed IP frame structure is displayed in Figure 4-2.

Figure 4-2 Default IP Datagram Structure

Ver. No. 0x4	HD Len (4 Bits)	T.O.S. (8 Bits)	Total Length (16 Bits)	
Identification (16 Bits)			Flags (3 Bits)	Fragmentation Offset (13 Bits)
Time To Live (8 Bits)		Protocol (8 Bits)	Header Checksum (16 Bits)	
Source IP Address (32 Bits)				
Destination IP Address (32 Bits)				
Options (if any)				
Data Field (Variable, Length determined by Total Length Field)				

4.2.1.1. Version Number Field

The version number is currently fixed at 0x4. Therefore the IP that S-7600A will implement is also referred to as IPv4. This field in the header is fixed and is always sent as such. Upon reception, any value other than 0x4 in the header field will cause the datagram to be silently discarded.

4.2.1.2. Header Length Field

This 4-bit number indicates the total length of the IP header, including the Options field in number of 32 bit words (4 bytes). Since this is limited to a 4-bit number, the maximum length of an IP header is fixed at 15 * 4 bytes, or 60 bytes. If no options are present, the value of the header length is always 0x5 (20 bytes). This will be the value in all IP datagrams sent by the S-7600A implementation (since we do not support any options in the IP header).

4.2.1.3. Type of Service Field (TOS)

The Type of Service byte is made up of a 3-bit precedence field, 4 TOS bits, and an unused bit that must be 0. The byte is defined as shown in Table 4-1.

Table 4-1 TOS Bit Definition

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	0	C	R	T	D	Precedence[2:0]		

The Precedence bits are ignored today and will be set to 3'b000 when sent.

The TOS bits are defined as Minimum Cost (Bit 6), Maximum reliability (Bit 5), Maximum Throughput (Bit 4), and Minimum Delay (Bit 3). These bits are set on an application by application basis via the iAPI register set. The default value is 0x0.

Upon reception the S-7600A IP implementation will ignore all bits in the TOS byte.

4.2.1.4. Total Length Field

This field represents the total number of bytes in the IP datagram. Using this field in conjunction with the header length, the start of the data field of the IP datagram can be calculated. Since this field is limited to 16 bits, the largest IP datagram that can be sent is 64K.

4.2.1.5. Identification Field

This 16-bit field is used to uniquely identify each IP datagram that is sent. For the S-7600A implementation, this field will initialize to 0x0000 and increment by 1 with each datagram sent.

When receiving datagrams, this field is ignored by the S-7600A implementation. This is possible since we do not support fragmented IP packets.

4.2.1.6. Flag Field

These bits control IP datagram fragmentation and are defined in Table 4-2.

Table 4-2 Flag Field Bit Definition

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Definition	-	-	-	-	-	MF	DF	-

NOTE: Reserved bits are signified by a dash (-). All reserved bits set to 0.

Bit 2	MF More Fragments – this bit indicates whether more fragmented data is coming or whether this is the last datagram in the fragmented original datagram. 0 = last fragment 1 = more fragments
Bit 1	DF Do not Fragment - This bit indicates whether the current datagram can be fragmented or not. 0 = O.K. to fragment 1 = don’t fragment

The S-7600A IP implementation will always transmit both bits as 0. When receiving datagrams, any datagram received with the MF bit set will be silently discarded. The DF bit is ignored upon receipt.

4.2.1.7. Fragmentation Offset Field

This field is used to indicate the fragmentation offset of the current fragment from the beginning of the original datagram. It is used when the original IP datagram is split into multiple datagrams. Since the S-7600A implementation does not support fragmented IP datagrams, this filed is always ignored upon receipt and set to 0x0000 when transmitting.

4.2.1.8. Time to Live Field

This field sets an upper limit on the number of routers through which a datagram can pass, and therefore limits the lifetime of the datagram. It is set by the sender and decrements once each time it passes through a router. When this field reaches 0x00, the datagram is discarded, and the sender notified by an ICMP message.

The S-7600A implementation will fix this value at 0xFF (255 decimal) when sending datagrams out.

4.2.1.9. Protocol

This field indicates the type of data encapsulated in the current IP datagram. Table 4-3 displays the supported protocols.

Table 4-3 Supported Protocols

Code	Protocol
0x01	ICMP
0x06	TCP
0x11	UDP

All other IP datagrams received with other protocols will be silently discarded.

4.2.1.10. Header Checksum

This checksum is calculated over the IP header and includes only the options that are sent. The checksum is calculated by performing a 16-bit one's complement summation on the header with the checksum field initially set to 0x0000. The 16-bit one's complement of this sum is then used as the checksum in the outgoing datagram.

When receiving a datagram, a one's complement is performed on the header. The checksum is considered valid if the resulting sum is 0xffff. When an IP datagram is received with an invalid header checksum, the entire datagram is silently discarded.

4.2.1.11. Source IP Address

This 32-bit address represents the source of the IP datagram when one is received, and our IP address when a datagram is sent. Upon receiving a datagram, the Source IP Address is extracted from the header and reported to the TCP layer.

4.2.1.12. Destination IP Address

This 32-bit address represents the IP datagram's destination, and our IP address when a datagram is received. Upon receiving a datagram, if the destination IP address is not ours then the datagram is discarded. The exception to this is if an IP broadcast message is received (0xFFFFFFFF). These packets will always be accepted. If it is not expected, then the TCP layer will discard the packet. When sending a datagram, the destination IP address is furnished from the socket information.

4.2.1.13. Options Field

This field is used to transmit various parameters such as security and handling restrictions, record routing, and timestamp information. The options field must end on a 32-bit boundary and pad bytes are added in as necessary (pad bytes are always 0x00).

The S-7600A IP implementation does not use any option data when sending nor does it recognize any option data when received.

4.2.2. Internet Control Message Protocol (ICMP)

ICMP messages are used to report error conditions when processing datagrams. To avoid message snowballing, ICMP messages are never generated in response to ICMP messages. In addition to reporting error conditions, ICMP is also used to query other machines. This is the protocol used to implement the PING application.

ICMP messages are embedded within the data field portion of an IP. The structure is shown in Figure 4-3.

Figure 4-3 IP Datagram Structure

Message Type (8 Bits)	Sub-Code (8 Bits)	Checksum (16 Bits)
ICMP Data (dependent on Message Type and Sub-Code)		

4.2.2.1. Message Type

This field is used to indicate the general class of the ICMP message. The S-7600A implementation will only support Echo Requests (type 0x08), Echo Replies (type 0x00, send only), and Redirect Messages (type 0x05).

Note: Destination Unreachable messages are NOT supported because if they are received in a TCP segment, then a reset response is issued. If they are received in a UDP segment, then the datagram is silently discarded. Since UDP is a connectionless-unreliable protocol, this is acceptable.

4.2.2.2. Sub Codes

This field is used to indicate sub codes within each message type. The S-7600A implementation will only support sub codes 0x00 with the Echo Request (0x08) and Echo Reply (0x00) message types, and sub-codes 0x0, 0x01, 0x02, and 0x03 within the Redirect message (0x05).

4.2.3. Echo Request and Echo Reply Messages

A host uses an echo request or echo reply message in order to determine if another machine is reachable. The S-7600A IP will send an echo reply automatically in response to any echo request, but it will not support echo requests. Because of the latter, an echo reply should never be received. If an echo reply is received, it will be silently discarded.

Table 4-4 Echo Request/Reply Messages

Type	Sub Code	Description	Supported
0x00	0x00	Echo Reply (PING Reply)	Yes
0x03		Destination Unreachable	
	0x00	Network Unreachable	
	0x01	Host Unreachable	
	0x02	Protocol Unreachable	
	0x03	Port Unreachable	Yes
	0x04	Fragmentation Needed but DF Bit Set	
	0x05	Source Route Failed	
	0x06	Destination Network Down	
	0x07	Destination Host Unknown	
	0x08	Source Host Isolated (Obsolete)	
	0x09	Destination Network Administratively Prohibited	
	0x0a	Destination Host Administratively Prohibited	
	0x0b	Network Unreachable for TOS	
	0x0c	Host Unreachable for TOS	
	0x0d	Communication Administratively Prohibited by Filtering	
	0x0e	Host Precedence Violation	
	0x0f	Precedence Cutoff in effect	
0x04	0x00	Source Quench	
0x05		Redirect	
	0x00	Redirect for Network	
	0x01	Redirect for Host	Yes
	0x02	Redirect for Type-Of-Service and Network	
	0x03	Redirect for Type-of-Service and Host	Yes
0x08	0x00	Echo Request (PING Request)	Yes
0x09	0x00	Router Advertisement	
0x0a	0x00	Router Solicitation	
0x0b		Time Exceeded	
	0x00	Time to Live equals 0 during transit	
	0x01	Time to Live equals 0 during reassembly	
0x0c		Parameter Problem	
	0x00	IP Header Bad	

	0x01	Required Option Missing	
0x0d	0x00	Time Stamp Requested	
0x0e	0x00	Time Stamp Reply	
0x0f	0x00	Information Requested (Obsolete)	
0x10	0x00	Information Reply (Obsolete)	
0x11	0x00	Address Mask Requested	
0x12	0x00	Address Mask Reply	

4.2.3.1. Message Format

The format and definitions for an Echo Request or Reply is shown in Figure 4-4 and Table 4-5.

