Chapter 4

The Point to Point Protocol (PPP)

This chapter gives an overview of the PPP, explains the various stages, some packet formats, implementation details and options, and discusses the PPP implementation on the WMI™.

4.1 Introduction to PPP

Much help was acquired from Internet Request-For-Comments (RFC) documents on the details of the point-to-point protocol. PPP is a protocol that can be used to establish communication between any two communicating devices that need to exchange information. Information is exchanged in the form of structured data packets. The point-to-point links that utilize this protocol should be able to support full-duplex communication.

PPP can be fragmented into three parts:
1. Encapsulation
2. Link Control Protocol (LCP)
3. Network Control Protocol (NCP)

4.2 Encapsulation

Encapsulation is provided by PPP so that different protocols at the network layer can be supported simultaneously. Data is sent in frames whose general structure is shown in Figure 4.1 [2]. Data is transmitted from the left to right.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Information field</th>
<th>Padding field</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 - 16 bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.1: Encapsulation of PPP packets**

4.2.1 Protocol field

This field is one or two bytes and it identifies the data being sent in the information field. All protocol values are odd numbers. The least significant bit of the lower byte is always set to "1" and that of the most significant bit is always set to "0". Frames that violate these rules are treated as unrecognized protocols.

Some examples of protocol field values are:
- 0xC021 for Link control protocol
- 0xC023 for Password Authentication Protocol
- 0x8021 for Internet Protocol Control Protocol
4.2.2 Information field

This field is zero or more bytes long. It has a maximum length (including padding and excluding the protocol field) of 1500 bytes. This limit is termed as the Maximum Receive Unit (MRU) at the receiving end and Maximum Transmit Unit (MTU) at the transmitting end. The default field length is also 1500 bytes. Negotiations are possible between peers as to the MRU values.

4.2.3 Padding

This is an optional field. The information field may be padded with as many bytes as needed to reach the MRU. However, both peers should be able to recognize the padding bytes from the true data.

4.3 Block diagram

To establish communication between two peers, LCP packets must first be sent both ways to configure the link. Any peer may request validation after link configuration. This phase is optional when there is no such request made. Figure 4.2 shows the phase diagram that the link passes through in order to support PPP [2].

The end of the link establishment phase (or the authentication phase as the case may be) triggers the next phase - the network phase. In this stage a Network Control Protocol (NCP) is first selected and then the link can proceed to obey the rules of sending/receiving NCP packets. Some of the available Network Layer Protocols (NLP) are ATCP (AppleTalk Control Protocol), IPCP (Internet Protocol Control Protocol), Novell IPX Control Protocol, etc. These protocols are similar to the LCP in message format, with varying details. It is at this layer, that messages can be sent to select IP addresses.

The link remains open for communications until special LCP/NCP packets are sent to close down the link or other events trigger a shutdown (time out, human intervention, etc.).

4.4 PPP phases

4.4.1 Link Dead phase

The link starts and stops in this phase. The detection of a carrier signal at the peer triggers the link to proceed to the next phase. Disconnecting from the modem line should bring the link back to this phase.

4.4.2 Link Establishment phase

Once the presence of the peer is detected, the link proceeds to this phase. In this phase, the LCP establishes a healthy connection by exchanging configuration packets. After the link configuration has been consented upon, Configure-Ack packets are sent and received [2].
Configuration options have defined default values, which can be modified during this phase. These options are independent of the network layer protocol being implemented. These options are negotiated between the peers based on the hardware and software abilities at both the ends.

Any non-LCP packets received during this phase should be discarded and logged. When the link is in the network layer protocol (NLP) phase, receiving a Configure-Request packet causes the link to move back to the Link Establishment phase [2].

The end of this phase indicates the LCP open state.

### 4.4.3 Authentication phase

This is an optional phase. Before proceeding to the NLP, a peer may request authentication or validation by the other peer. By default, this phase is optional. If a peer requests authentication, a request must be issued during the link establishment process where configuration options are negotiated. If requested, this phase must be entered as soon as link establishment is complete. It is possible that link quality determination may occur during this phase. If link quality determination needs to be performed while in the authentication phase, appropriate priority levels should be given to the quality determination process.

Entering the NLP requires passing the authentication phase. Failing at validation necessitates the link to move to the termination phase, only after a sufficient number of failed attempts. Only authentication protocol, LCP, and link quality determination packets can be sent and received during this phase. All other packets received must be discarded and logged.