Figure 4-4 Echo Request/Reply Structure

Message Type (0x00 or 0x08)	Sub Code (0x00)	Checksum (16 Bits)
Identifier (16 Bits)		Sequence Number (16 Bits)
Optional Data (Variable Length)		

Table 4-5 Echo Request/Reply Definitions

Name	Definition
Message Type	The message type is 0x00 for an Echo Reply, and 0x08 for an Echo Request.
Sub Code	The Sub Code is always 0x00 for any Echo Request or Reply.
Checksum	The checksum is the one's complement of the one's complement sum of the entire Echo message including any data that is sent. If the data size is odd, then one additional padding byte (0x00) is appended to round the data field to 16 bits. For Echo Replies sent, the checksum is calculated by taking the checksum from the received echo request and adding 0x08 (echo request type) using one's complement addition. This is possible since the checksum represents the one's complement of the one's complement sum of the packet. Therefore to subtract 0x08, we normally should invert all bits before the operation, but since the result is a one' complement of the sum, we do not need to do this.
Identifier	This is normally the process ID of the sending process. When an Echo Request is received, the Identifier is copied into the Identifier field of the Echo Reply. When sending an Echo Reply. The Identifier will be fixed at 0x01.
Sequence Number	This field is usually used to keep track of Echo Requests sent. It is incremented with each Echo Request sent. When an echo Request message is received, the Sequence number is copied into the Sequence Number field of the Reply. The S-7600A implementation will always start the sequence number at 0x00 and increment it each time an ECHO Request is sent.
Optional Data	This field is used to send any data the sender wishes. Upon receiving an Echo Request, the Data is copied into the Data field of the Echo Reply message. For the S-7600A implementation.

4.2.4. Redirect Messages

These messages are used by a host to indicate to the sender that the datagram should have been sent to a different router. This message is supported for implementations that suit on broadband type physical connections such as Ethernet. In this case the Redirect Message is used in conjunction with the IP routing block. ICMP Redirect messages are only recognized when received, and will never be transmitted.

4.2.4.1. Message Format

The format for an ICMP redirect message is shown in Figure 4-5.

Figure 4-5 **ICMP Redirect Message Format**

Message Type (0x00 or 0x08)	Sub-Code (variable)	Checksum (16 Bits)
Gateway IP Address (32 Bits)		
IP Header + first 64 bits of Original Data (28 bytes)		

4.2.4.2. Message Type
The message type is 0x05 for Redirect Messages.

4.2.4.3. Sub Code
Valid sub codes are 0x0 to 0x3, inclusive. The S-7600A IP implementation will only accept Redirect messages with sub codes 0x1 (redirect for host) and 0x3 (redirect for TOS and host). Any Redirect message received with a sub code of 0x0 or 0x2 will be thrown away.

4.2.4.4. Checksum
The checksum is the 1's complement of the 1's complement sum of the entire redirect message.

4.2.4.5. Gateway IP Address
This field represents the address of the gateway to which future packets for the network specified in the IP destination field of the original datagram's data should be sent. This address is sent as a bus to the IP Routing block.

4.2.4.6. IP Header + 64 Bits of Data
This field represents the original IP header from the redirected IP datagram, plus the first 8 bytes of the data section of the packet. Address of the gateway to which future packets for the network specified.

5. PPP Module

5.1. Overview

Point-to-Point Protocol (PPP) is a data link layer protocol used to establish a connection between two peers. The PPP layer sits between the Internet Protocol (IP) layer and the Physical Transport layer. The protocol contains 2 sub protocols: Link Control Protocol (LCP) and Network Control Protocol (NCP). The NCP is also sometimes referred to as Internet Protocol Control Protocol (IPCP). The PPP serves three main purposes:

1. Data Encapsulation – The PPP layer encapsulates IP packets by placing a PPP header in front and appending it with a CRC block. The PPP block also inserts escape characters to distinguish data bytes from the end flag and other control characters.

2. Link Configuration – PPP configures the link between the client and a server. The options include compression, maximum receive units (MRUs), and magic number generation.

3. Network Configuration – The PPP layer also allows the transmission of multiple network protocols. Level 1 PPP performance will only support IP as a protocol. The NCP of the PPP allows for the negotiation of an IP address and for Van Jacobson Compression of TCP/IP headers.

The features of S-7600A's PPP implementation include:

- Auto-configuring timeout timer
- Asynchronous character control maps (ACCM)
- Protocol, address & control field compression
- Fixed or floating IP address support

The PPP Module interfaces with the IP/IP Router, Memory Interface, CPU Interface and the Modem interface as shown in Figure 5-1.

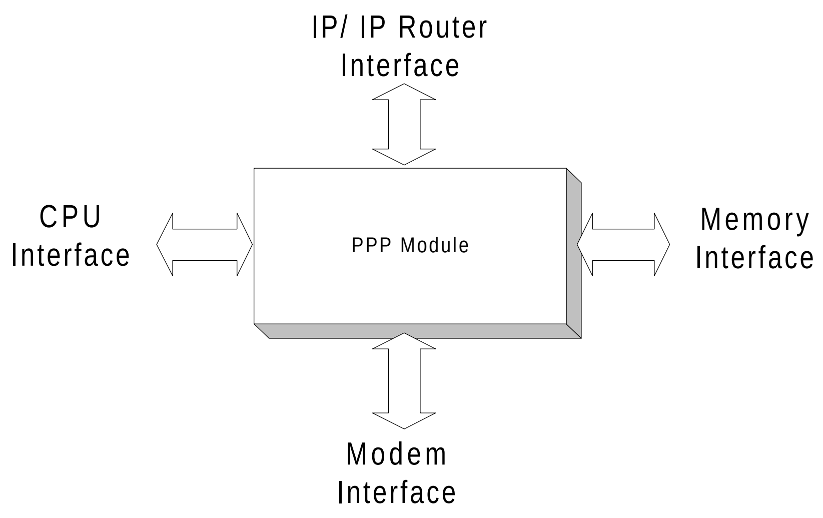


Figure 5-1 PPP Module Block Diagram

5.1.1. Definitions

ACCM	Asynchronous Character Control Map
FCS	Frame Check Sequence
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
LCP	Link Control Protocol
MRU	Maximum Receive Unit
NCP	Network Control Protocol
PAP	Password Authentication Protocol
PPP	Point-to-Point Protocol

5.1.2. Applicable Documents

RFC1332, The PPP Internet Protocol Control Protocol	G. McGregor, May 1992
RFC1334, PPP Authentication Protocols	B. Lloyd, W. Simpson, Oct. 1992
RFC1661, The Point-to-Point Protocol	W. Simpson, July 1994
RFC1662, PPP in HDLC-like Framing	W. Simpson, July 1994
RFC1700, Assigned Numbers	Reynolds & Postel, Oct. 1994

5.2. *Functional Description*

5.2.1. *PPP Frame Description*

The default, uncompressed PPP frame will have the structure shown in Figure 5-2.

Figure 5-2 Default PPP Frame Structure

Flag 0x7E	Address 0xFF	Control 0x03	Protocol (1 or 2 bytes)	PPP Frame Data (Up to 1498 Bytes)	FCS (2 Bytes)

The flag, address, and control fields have fixed values according to the above diagram. The protocol field is used to identify the data contained in the PPP frame, and it consists of one or two bytes depending on whether the protocol field compression is enabled. The protocol field has one of the values listed in Table 5-1.

Table 5-1 Protocol Field Values

Protocol Field (Uncompressed)	Protocol Field (Compressed)	Data Contained
0x0021	0x21	IP Datagram
0xC021	0xC021	Link Control Data
0xC023	0xC023	PAP
0x8021	0x8021	Network Control Data n (IPCP)

If the address and control field compression are enabled, the PPP frame will have the format shown in Figure 5-3.

Figure 5-3 PPP Frame Structure (w/address & control compression)

Flag 0x7E	Protocol (2 bytes)	PPP Frame Data (Up to 1498 Bytes)	FCS (2 Bytes)

In this document, all frames are transmitted from left to right (i.e. in Figure 5-2, the address field is sent before the control field).

5.2.2. PPP Protocol Sequence

The PPP connection is defined as being made up of five distinct phases. These are the phases in sequential order:

1. Initial Connection Phase
2. Link Control Phase
3. Network Control Phase
4. Data Transmission Phase
5. Termination Phase

Each of these phases has their own set of valid codes that are explained in the following sections. The Termination Phase can actually be entered at any time from the Link Control, Network Control, or Data Transmission Phases.

5.2.2.1. Initial Connection Phase

The PPP layer always starts in the Initial Connection Phase after a reset or after the Termination Phase. The PPP state machine will remain in this state until it is instructed by a higher layer to start a session (usually PPPEN from the iAPI register set) and the lower physical layer indicates that the connection is valid. When these two events occur, the PPP state-machine will transition into the Link Control Phase.

5.2.2.2. Link Control Phase

This phase is used to configure the communication link between two nodes, and is entered when a higher layer indicates to the PPP state machine to establish a session, and the lower physical transport layer indicates that the connection is valid.

In this phase, a set of communication and formatting options must be negotiated for each direction of data flow. This is accomplished by sending a series of Configuration Requests, Naks, Acks, and Reject commands. When a Configuration Acknowledge has been both sent and received, the PPP state machine will transition into the Network Control Phase.

5.2.2.3. Network Control Phase

This phase is used to negotiate the format of IP packets that are transferred between the two nodes, and is entered from the Link Control Phase after the transmission and reception of a Configuration-Acknowledge Packet.

Prior to completion of this phase, any IP protocol packet received will be silently discarded. All other protocol packets received will cause a Protocol-Reject response to be sent.

When a Configuration Acknowledge has been sent and received, the PPP state machine will enter the Data Transmission phase.

5.2.2.4. Data Transmission Phase

This phase is entered after an NCP Configuration Acknowledge packet has been both sent and received. At this point, the PPP state machine is ready to transmit and receive IP datagrams.

When transmitting data, the PPP state machine will take data from the IP layer and prepend the proper PPP header. The IP datagram is then inserted into the Data field within the PPP frame and the whole sequence passed through a FCS (Field Check Sum) generator. After the FCS is calculated, the stream is passed through an escape byte watcher. This block detects certain bytes and inserts a special escape byte sequence when one is detected. After all critical bytes are escaped, the data stream is passed to the Physical Transport (P.T.) Interface.

When data is received from the P.T. Interface, it is processed in the reverse order from the transmit process. First, the data is passed through the escape byte detector. In this block, all 0x7d bytes are removed, and the following byte is xor'ed with 0x20 before it is passed on. After the escape byte detector, the data stream is passed through the FCS checker. Any PPP Frame that fails the FCS checker is silently discarded. Frames that pass the checker are passed to a PPP frame de-multiplexer. This block will filter any LCP or NCP packets to the proper sub modules, and pass any IP packets up to the next layer.

The Data Transfer Phase remains active until the higher layer disables the PPP state machine (it wishes to disconnect from the host), a Termination Request packet is received from the host (the host wishes to disconnect the link), or the physical layer indicates that it has lost the current connection. After the Data Transfer Phase is complete, the PPP state machine transitions to the Termination Phase.

5.2.2.5. Termination Phase

This phase is entered when either the higher layer indicates that it wishes to end the session, or a Termination Request packet is received from the host, or the lower physical layer indicates that it has lost the current connection.

If this phase is entered due to a higher layer terminating the session (e.g. PPPEN de-asserting in the iAPI register set), the PPP state machine will send a Termination-Request Packet to the host. Termination-request packets will continue to be sent at intervals of 3 seconds until a Termination-Ack packet has been received. If 10 Termination-Request packets have been sent without a response, or if the lower physical transport layer indicates that it has lost the connection, then the link will be considered closed, and the PPP state machine will revert to the Initial Startup Phase. After a Termination Request packet is sent, all non LCP packets will be silently discarded. Once a Termination Ack packet is received in response to a Termination Request packet, the link will be immediately considered closed, and the PPP state machine will revert to the Initial Startup Phase.

If an unsolicited Termination-Ack is received, the link will also be considered closed, and some indication should be given to the upper layer indicating that an unexpected closing condition has occurred.

If this phase is entered in response to a Termination Request packet being received from the host, a Termination Ack will be sent, and the higher layer informed that the host has terminated the current session. After the Termination Ack is sent, the PPP state machine will revert back to the Initial Startup Phase.

If this phase is entered because the Physical Transport layer has lost its connection, the higher layer will be notified, and the PPP state machine will revert back to the Initial Startup Phase. No Termination Requests or Acks are sent under this condition.

5.2.3. Link Control Protocol (LCP) Phase

5.2.3.1. Link Control Phase Procedure

The LCP phase is used to configure the operating mode of the communication channel. This is accomplished via a set of Configuration commands. The first LCP command sent when the phase is first entered, is a Configuration-Request Packet. This packet will request that all the default settings for a PPP connection be used, with the exception of the ACCM option. The default and requested settings are reflected in Table 5-2.

Table 5-2 PPP Link Control Phase Default Option Settings

Option	Option Type	Default Value	Requested Value
Maximum Receive Unit (MRU)	0x01	1500 bytes	1500 bytes
Async. Control Character Map (ACCM)	0x02	0xffffffff	0x00000000
Authentication Protocol	0x03	Not Used	Not Used
Quality Protocol	0x04	Not Used	Not Used
Magic Number	0x05	Not Used	Not Used
Protocol Field Compression	0x07	Not Used	Not Used
Address & Control Field Compression	0x08	Not Used	Not Used

When a Configuration request packet is received, it is examined and based upon the options it contains different responses are sent. The responses to the different options are summarized in Table 5-3.

Table 5-3 PPP Link Control Protocol Responses

Option	Value Received	Response
MRU	Any Value	Accepted, Configuration Ack sent if all other options listed are acceptable also.
Authenti-cation Protocol	Any protocol other than PAP	Configuration Nak is sent and the PAP protocol requested.
	No protocol requested	Accepted, Configuration Ack sent if all other options listed are acceptable also.
Quality Protocol	Any protocol	Configuration Reject is sent and the Quality Protocol field received is returned (Not supported).
Magic Number	Any Number	Configuration Reject Sent and the magic number returned. (Not supported)
Protocol Field Compres-sion	Enable or Disabled	Accepted, Configuration Ack sent if all other options listed are acceptable also.
Address & Control Field Compres-sion	Enable or Disabled	Accepted, Configuration Ack sent if all other options listed are acceptable also.
Async. Control Character Map	Any Value	Accepted, Configuration Ack sent if all other options listed are acceptable also.

A Configuration Ack packet is sent only when the Configuration Request packet received contains only requests for a certain MRU size, and / or Protocol and Address & Control Field Compression, an ACCM, and PAP. Configuration Rejects will continue to be sent until this condition is met.

5.2.3.2. Restart Counter and Timeout Timer Settings

The LCP phase calls for the use of a Restart Counter and Timeout Timer to keep track of Configuration and Termination Requests, and for Configuration Naks. Each command uses a separate Restart Counter. Each counter decrements every time the command is sent, including the first time. The Time Out Timer is used to keep track of response times. Each time the Time Out Timer decrements to zero, a Time Out event is triggered.

5.2.3.2.1. Timeout Timer

The Timeout Timer is used to keep track of response times. Expiration of the timer usually causes a retransmission of the corresponding command.

In all cases, the timer will start with an initial time out period of 600 ms. Each successive time-out will increase the time out period by a factor of 2, up to a maximum of 4.8 seconds.

5.2.3.2.2. Restart Counter

The restart counter is used to keep track of how many times a Configuration Request, Nak, or Termination Request is sent.

In the case of Configuration Requests, the Restart counter keeps track of how many requests are sent before receiving an Ack, Nak, or Reject response. Anytime any of these commands are received, the Configuration Request Restart counter will reinitialize. The initial value of this counter is 10.

In the case of Configuration Naks, a count is kept of received Nak commands. Since the S-7600A PPP implementation will never Nak a Configuration Request a counter is not needed for sent commands. The receive Nak counter is kept to determine if the negotiations are not converging. It is possible for two PPP implementations to be set up such that they do not converge on a common set of configuration options. The Nak counter will decrement once each time a command is received. If a Configuration Ack is received, the Nak counter will reinitialize. The initial value for the counters is 5.

For Termination Requests, the Restart Counter keeps track of how many are sent without receiving a Termination Ack. If the Restart Counter reaches 0, then the link will be considered closed. The initial value for the Termination Ack Restart Counter is 5.

5.2.3.3. LCP Packet Format

The following is the format of LCP packets. It is embedded in the Data segment of a PPP frame. All PPP frames that contain LCP packets have their protocol field set to 0xC021.

Figure 5-4 LCP Packet Format Structure

Code (1 byte)	Identifier (1 byte)	Length (2 bytes)
Data ... (variable number of bytes)		

Valid Code field values are shown in Table 5-4.