There are two types of authentication protocols that can be implemented.
4.4.3.1 Password Authentication Protocol (PAP)

The PAP provides an easy implementation of peer authentication. It is only performed after the link establishment phase. The peer repeatedly requests an ID/Password pair until authentication is acknowledged. If invalid authentication is received after repeated multiple requests, the link is terminated.

This protocol is not the most secure implementation since passwords are sent without any encryption over the links. There is no protection from repeated trial attacks to hack the password.

4.4.3.1.1 Configuration option

Figure 4.3 shows the format for the authentication protocol configuration option to request PAP validation by the peer [2]. In this format,

- **Type** = 3
- **Length** = 4
- **Authentication-Protocol** = 0xC023 for PAP

<table>
<thead>
<tr>
<th>Type field</th>
<th>Length field</th>
<th>Authentication Protocol</th>
</tr>
</thead>
</table>

**Figure 4.3: Configuration format for PAP**

4.4.3.1.2 Packet format

The generic PAP packet is shown in Figure 4.4 [2]. Each PPP frame houses only one PAP packet in its information field. The protocol field is set to 0xC023, reserved for the PAP.

The **code** field is 1-byte wide and is either 1, 2 or 3 depending on whether an Authentication-Request, Authentication-Ack, or Authentication-Nak is being sent or received.

<table>
<thead>
<tr>
<th>Code field</th>
<th>Identifier field</th>
<th>Length field</th>
<th>Data field</th>
</tr>
</thead>
</table>

**Figure 4.4: Configuration format for PAP,LCP**

The **identifier** field (1-byte wide) is used to match requests and replies. The **length** field (2-bytes wide) holds the length of the PAP packet, which includes all the fields transmitted (**Code**, **length**, **identifier**, and **data** fields).

The **data** field is zero or more bytes wide and its content depends on the **code** field contents (Request, Acknowledge or Not-Acknowledge).
For more details on the different command structures for Authentication-Request, Authentication-Ack, or Authentication-Nak, please refer to [2].

4.4.3.2 Challenge-Handshake Authentication Protocol (CHAP)
Unlike the PAP, where authentication is requested only at the initial time of link establishment, the CHAP necessitates periodic peer validation. This is done at initial link establishment, and could also be requested after link establishment. The authenticator sends a challenge signal to the peer who responds with a value computed from a complex algorithm. This returned value is compared at the authenticator's end with its expected value. If the values match, the peer is validated or else the link is terminated after a specified number of failed attempts. This ensures greater security in the implementation. If this protocol is implemented, the protocol field value in the PPP frames has a value of 0xC223. More information on this protocol can be found in reference [2].

4.4.4 Network Layer Protocol phase
Once the PPP has successfully passed through the authentication phase, the NLP phases must be configured (similar to the LCP phases). Some examples of NLPs are Internet Protocol (IP), AppleTalk (AT) etc. The configuration of these NLPs is achieved by implementing the appropriate NCPs. The corresponding NCPs are Internet Protocol Control Protocol (IPCP) for IP, AppleTalk Control Protocol (ATCP) for AT etc.

Each NCP can be opened and closed independently at any time. The RFC rules strongly recommend the avoidance of fixed timeouts while waiting for NCPs to configure. This is due to the significant latency involved in piercing through the link establishment phase (including quality determination and possible authentication). Any supported NLP packets received when the corresponding NCP is closed are discarded after logging. Similarly, any unsupported NLP packet must, in the LCP open state, be returned with a Protocol-Reject packet.

4.4.5 Link Termination phase
The link can be terminated at any point of time. This can happen due to any of the following factors -- carrier can not be detected, authentication failure, idle-period time-out, human intervention or bad link quality.

The link is shut down after sending and receiving Terminate packets. Before shutting down, PPP informs the upper NLPs so that appropriate action (termination) is taken at all layers.

After Terminate packets are exchanged, the implementation closes the physical-link thus terminating the link. Upon sending a Terminate-Request, the requester waits for a Terminate-Ack or waits for a timer to expire before actual termination. Likewise, the receiver of a Terminate-Request packet waits for the peer to disconnect or waits for at least one time-out period after issuing a Terminate-Ack packet before disconnecting.

Any non-LCP packets received after the link is terminated are logged and discarded. Link closure at the LCP level is sufficient for termination. It is not
required that there be termination at each NCP level. Contrarily, each NCP closure is not sufficient reason for link termination. The link has now reached the link dead phase again.