Table 5-4 Valid LCP Codes

Code	Definition	Comment
0x01	Configuration-Request	Supported. Used to request a certain configuration
0x02	Configuration-Ack	Supported. Used to acknowledge a valid Configuration-Request.
0x03	Configuration-Nak	Recognized upon receipt, but never sent. Used to Not-acknowledge a Configuration-Request
0x04	Configuration-Reject	Supported. Used to reject a Configuration-Request
0x05	Termination-Request	Supported. Used to request a Termination of link
0x06	Termination-Ack	Supported. Used to acknowledge a Termination-Request
0x07	Code-Reject	Supported. Used to reject unknown codes within LCP packets.
0x08	Protocol Reject	Supported. Used to reject non-IP protocol packets.
0x09	Echo-Request	Recognized upon receipt, but not sent
0x0A	Echo-Reply	Supported. Used in conjunction with Echo-Requests.
0x0B	Discard-Request	Recognized upon receipt, but not sent.

Any codes received in an LCP packet not listed in the above table will cause the LCP state machine to send a Code-Reject response.

The Identifier field is used to track Configuration-Requests, Acks, Naks, and Rejects. Responses are matched to the corresponding Requests by matching the identifier number. This number will always start at 0x01 for the first request sent out, and will increment with each modified request sent. Requests sent as retries will not have their identifier field incremented.

The Length field is a two-byte value that indicates the length of the LCP Packet (including code, identifier, length and data fields).

The Data field is variable in length and its format is determined by the Code field.

LCP Codes

Note: Any LCP packet received with a code not listed in the following sections will result in a Code Reject Packet command being sent.

Configuration-Request (Code 0x01)

This is the first code sent during the Link Control Phase. The format for the Configuration-Request is shown in Figure 5-5.

Figure 5-5 LCP Configuration Request Format

Config. Req. Code (0x01)	Config. Req. ID (1 Byte)	Config. Request Length (2 Bytes, variable)
Configuration Request Options (Variable Length)		

The Identifier Byte starts at 0x01 and will increment with each modified Configure-Request Packet sent.

Requests that are sent as part of a TimeOut resend will not have their identifier field incremented.

The Length field represents the total length (in bytes) of the Configuration Request command, including the Code, Identifier, Length and Option fields.

The Options field is variable in length, and contains a list of configuration options that the sender wants to negotiate. The general format for the Options field is shown in Figure 5-6.

Figure 5-6 LCP Options Field Format

Option 1 Type (1 Byte)	Option 1 Length (1 Byte)	Option 1 Variables (2 Bytes, variable, depends on option)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (2 Bytes, variable, depends on option)
. .	. .	. .
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (2 Bytes, variable, depends on option)

Valid options are listed in Table 5-2, along with their option types. The Option Length field represents the total length of the option subsection, including the Option Type, Length, and Variable Fields.

The S-7600A PPP state machine will be designed to request all default PPP values with the exception of the ACCM option. The first Configuration Request frame will then have the format displayed in Figure 5-7.

Figure 5-7 First Configuration-request Frame Format

Config. Req. Code (0x01)	Config. Req. ID (0x01)	Config. Request Length (0x000e)
ACCM Type (0x02)	ACCM Length (0x06)	ACCM (0x0000)
ACCM (Continued) (0x0000)		

If this request is Nak’ed because of the ACCM, a Configuration Request will then be sent requesting the default ACCM (0xffffffff). This final request will have the format shown in Figure 5-8.

Figure 5-8 Final Configuration-request Frame Format

Config. Req. Code (0x01)	Config. Req. ID (0x02)	Config. Request Length (0x0004)

Note that since all default options are now being requested, the Options field is eliminated. A remote host may Nak the original Configuration Request because it wishes to negotiate an option not listed explicitly in it. If this occurs, the original Configuration Request will be sent again, but with the ID field incremented.

Magic numbers are used to detect looped back connections (i.e. a client is attempting to connect with a server application on the same platform). Since this will never occur in an embedded application, the

Magic Number option is not supported.

The 0x00000000 ACCM value is requested to cut down on the number of escaped bytes that are received thus providing more download.

A Configuration Request command received from a host will be acknowledged only when it contains the options shown in Table 5-5.

Table 5-5 Accepted LCP Configuration Options

Parameter	Value
Maximum Receive Unit (MRU)	Any Value
Authentication Protocol	PAP or none
Quality Protocol	Not Used
Magic Number	Not used
Protocol Field Compression	Any State
Address & Control Field Compression	Any State
Async Character Control Map (ACCM)	Any Value

If the Quality Protocols or Magic Number options are enabled, a configuration reject will automatically be sent. Any state for The Protocol Field Compression, Address & Control Field Compression or ACCM will be accepted.

5.2.3.3.1. Configuration-Ack (Code 0x02)

Configuration Acks are used to acknowledge Configuration requests if all options listed are acceptable. As stated earlier, since the S-7600A PPP implementation calls for the use of all default values, and since all PPP implementations MUST work with all default values, it is expected that we should always receive a Configuration Ack in response to our Configuration Requests.

An example of a Configuration-Ack, with an MRU, Protocol Field Compression and an ACCM option listed is shown in Figure 5-9.

Figure 5-9Example of a Configuration-ack Format

Conf. Ack Code (0x02)	Conf. Ack Identifier (Same as the ID byte of the Conf Req. Packet)	Configuration Ack Length (0x10)	
MRU Type (0x01)	MRU Length (0x04)	MRU Value (2 Bytes)	
Protocol Field Comp. Type (0x07)	Protocol Field Comp. Length (0x02)	ACCM Type (0x02)	ACCM Length (0x06)
ACCM (4 Bytes)			

The Identifier will be the same ID as the Configuration-Request Packet that it is responding to.

In the S-7600A PPP implementation, if the Identifier in the received Conf-Ack command does not match the identifier in the last Conf-Request command sent, then the entire packet is discarded.

5.2.3.3.2. Configuration-Nak (Code 0x03)

A Configuration Nak command is sent in response to a Configuration Request that contains an option that is recognized, but the value is unacceptable. The general format of the command is shown in Figure 5-10.

Figure 5-10 Configuration-nak Format

Configuration Nak Code (0x03)	Identifier (Same as the ID byte of the Conf Req. Packet)	Length (variable)
Option 1 Type (1 Byte)	Option 1 Length (1 Byte)	Option 1 Variables (variables)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (variables)
. .	. .	. .
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (variables)

The Code for a Configuration Nak command is 0x03. The Identifier byte is a copy of the Identifier byte from the Configuration request command that triggered the Configuration Nak command. The Length field represents the total length of the Configuration Nak packet, including the Code Field, Identifier Field, Length Filed, and Options Field.

A Configuration Nak must list only those options listed in the received Configuration Request that have unacceptable values. For options that are either enabled or disabled (e.g. Protocol Field Compression), the Configuration Reject command is used instead. Also, the Configuration Nak command must list options in the same order that they were received in.

The S-7600A PPP implementation will only send an LCP Configuration NAK in response to an Authentication Protocol request that lists anything other than the PAP protocol.

5.2.3.3.3. Configuration-Reject (Code 0x04)

A Configuration Reject is sent if a Configuration-Request Packet is received that contains any option types not understood by the receiver, or if it contains any recognized options that are not supported and have only a binary value (e.g. Quality Protocol option). It must send the options in the same order as they were received in the Configuration Request command. The general form of the Configuration Reject command is shown in Figure 5-11.

Figure 5-11 Configuration-reject Code Format

Configuration Rej. Code (0x04)	Identifier (Same as the ID byte of the Conf Req. Packet)		Length (variable)
Option 1 Type (1 Byte)	Option 1 Length (1 Byte)		Option 1 Variables (variables)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (variables)	
. .	. .	. .	
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (variables)	

The Code for a Configuration Rej. command is 0x04. The Identifier byte is a copy of the Identifier byte from the Configuration request command that triggered the Configuration Rej. command. The Length field represents the total length of the Configuration Rej. packet, including the Code Field, Identifier Field, Length Filed, and Options Field.

Since the S-7600A PPP implementation is requesting only default option values, it is not expected that a Configuration Reject will ever be received.

A Configuration Reject will be sent if the received Configuration Request contains the Quality Protocol or Magic Number Options. An example of this reject command is shown in Figure 5-12 responding to all three options.

Figure 5-12 Example of a Configuration-reject Format

Code (0x04)	Identifier (Same as the ID byte of the Conf Req. Packet)	Length (variable)
Qual. Prot Type (0x04)	Qual. Prot. Length (Same as received)	Quality Protocol Data (Same as received)
Magic Num. Type (0x05)	Magic Num. Length (0x06)	Magic Number (4 Bytes)

5.2.3.3.4. Terminate-Request (Code 0x05)

A Terminate Request is sent whenever a sender wishes to end the current PPP connection. In the S-7600A PPP implementation, this occurs when a higher layer (e.g. CPU or O.S. State Machine), wishes to end the current PPP connection. Normally this is accomplished by having PPPEN in the iAPI register set deasserted. The general format for the Terminate Request Packet is shown in Figure 5-13.

Figure 5-13 Terminate-request Packet Format

Term. Request Code (0x05)	Identifier (0x01)*	Length (variable)
Data (Optional) (variable)		

* increments whenever the Data field changes

The Code for Terminate Request commands is 0x05. The Identifier starts at 0x01 and will increment every time a Terminate Request Command is sent with the data field changed. The length field represents the total length of the Terminate Request command, including the Code, Identifier, Length, and Data fields.

The S-7600A PPP implementation will always send a Terminate Request with no Data field. As such, the length of the Terminate Request command sent will always be 0x04, and the Identifier will always be 0x01 (it never changes since the data field never changes).

Termination Request Packets will continue to be sent at the current Time Out intervals or until a Termination-Ack Packet is received. If 5 Termination-Request Packets are sent without an acknowledge, the link will be considered closed.

If a Termination-Request Packet is received, the PPP state machine will interrupt the higher layer, send a Terminate Ack command, and enter the Termination phase.

5.2.3.3.5. Terminate-Ack (Code 0x06)

Terminate-Ack commands must be sent in response to any Terminate-Request Packet received. The general format of the Terminate Ack command is shown in Figure 5-14.

Figure 5-14 Terminate-ack Packet Format

Term. Ack Code (0x06)	Term. Ack Identifier (Same as the ID of the Termination-Request Packet)	Terminate Ack Length (2 Bytes)
Data (Optional) (This will be a copy of the Data segment of the Terminate-Request Packet received)		

The Identifier field is a copy of the Terminate Request command received that triggered the Terminate-Ack. The Length field represents the total length of the Term. Ack packet, including the Code, Identifier, Length, and Data fields. The data field is a copy of the data field received in the Terminate Request command that triggered the Terminate-Ack.

Since the S-7600A PPP implementation will always send Terminate Requests with no data fields, it is expected that all Terminate Ack commands received will have a length of 0x04, and no data field.

If an unsolicited Terminate Ack command is received, the link will be considered down, and the PPP state machine returned to the LCP phase. The higher layer will be interrupted, and a Configuration Request will be sent to the host.

5.2.3.3.6. Code Reject (Code 0x07)

Code Reject commands are sent in response to any LCP packet that contains an unknown code. The format for this command is shown in Figure 5-15.

Figure 5-15 Code-reject Packet Format

Code Reject Code (0x07)	Code Rej. Identifier (0x01)*	Code Reject Length (2 Bytes)
Data (This will be a copy of the Data segment of the unknown code Packet)		

* increments with each Code reject sent

The Identifier field is incremented every time a code reject command is sent. The length filed represents the total length of the Code Reject command, including the Code, Identifier, Length, and Data fields. The data field contains a copy of the LCP packet which is being rejected. It begins with the Code field of the rejected code, but does not include any data link headers or FCS bytes.

In the case of the S-7600A PPP implementation, a Code reject will be sent in response to any LCP code received that is not listed in Table 5.1. Also, Code Rejects received fall into one of two categories; acceptable, and catastrophic. Acceptable code rejects are those received in response to Protocol Reject, Code Reject, Echo Requests, Echo Replies, and Discard Request commands. When a code reject is received in response to one of these commands, the PPP implementation will continue on as normal. Catastrophic code rejects are those received in response to any Configuration or Termination commands. If any of these are received, the higher layer is notified, and the link will be considered closed, since it is assumed that the host side is operating with a down revision PPP specification and correct operation is highly unlikely.

5.2.3.3.7. Protocol Reject (Code 0x08)

A Protocol Reject command is sent anytime a PPP frame is received with an unknown packet. The format for the code is shown in Figure 5-16.

Figure 5-16 Protocol-reject Packet Format

Protocol Rej. Code (0x08)	Protocol Reject ID (1 Byte)	Protocol Reject Length (2 Bytes, variable)
Rejected-Protocol (2 Bytes)		Rejected Information (variable)

The Protocol Reject ID field is incremented for every Protocol Reject packet sent. The Length field represents the total length of the Protocol Reject command, including the Code, ID, Length, Rejected Protocol, and Rejected Information fields. The Rejected Protocol field contains the PPP Protocol field of the frame which is being rejected. In the case that Protocol Field Compression is turned on, the Protocol is decompressed back to 2 bytes before being placed in the rejected protocol field. The Rejected Information field contains a copy of the information field from the PPP Frame that is being rejected, but does not include any Data Link Layer Headers or FCS fields. The Rejected Protocol command is truncated to 1500 bytes if necessary to comply with the receiver's MRU.

Protocol rejects are only sent after the LCP reaches the Opened State. Until then all, non-LCP packets are silently discarded.

In the case of the S-7600A PPP implementation, a Protocol reject is sent in response to any PPP frame containing a protocol other than 0x0021 (TCP/IP), 0x8021 (NCP), or 0xc021 (LCP). If a Protocol reject is received in response to any of these protocols, a Terminate Request will be sent, the connection terminated, and the higher layer interrupted.

5.2.3.3.8. Echo Request (Code 0x09)

An Echo request is used to test, debug, and determine the link quality of a connection. It can only be sent after the PPP automaton reaches the Opened State. Prior to this any Echo Request command received is silently discarded. The command has the format shown in Figure 5-17.

Figure 5-17 Echo-request Packet Format

Echo Request Code (0x09)	Echo Req. Identifier (1 Byte)	Echo Request Length (2 Bytes)
Magic Number (4 Bytes)		
Echo Request Data (variable)		

The Echo Request ID field is incremented every time the Echo Request Data field changes. The Length field represents the total length of the Echo Request command, including the Code, ID, Length, Magic Number, and Data fields. Since Magic Numbers are not needed in Xavier, this field is filled with 0x0000. The data field contains any information that the sender wishes to include. The end of the data field can be determined by the length value.

For the S-7600A PPP implementation, Echo Requests are never sent. They are recognized upon receipt however, and will always trigger an Echo Reply in response once the LCP has reached the open state.

5.2.3.3.9. Echo Reply (Code 0x0A)

Echo Replies are sent in response to Echo-Requests received after the LCP automaton reaches the Opened state. The format is shown in Figure 5-18.

Figure 5-18 Echo-reply Packet Format

Echo Reply Code (0x0A)	Echo Reply ID. (Same as the ID of the Echo-Request)	Echo Reply Length (2 Bytes)
Magic Number (4 Bytes)		
Echo reply Data (variable)		

The Echo Reply ID is a copy of the ID field of the Echo request that triggered the echo Reply. The Length field represents the total length of the Echo Reply command, including the Code, ID, Length, Magic Number, and Data fields. The data field is any arbitrary data the sender wishes to include.

For the S-7600A PPP implementation, the Magic number will always be 0x00000000, and the data will always be a copy of the data field from the triggering Echo Request command. Also, since no echo requests are ever sent, no Echo Replies are expected to be received. If one is received, it will be silently discarded.

5.2.3.3.10. Discard Request (Code 0x0B)

The Discard Request command is used for testing the local to remote data connection, and can only be sent in the LCP Opened State. The command has the format shown in Figure 5-19.

Figure 5-19 Discard-request Packet Format

Discard Req. Code (0x0B)	Discard Req. ID (1 Byte)	Discard Request Length (2 Bytes)
Magic Number (4 Bytes)		
Discard Request Data (variable)		

The ID field is incremented for each Discard Request sent. The Length field represents the total length of the Discard request command, including the Code, ID, Length, Magic Number, and data fields. The Discard Request Data Field is contains any data the sender wishes. The Length field can determine the end of the data field.

In the case of the S-7600A PPP implementation, the Discard Request command is never sent. It is recognized upon receipt, but is always silently discarded regardless of the state of the PPP automaton.

5.2.4. LCP Configuration Options

LCP Configuration Options are used to negotiate the communication channel between the local and remote host, and are included in the Configuration Request command. If a Configuration Option is NOT listed in the Configuration Request command, then the default value is assumed. Default values for Configuration Options should NOT be included in the Configuration Request command. The end of the list of Configuration Options can be determined by examining the Length field of the Configuration Request command. Configuration options apply in a half-duplex fashion from the point of view from the sender of the Configuration Request. A complete list of LCP Configuration Options, type numbers, default values, and S-7600A PPP accepted values are shown in Table 5-6.