4.5 Option negotiation for LCP

In order to reach the open state and enter the NLP phase of the block diagram, a well-defined finite state automaton is described by the RFC [2]. This is briefly discussed in this section. The different attributes of this automaton are events, actions and state transitions, as with most finite state machines. Some definitions are given below for more clarity.

An **event** is any external occurrence or command -- for e.g., receiving packets from the peer, link **open** and link **close** commands, time-out of the restart timer etc. An even triggers the state machine to change state or remain in its current state.

An **action** is the outcome of the event. Examples of actions include re-initializing the restart timer, sending packets to the peer, triggering link status flags etc. Every event need not cause an action to be performed.

A **state transition** is the reaction of the state machine to an event. A state transition may keep the machine in its current state. For example, in the **Closed** state, receiving a link **close** event makes the state machine to remain in the **Closed** state without performing any actions.

4.5.1 State Transition table

Figure 4.5 [2] shows the complete state transition table that must be used during implementation. This is the core of the LCP implementation. The horizontal title bar reads the **states** and the vertical bar reads the **events**. Entries with a '-' indicate illegal state transitions. State transitions and actions are represented as action/next-state. Commas separate multiple actions.

The different **states** are - **initial**, **starting**, **closed**, **stopped**, **closing**, **stopping**, **request-sent**, **ack-received**, **ack-sent**, and **open**.

The different **events** are - **up**, **down**, **open**, **close**, **TO+**, **TO-**, **RCR+**, **RCR-**, **RCA**, **RCN**, **RTR**, **RTA**, **RUC**, **RXJ+**, **RXJ-**, **RXR**.

The different **actions** are - this-layer-up (**tlu**), this-layer-down (**tld**), this-layer-started (**tls**), this-layer-finished (**tlf**), initialize-restart-counter (**irc**), zero restart counter (**zrc**), send-configure-request (**scr**), send-configure-acknowledge (**sca**), send-configure-Not-acknowledge (**scn**), send-terminate-request (**str**), send-terminate-acknowledge (**sta**), send-code-reject (**scj**), and send-echo-reply (**ser**).

Details on definitions of state transitions, actions and events are available in the [2] along with implementation suggestions.
<table>
<thead>
<tr>
<th>States →</th>
<th>Initial</th>
<th>Starting</th>
<th>Closed</th>
<th>Stopped</th>
<th>Closing</th>
<th>Stopping</th>
<th>Req-sent</th>
<th>Ack-Rx'd</th>
<th>Ack-sent</th>
<th>Opened</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Up</strong></td>
<td>2</td>
<td>irc,scr/6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Down</strong></td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>tls/1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>tld/1</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>tls/1</td>
<td>1</td>
<td>irc,scr/6</td>
<td>3[r]</td>
<td>5[r]</td>
<td>5[r]</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9[r]</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>0</td>
<td>tlf/0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>irc,str/4</td>
<td>irc,str/4</td>
<td>irc,str/4</td>
<td>tlf,irc,str/4</td>
</tr>
<tr>
<td><strong>Timeout w/ctr &gt; 0</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>str/4</td>
<td>str/5</td>
<td>scr/6</td>
<td>scr/6</td>
<td>scr/8</td>
<td>-</td>
</tr>
<tr>
<td><strong>Rx'd valid Cfg. Req.</strong></td>
<td>-</td>
<td>-</td>
<td>sta/2</td>
<td>irc,scr, sca/8</td>
<td>4</td>
<td>5</td>
<td>Sca /8</td>
<td>Sca /8</td>
<td>Sca /8</td>
<td>tld,scr, sca/8</td>
</tr>
<tr>
<td><strong>Rx'd invalid Cfg. Req.</strong></td>
<td>-</td>
<td>-</td>
<td>sta/2</td>
<td>irc,scr, scl/6</td>
<td>4</td>
<td>5</td>
<td>Sen /9</td>
<td>Sen /6</td>
<td>Sen /6</td>
<td>tld,scr, scl/6</td>
</tr>
<tr>
<td><strong>Rx'd Cfg. Ack.</strong></td>
<td>-</td>
<td>-</td>
<td>sta/2</td>
<td>sta/3</td>
<td>4</td>
<td>5</td>
<td>irc/7</td>
<td>irc,tl u /9</td>
<td>irc, tl/9/9</td>
<td>tld,scr/6 [x]</td>
</tr>
<tr>
<td><strong>Rx'd Cfg. Reject</strong></td>
<td>-</td>
<td>-</td>
<td>sta/2</td>
<td>sta/3</td>
<td>4</td>
<td>5</td>
<td>irc, scl/6</td>
<td>irc, scl/8</td>
<td>irc, scl/8</td>
<td>tld,scr/6 [x]</td>
</tr>
<tr>
<td><strong>Rx'd Terminate Req.</strong></td>
<td>-</td>
<td>-</td>
<td>sta/2</td>
<td>sta/3</td>
<td>sta/4</td>
<td>sta/5</td>
<td>sta/6</td>
<td>sta/6</td>
<td>sta/6</td>
<td>tld,zrc, sta/5</td>
</tr>
<tr>
<td><strong>Rx'd Terminate Ack.</strong></td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>tlf/2</td>
<td>tlf/3</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>tld,scr/6</td>
</tr>
<tr>
<td><strong>Rx'd Unknown Code</strong></td>
<td>-</td>
<td>-</td>
<td>scj/2</td>
<td>scj/3</td>
<td>scj/4</td>
<td>scj/5</td>
<td>scj/6</td>
<td>scj/8</td>
<td>scj/8</td>
<td>scj/9</td>
</tr>
<tr>
<td><strong>Rx'd Echo Req/Reply</strong></td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>ser/9</td>
</tr>
</tbody>
</table>