Table 5-6 LCP Configuration Options

Option Type	Description	Default Value	Accepted Value
0x00	Reserved	-	-
0x01	MRU	1500	Any Value
0x02	ACCM	0xffffffff	Any Value
0x03	Authentication Protocol	Not Used	PAP
0x04	Quality Protocol	Not Used	Not Used
0x05	Magic Number	Not Used	Not Used
0x07	Protocol Field Compression	Not Used	Any Value
0x08	Address & Control Compression	Not Used	Any Value

Some notes on LCP Options Implementations:

- All options must be negotiated at he same time.
- Default values are not included in the Configuration Request
- Configuration Acks, Naks, and Rejects must not reorder any options.

- If the length field of an option is invalid, the whole option should be Nak'ed with a corrected length field.
- If the first option extends beyond the length of the Configuration Request field, the whole frame should be silently discarded.

5.2.4.1. Maximum Receive Unit (MRU)

This option is used to specify the maximum byte size of the information (data) field of a PPP frame (not including any header information). The peer is not required to send frames filled to this value, and may send smaller frames. A peer need not Nak an MRU option that requests smaller frames than 1500 since all implementations must accept 1500 byte frames. The largest MRU that can be requested is 256k bytes (0xffffffff). The format of the MRU option is shown in Figure 5-20.

Figure 5-20 MRU Option Format

MRU Type	MRU Length	MRU Value
(0x01)	(0x04)	(2 Bytes)

The Length of the MRU option will always be 0x04 (4 bytes). The MRU value represents the maximum receive unit in terms of number of bytes. Again the option is negotiated in a half duplex fashion from the point of view of the Configuration Request Sender. Therefore, if the S-7600A PPP implementation requests an MRU of 1500 bytes, that means it only wishes to receive 1500 byte maximum PPP frames.

The S-7600A PPP implementation will request the default value of 1500 bytes, and will accept what ever value the host wishes to use. This is possible since the PPP implementation need not fill each frame up to the maximum value.

5.2.4.2. Asynchronous Character Control Map (ACCM)

This option is used to dictate which Control characters (i.e. ASCII codes below 0x20) need to be escaped. When a character is escaped, it is preceded in the PPP frame by the escape byte 0x7d, and then xor'ed with 0x20. For example if the control character 0x14 (DC4) is escaped it will appear in the PPP frame as 0x7d, 0x34. The format of the ACCM option is shown in Figure 5-21.

Figure 5-21 ACCM Option Format

ACCM Type (0x02)	ACCM Length (0x06)	ACCM (2 Bytes)
ACCM (Continued) (2 Bytes)		

The ACCM length will always be 0x06 (6 bytes), and the ACCM value is always 4 bytes (32 bits). Each bit corresponds to a Control character, where bit0 is mapped to 0x00 (NULL), and bit31 is mapped to 0x1f (Unit Separator). If a bit position is set to "1", then the corresponding Control Character should be escaped. If the bit is set to "0", the Control Character can be passed unaffected. By default, the ACCM is set to 0xffffffff (all characters escaped).

The S-7600A PPP implementation will request that the ACCM be set to 0x00000000. This is meant to limit the number of escaped bytes received, therefore increasing the bandwidth available. If this option is Nak'ed, the default ACCM will be requested.

Regardless of the ACCM negotiated, the escape byte itself (0x7d), and the Flag byte (0x7e) are always escaped. Also the S-7600A PPP implementation will automatically unescape any byte that follows a 0x7d regardless of the ACCM negotiated.

5.2.4.3. Authentication Protocol

This option is used to authenticate a peer before allowing it to reach the NCP phase. The format of the option is as shown in Figure 5-22.

Figure 5-22 Authentication Protocol Option Format

Auth. Protocol Type (0x03)	Auth. Prot. Length (1 Byte, variable)	Authentication Protocol (2 Bytes)
Authentication Protocol Data (Optional) (variable)		

The Authentication Protocol Length is always 4 bytes or more. The Authentication Protocol field is 2 bytes and indicates the type of Authentication protocol desired. Currently assigned values are as shown in Table 5-7.

Table 5-7 Authentication Protocols

Value (hex)	Protocol
0xc023	Password Authentication Protocol (PAP)

The Authentication Protocol Data field is variable, and is determined by the actual protocol being used. By default the Authentication protocol is not used.

The S-7600A PPP implementation only supports PAP. Therefore, if the Authentication protocl requests anything other than PAP, a Configuration Nak will be sent in response. Also, the Authentication protocol is always requested from the host-server side. That is, the S-7600A generated Configuration Request command will never include the Authentication option. If the server-host does not include any Authentication protocol requests, this is also accepted.

5.2.4.4. Quality Protocol

This option is used to determine the quality of a link for diagnostic purposes. The format of the option is shown in Figure 5-23.

Figure 5-23 Quality Protocol Option Format

Qual. Protocol Type (0x04)	Qual. Prot. Length (1 Byte, variable)	Quality Protocol (2 Bytes)
Quality Protocol Data (Optional) (variable)		

The Quality Protocol Length is always 4 bytes or more. The Quality Protocol field is 2 bytes and indicates the type of Quality protocol desired. Currently, the only Quality protocol defined for PPP is *Link Quality Report* (0xc025). The Quality Protocol Data field is variable, and is determined by the actual protocol being used. By default the Quality protocol is not used.

The S-7600A PPP implementation does not support any Quality Protocols and will not request any. If any protocol is received in a Configuration Request, it will be rejected.

5.2.4.5. Magic Number

This option is used to as a method of detecting loop back links (i.e. a client to connecting to a server on the same node. It is also used in conjunction with some LCP commands such as echo Requests / Replies, and Discard Requests. The format of the option is as shown in Figure 5-24.

Figure 5-24 Magic Number Option Format

Magic Number Type (0x05)	Magic Num. Length (0x06)	Magic Number (2 Bytes)
Magic Number (Continued) (2 Byte)		

The Magic Number Length is always set to 0x06 (6 bytes), and the Magic Number itself is always a 4-byte number.

The S-7600A PPP implementation does not make use of Magic Numbers since loop back connections will never happen. Therefore, it is never negotiated for, and if one is received in a Configuration Request, it is rejected.

5.2.4.6. Protocol Field Compression

This option is used compress the Protocol field in the PPP header from 2 bytes down to 1 byte. This is accomplished by simply deleting the most significant byte if it is 0x00. This will be the case if the encapsulated data is IP (0x0021). The format of the option is as shown in Figure 5-25.

Figure 5-25 Protocol Field Compression Format

Protocol Comp. Type (0x07)	Protocol Comp. Length (0x02)

The Protocol Length field will always be 0x02 (2 bytes). Regardless of whether this option is negotiated, every PPP implementation must be able to receive 2 byte Protocol fields at any time. Also, protocols which have numbers greater than 256 cannot be compressed. Also, when compressed Protocol fields are being used, the FCS is calculated on the compressed value, and NOT the original uncompressed values.

The S-7600A PPP implementation will not negotiate for Protocol Compression on transmissions, but will accept it on receptions.

5.2.4.7. Address and Control Field Compression

This option is used compress the Address and Control Fields in the PPP header. This is accomplished by simply eliminating the fields. This is possible since in PPP implementations, the Address field is fixed to 0xff and the Control field is fixed to 0x03. The option format is as shown in Figure 5-26.

Figure 5-26 Address & Control Field Compression Format

Add & Control Comp. Type (0x08)	Add & Control Comp. Length (0x02)

The Address & Control Length field will always be 0x02 (2 bytes). Regardless of whether this option is negotiated, every PPP implementation must be able to receive uncompressed PPP headers at any time.

The S-7600A PPP implementation will NOT negotiate for Address & Control Compression on transmissions, but will accept it on receptions. This is because it is not worth the added gates and negotiate logic needed to save two byte on transmitted packets.

5.2.5. Network Control Protocol (NCP) Phase

5.2.5.1. Network Control Protocol Phase Procedure

The PPP state machine will enter the NCP phase after the LCP phase reaches the Opened State. And the Authentication phase (if negotiated) is complete. Any NCP packets received prior to this point are silently discarded. In the NCP phase, PPP will attempt to negotiate and configure the Network layer. In the case of the S-7600A PPP implementation, this layer is always IP. As such, NCP can also be referred to as IPCP (Internet Protocol Control Protocol).

As in the case of the LCP phase, negotiation is handled via a series of Configuration commands. Just as with LCP, a Configuration Ack must be both sent and received in order to complete this phase. The only option that needs to be configured in this phase is the IP addresses.

Upon successful negotiation of a set of NCP options, the PPP automaton will enter the Data Transfer phase.

5.2.5.2. NCP Packet Format

Figure 5-27 shows the format of NCP packets. It is embedded in the data segment of a PPP frame. All PPP frames, which contain NCP packets, will have their protocol field set to 0x8021.

Figure 5-27 NCP Packet Format

NCP Code (1 Byte)	Identifier (1 Byte)	Length (2 Bytes)
Options ... (Variable Number of Bytes)		

The NCP Code field is a one-byte value used to indicate the type of NCP Code being sent. Refer to section 4.3 for a list of valid NCP Codes. The Identifier field initializes to 0x01 and will increment every time the option field changes or a valid acknowledge is received in response to a command. The Identifier is NOT incremented for retransmissions. The Length field represents the total length of the NCP packet including the Code, Identifier, Length, and Option fields. The Options field is variable and is determined by the NCP Code being sent.

5.2.5.3. NCP Codes

5.2.5.3.1. Code Overview

Valid NCP codes are shown in Table 5-8.

Table 5-8 Valid NCP Codes

Code	Definition	Comment
0x01	Configuration-Request	Supported. Used to request a certain configuration
0x02	Configuration-Ack	Supported. Used to acknowledge a valid Configuration-Request.
0x03	Configuration-Nak	Supported. Used to Not-acknowledge a Configuration-Request
0x04	Configuration-Reject	Supported. Used to reject a Configuration-Request
0x05	Termination-Request	Supported. Used to request a Termination of link
0x06	Termination-Ack	Supported. Used to acknowledge a Termination-Request
0x07	Code-Reject	Supported. Used to reject unknown codes within NCP packets.

Any codes received in an NCP packet not listed in the above table will cause the NCP state machine to send a Code-Reject response.

5.2.5.3.2. Configuration-Request (Code 0x01)

This is the first code sent during the Network Control Phase. The general format for the Configuration-Request is as shown in Figure 5-28.

Figure 5-28 NCP Configuration-request Format

Config-Req. Code (0x01)	Identifier (1 Byte)	Config. Req. Length (2 Bytes, variable)
Option 1 Type (1 Byte)	Option 1 Length (1 Byte)	Option 1 Variables (variable, depends on option)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (variable, depends on option)
. .	. .	. .
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (variable, depends on option)

Valid NCP options, their types, default values, and S-7600A accepted and preferred values are shown in Table 5-9.

Table 5-9 Valid NCP Configuration Options

Type	Option	Default	Preferred	Accepted
0x01	IP Addresses *	-	-	-
0x02	IP-Compression -Protocol	-	-	-
0x03	IP-Address	No IP Address Assigned	Any Value	Any Value

* The IP-Addresses Option has been deprecated, and replaced with the IP-Address Option.

The S-7600A PPP implementation does not support the IP-Addresses Option (0x01) as this option has been deemed obsolete. This is due to the fact that it has been determined through implementation experience that it is difficult to ensure negotiation convergence using this option. The IP-Address Option (0x03) is used instead.

The S-7600A PPP implementation also does not support the use of Van Jacobson IP compression. As in the case of LCP Configuration Options, all NCP options must be negotiated for simultaneously. Therefore the first NCP Configuration Request will have the format shown in Figure 5-29.

Figure 5-29 First NCP Configuration-request Format

Config-Req. Code (0x01)	Identifier (1 Byte)	Config. Req. Length (0x000A)
IP Addresses Type (0x03)	IP Ad. Length (0x06)	IP Address (2 msbs)
IP Address (2 lsbs)		

In the case of floating IP addresses, it is expected that the first IP address request will be Nak'ed. After this is received, the IP Address recommended in the Configuration Nak command will be used in the next Configuration request command.

5.2.5.3.3. Configuration-Ack (Code 0x02)

Configuration Acks are used to acknowledge Configuration requests if all options listed are acceptable. Unlike the case of the LCP Configuration Request, the first NCP Configuration Ack is expected to be Nak'ed instead of Ack'ed due to the IP-Address option.

Under the S-7600A PPP implementation an NCP Configuration Ack will only be sent if the Configuration Request received contains no IP compression and it does not call out for the IP-Addresses Option (0x01) which is not supported. Any value contained in the IP-Address Option (0x03) will be accepted.

An example of a Configuration-Ack is shown in Figure 5-30.

Figure 5-30 Example of a NCP Configuration-ack Format

Conf. Ack Code (0x02)	Conf. Ack Identifier (Same as the ID byte of the Conf Req. Packet)	Configuration Ack Length (0x000A)
IP Add Type (0x03)	Length (0x06)	IP Address (0xWWXX)
IP Address (0xYYZZ)		

The Identifier will be the same Identifier as the Configuration-Request Packet that triggered the Configuration Ack.

When a Configuration Ack is both sent and received, the PPP automaton will transition into the Data Transfer phase.

5.2.5.3.4. Configuration-Nak (Code 0x03)

A Configuration Nak command is sent in response to a Configuration Request that contains an option that is recognized, but the value is unacceptable. The general format of the command is show in Figure 5-31.

Figure 5-31 NCP Configuration-nak Format

Configuration Nak Code (0x03)	Identifier (Same as the ID byte of the Conf Req. Packet)	Length (0x000a)
Option 1 Type (1 Byte)	Option 1 Length (1 Byte)	Option 1 Variables (variables)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (variables)
. .	. .	. .
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (variables)

The Code for a Configuration Nak command is 0x03. The Identifier byte is a copy of the Identifier byte from the Configuration request command that triggered the Configuration Nak command. The Length field represents the total length of the Configuration Nak packet, including the Code Field, Identifier Field, Length Filed, and Options Field.

A Configuration Nak must list only those options listed in the received Configuration Request that have unacceptable values, and must list the options in the same order that they were received in. It may also contain options wanted by the receiver but not sent in the configuration Request. In this latter case, the Conf Nak must contain all the accepted values for that particular option.

The S-7600A PPP implementation will only send a Configuration-Nak if an NCP configuration request packet is received without an IP address type option (0x03) included. The format for this Configuration Nak is shown in Figure 5-32.

Figure 5-32 NCP Configuration-nak w/Compression Format

Configuration Nak Code (0x03)	Identifier (Same as the ID byte of the Conf Req. Packet)	Length (0x000A)
IP Address Type (0x03)	Length (0x06)	IP Address Field (0x0000)
IP Address Field (0x0000)		

5.2.5.3.5. Configuration-Reject (Code 0x04)

A Configuration Reject is sent if a Configuration-Request Packet is received that contains any option types not understood by the receiver. It must send the options in the same order as they were received in the Configuration Request command. The general form of the Configuration Reject command is shown in Figure 5-33.

Figure 5-33 General Format for a Configuration-reject Command

Configuration Rej. Code (0x04)	Identifier (Same as the ID byte of the Conf Req. Packet)	Length (variable)
Option 1 Type (1 Byte)	Option 1 Length (1 Byte)	Option 1 Variables (variables)
Option 2 Type (1 Byte)	Option 2 Length (1 Byte)	Option 2 Variables (variables)
. .	. .	. .
Option “n” Type (1 Byte)	Option “n” Length (1 Byte)	Option “n” Variables (variables)

The code for a configuration-reject command is 0x04. The Identifier byte is a copy of the Identifier byte from the configuration-request command that triggered the configuration- reject command. The Length field represents the total length of the configuration-reject packet, including the Code Field, Identifier Field, Length Filed, and Options Field.

The S-7600A PPP implementation will issue an NCP Configuration Reject if the received Configuration request command contains an IP-Addresses Option (0x01), or any other Configuration Option not listed.

If a Configuration Reject is received in response to the IP-Address Option, the link will be terminated and the higher layer interrupted. This is because the host is probably operating under an obsolete PPP implementation and negotiation is probably not possible.

5.2.5.3.6. Terminate-Request (Code 0x05)

A Terminate Request is sent whenever a sender wishes to end the current Network connection. In the S-7600A PPP implementation, this command is never issued. Instead, when the appliance wishes to end a connection, an LCP Terminate request command is sent instead. When an NCP Terminate

Request is received, an NCP Terminate Ack is always sent. When this occurs, the NCP will be considered closed. All further data transfers are halted, the higher layer interrupted, and the PPP automaton will attempt to reestablish a network connection by sending an NCP Configuration Request. The general format for the Terminate Request Packet is shown in Figure 5-34.

Figure 5-34 NCP Terminate-request Format

Term. Request Code (0x05)	Identifier (0x01 *)	Length (variable)
Data (Optional) (variable)		

* increments whenever the Data field changes

The Code for Terminate Request commands is 0x05. The Identifier starts at 0x01 and will increment every time a Terminate Request Command is sent with the data field changed. The length field represents the total length of the Terminate Request command, including the Code, Identifier, Length, and Data fields.

5.2.5.3.7. Terminate-Ack (Code 0x06)

NCP Terminate-Ack commands must be sent in response to any NCP Terminate-Request Packet received. The general format of the Terminate Ack command is shown in Figure 5-35.