**Figure 4.5: State Transition Table**

### 4.6 LCP packets

LCP packets can be categorized into configuration packets, termination packets and maintenance packets. Only one LCP packet is bundled into the PPP information field. The protocol field in the PPP frame reads 0xC021 (for LCP). The generic format of an LCP packet is shown in Figure 4.4 [2]. It is similar to that of the PAP packets.

The **code** field indicates the type of LCP packet. For example, **Configure-Request** packet has a code field of 1, **Configure-Ack** has a code field of 2, etc.

The **identifier** as in the case of the PAP packet matches the requests and replies.

The **length** field holds the length of the LCP packet including the code, identifier, length and data fields. This length is limited by the MRU negotiated.
The **data** field is zero or more bytes and is determined by the **code** field. For e.g., when sending a **Configure-Ack** to the peer, the data field holds the configuration options that can be negotiated.

For a more in-depth study on individual packet structures for various actions of the state machine (**Configure-Ack**, **Configure-Request**, etc.) the RFC [2] can be referred to.

### 4.7 LCP Configuration Options

Negotiating these options modify the attributes of the PPP link. If the **Configure-Request** packet does not contain any configuration option information, the default values are assumed.

The options that can be negotiated are MRU (Maximum Receive Unit), Authentication Protocol (discussed in Section 4.4.3), Quality Protocol, Protocol Field Compression, and Address and Control Field Compression [2].

#### 4.7.1 Maximum Receive Unit

This is the maximum number of bytes that the peer can handle in an LCP packet. The default value is 1500 bytes. A higher value does not necessitate the peer to send MRU length packets.

#### 4.7.2 Authentication Protocol

By default, authentication is not needed. As mentioned before the Password Authentication Protocol and the Challenge-Handshake Authentication Protocol are the available two protocols in case either peer requests authentication.

#### 4.7.3 Quality Protocol

It may be required by some implementations that the link quality be monitored. If such a request is made, the implementation receiving such a request should send a report on the data statistics. The protocol to be followed is the Link Quality Report protocol. By default this option is disabled.

#### 4.7.4 Protocol Field Compression

On low speed links this compression can be used for eliminating redundant data. This option is used for low bandwidth applications at the cost of increased implementation complexity. Protocol fields for LCP packets are never compressed thus making them easily identifiable. If used, the FCS is computed on the compressed frame and not on the original.

#### 4.7.5 Address-and-Control-Field Compression

Since these fields are invariant once the link is established, they can be compressed as well. As with Protocol Field Compression, LCP packets should not be compressed in these fields in order to have them recognizable.
4.8 HDLC Framing

Now that the theoretical part of the link control protocol has been discussed, it is time to discuss the framing of PPP packets. PPP uses HDLC (High-level Data Link Control) framing format for sending its data. HDLC is a set of protocols for transmitting data between two network nodes. According to these protocols, data is sent in units of frames with defined header bytes, escape sequences etc [2].