Figure 5-35 NCP Terminate-ack Format

Term. Ack Code (0x06)	Term. Ack Identifier (Same as the ID of the Termination-Request Packet)	Terminate Ack Length (2 Bytes)
Data (Optional) (This will be a copy of the Data segment of the Terminate-Request Packet received)		

The Identifier field is a copy of the Terminate Request command received that triggered the Terminate-Ack. The Length field represents the total length of the Term. Ack packet, including the Code, Identifier, Length, and Data fields. The data field is a copy of the data field received in the Terminate Request command that triggered the Terminate-Ack.

Since the S-7600A PPP implementation will never send a Terminate Request, no Terminate Ack packets are ever expected.

If an unsolicited Terminate Ack command is received, the TCP/IP will be considered down, and the PPP state machine returned to the NCP phase. The higher layer will be interrupted, and an NCP Configuration Request will be sent to the host in order to try and reestablish the link.

5.2.5.3.8. Code Reject (Code 0x07)

Code Reject commands are sent in response to any NCP packet that contains an unknown code. The format for this command is shown in Figure 5-36.

Figure 5-36 NCP Code-reject Format

Code Reject Code (0x06)	Code Rej. Identifier (0x01*)	Code Reject Length (2 Bytes)
Data (This will be a copy of the Data segment of the unknown code Packet)		

* increments with each Code reject sent

The Identifier field is incremented every time a code reject command is sent. The length filed represents the total length of the Code Reject command, including the Code, Identifier, Length, and Data fields. The data field contains a copy of the NCP packet which is being rejected. It begins with the Code field of the rejected code, but does not include any data link headers or FCS bytes.

For the S-7600A PPP implementation, any NCP Code Reject received fall into one of two categories; acceptable, and catastrophic. Acceptable code rejects are those received in response to a Code Reject command. When a code reject is received in response to this, the Code Reject is silently discarded, and the PPP implementation will continue on as normal. Catastrophic code rejects are those received in response to any Configuration or Termination commands. If any of these are received, the higher layer is notified, and the link will be considered closed, since it is assumed that the host side is operating with a down revision PPP specification and correct operation is highly unlikely.

5.2.5.4. NCP Configuration Options

NCP Configuration Options are used to negotiate network control parameters unique to each supported network. The only network that the S-7600A PPP implementation will support will be TCP/IP. As such the NCP support will be the same as IPCP (Internet Protocol Control Protocol). If a configuration option is not listed in the configuration request command, then the default value is assumed. Default values for configuration options should not be included in the configuration request commands. The end of the list of configuration options can be determined by examining the length field of the configuration request command. Configuration options apply in a half duplex fashion from the point of view of the sender of the configuration request.

5.2.5.4.1. IP Addresses

According to RFC 1332, the use of the Configuration Option IP-Addresses has been deprecated. It has been determined through implementation experience that it is difficult to ensure negotiation convergence in all cases using this option. The IP-Address Configuration Option (0x03) replaces this option, and its use is preferred.

The S-7600A implementation will reject this option when received and will never send this option.

5.2.5.4.2. IP Address

The S-7600A PPP implementation uses the IP Address option (0x03) to request an IP Address. This procedure will support both fixed and floating IP Addresses. Determination of the type of IP Address to use, is done via an input bit to the PPP automaton (normally from the iAPI register set). The format for this option is shown in Figure 5-37.

Figure 5-37 IP Address Option Format

IP Address Type (0x03)	IP Address Length (0x06)	IP Address (2 Most Significant Bytes)
IP Address (Continued) (2 Least Significant Bytes)		

If the PPP automaton is instructed to use a fixed IP Address, one must be provided. Again, this is normally done via the iAPI register set. In this case, the supplied address is inserted into the IP Address field of the IP Address option. If the IP Address is Nak'ed by the host, the higher layer will be interrupted, and the automaton will continue using the Nak'ed address instead.

If the PPP automaton is instructed to use a floating IP Address, then a bogus address of 00.00.00.00 will be inserted into the IP Address field of the IP Address option. This is a request for the host to furnish a valid IP address. The IP Address received in the Configuration Nak command sent by the host is then used in the next NCP Configuration request.

5.2.5.5. Escape Byte Insertion

A byte watcher will detect any IP data matching certain byte patterns and then insert the escape byte 0x7d. The critical bytes that are escaped when detected within the data field are summarized in Table 5-10.

Table 5-10 Critical Escape Bytes

Byte	Description	Escape Sequence
0x7e	Flag Byte	0x7d, 0x5e
0x7d	Control Escape Byte	0x7d, 0x5d
0x00 to 0x20	Control Characters	0x7d, (Control Byte XOR 0x20)
0xff	DEL Character	0x7d, 0xbf

The basic formula is that when a listed byte is detected, the escape byte 0x7d is inserted into the data stream. The listed byte is then xor'ed with 0x20 and sent out.

The exception to the above table is that the LCP ACCM option can be used to let certain Control Characters pass without being escaped.

Also, after an ACCM option has been negotiated, the DEL character (0xff) will no longer be escaped.

On the receiving end, the S-7600A PPP implementation will always remove any escape byte, and will always xor 0x20 with any byte that follows the escape byte regardless of the state of the ACCM. Escape byte insertion occurs after the FCS is calculated, and deleted prior to FCS checking.

5.3. PAP Support

The Password Authentication Protocol is used as a way of authenticating a user when logging onto a host system. The use of PAP is negotiated for at the LCP level. Once the LCP phase is complete, a PAP authentication Request packet is sent. The host-system will then either acknowledge the request or NAK it if it contains a bad username or password. If PAP is used, it must be acknowledged prior to entering the NCP negotiation phase. The protocol type for PAP is 0x023.

5.3.1. PAP Commands

PAP contains only 3 different commands as shown in Table 5-11.

Table 5-11 PAP Commands

PAP Command	Code Type
Authentication-Request	0x01
Authentication-ACK	0x02
Authentication NAK	0x03

5.3.2. Authentication-Request

This command is used to provide the authentication parameters from the client to the host, and has the format shown in Figure 5-38.

Figure 5-38 Authentication-request Format

Auth. Request Code (0x01)	Code Identifier (1 Byte)	Length (2 Bytes)
Peer-ID Length (1 Byte)	Peer-ID (variable length)	
Password Length (1 Byte)	Password (variable length)	

The Code Identifier byte will start at 0x01 and will increment by 1 each time an Authentication Request is sent out. The Length field indicates the total length of the packet including the Code-Type, Code-ID, Length, and data fields. The Peer-ID Length and Password Length fields indicate the length of the respective data fields. The Peer-ID indicates the name of the peer to be authenticated (same as a username). The Password field indicates the password to be used for authentication.

5.3.3. Authentication-ACK

This command is used to acknowledge an Authentication-Request. It is sent if both the Peer-ID and Password contained in the Authentication-Request are acceptable to the server-system. The command has the format shown in Figure 5-39.

Figure 5-39 Authentication-ack Format

Auth. ACK Code (0x02)	Code Identifier (1 Byte)	Length (2 Bytes)
Message Length (1 Byte)	Message (variable length)	

The Code Identifier must match that of the last Authentication request sent out. The length field indicates the length of the entire packet including the Code-Type, Code-ID, Length, Message Length and Message fields. The Message length is a one-byte field indicating the length of the message field. The message field is any arbitrary data that the host-system wishes to send.

The S-7600A PPP implementation will not store the received message. It will only keep track of the fact that an Authentication ACK packet was in fact received.

5.3.4. Authentication-NAK

This command is used to reject an Authentication-Request. It is sent if either the Peer-ID and Password contained in the Authentication-Request are not acceptable to the server-system. The command has the format shown in Figure 5-40.

Figure 5-40 Authentication-nak Format

Auth. ACK Code (0x02)	Code Identifier (1 Byte)	Length (2 Bytes)
Message Length (1 Byte)	Message (variable length)	

The Code Identifier must match that of the last Authentication request sent out. The length field indicates the length of the entire packet including the Code-Type, Code-ID, Length, Message Length and Message fields. The Message length is a one byte field indicating the length of the message field. The message field is any arbitrary data that the host-system wishes to send.

If an Authentication-NAK is ever received, the PPP automaton will resend the Authentication Request. It will retry for a specified number of times and then give up. If this situation occurs, and interrupt will be generated and reported to the higher layer.

Seiko Instruments Inc.
1-8, Nakase, Mihama-ku, Chiba-shi, Chiba 261, Japan
Components Sales Div.
Telephone : +81-43-211-1196 Facsimile : +81-43-211-8032
E-mail : component@sii.co.jp

Seiko Instruments USA Inc.
Electronic Components Div.
2990 W. Lomita Blvd, Torrance, CA 90505, USA
Telephone : +1-909-934-9334 Facsimile : +1-909-975-5699
E-mail : seiko-ecd@salessupport.com
hrrp://www.seiko-usa-ecd.com