4.8.1 Frame structure

The PPP HDLC frame is shown in Figure 4.6 [2]. All fields are transmitted from left to right. All frames begin and end with a Flag Sequence, which is hexadecimal 0x7E. This is used as a sync byte. Two flag sequences in a row indicate an empty frame that is rejected and logged.

The Address Field is one byte wide and has the value 0xFF, the address for All-Stations. Unrecognizable addresses should be rejected and logged.

The Control Field is defined as hexadecimal 0x03. Unrecognizable control field frames should be logged and rejected.

<table>
<thead>
<tr>
<th>Flag (0x7e)</th>
<th>Address (0xFF)</th>
<th>Control (0x03)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol 1/2 bytes</td>
<td>Information</td>
<td>Padding</td>
</tr>
<tr>
<td>FCS 2/4 bytes</td>
<td>Flag (0x7e)</td>
<td>Next address or Inter-frame fill</td>
</tr>
</tbody>
</table>

Figure 4.6: PPP datagrams in HDLC framing

The Protocol, Information, and Padding fields are as discussed in the LCP packet structure. The Frame Check Sequence (FCS) is computed over the entire packet - Address, Control, Protocol, Information, Padding, and excluding the FCS bytes and the Flag Sequences.

Received bytes that are marked in the Async-Control-Character-Map (ACCM) are neglected before FCS calculation. The end of the Information and padding fields is found by backing off two bytes (for FCS) from the closing 0x7E byte.
4.8.2 Modified frame structure

The LCP can have modified frame structures after negotiations using Protocol Field Compression (PFC) and Address-and-Control-Field-Compression (ACFC) techniques already described. For e.g., using ACFC the default values of 0xFF and 0x03 for the address and control fields, respectively, can be left out. Upon reception, if the first two received bytes are 0xFF and 0x03 it is assumed that no ACFC is used. Else, it is assumed that ACFC is used and that these bytes were not transmitted. Simultaneously, 0xFF can never be the first byte of a protocol field (because the first byte of a protocol field should always be even and the second byte should always be odd) and the value 0x00FF is reserved when PFC is being used with the first information byte being 0x03.

The control escape sequence 0x7D is used to precede special bytes sent on the link. The byte being escaped is XOR'ed with 0x20. For e.g., 0x7D is sent out as the byte set - 0x7D 0x5D and 0x11 (XON) is transmitted as 0x7D 0x31. Invalid frames (shorter than 4 bytes, flag sequence followed by flag sequence, etc.) are logged and discarded without being treated as FCS errors.

4.8.3 PPP frame detection

Some examples of PPP frame beginning patterns are (all hexadecimal values) [2]:
7E FF 03 C0 21…
7E FF 7D 23 C0 21…
7E 7D DF 7D 23 C0 21…

The first example shows the simplest format of PPP frames without any compression’s or escape sequences. The second example shows only the control field byte escaped with an XOR (0x03 ^ 0x20 = 23). The final example shows both the control and address field bytes escaped with 0x7D.

4.8.4 FCS computation

The RFC 1662 contains FCS generation code that was borrowed and recreated during the LCP implementation on the WMI. The FCS algorithm prevents against erroneous frame transmissions or invalid frames. For more information on the FCS the [2] is recommended.

4.9 Network layer protocols

All of the above discussion deals in detail with the LCP and the framing of PPP packets [2]. The actual data carrying protocols are the Network Layer Protocols (NLP’s) that have similar LCP-type packet structure. The Network Layer transactions are performed once the LCP has successfully reached the Open State. For more details on the AppleTalk, Internet Protocol or other Network Layer Protocol details, help could be found in the RFC’s for the appropriate protocols.
4.10 Implementation on the WMI

Implementation was commenced on the WMI by first starting with the PC setup shown in Figure 4.7. With this set up, using one of the COM ports, code was written to communicate to the modem and dial the Internet Service Provider (physical layer in the Open Systems Interconnect model). This application software was successful in dialing into the modem pool, establishing a connection and receiving a "login/password" prompt from the ISP. Code was also written to send out a valid login/password response. Having sent this information, I observed that the peer reciprocated by sending out LCP packets (refer to copy of transaction in Figure 4.9) in the HDLC [2] format. This code was then transported to the WMI set up (Figure 4.8) with appropriate modifications. Code modifications were needed because the serial ports on the WMI were not similar to those on the PC as mentioned in the previous chapter. The implementation on the PC set up was relatively simpler because the COM ports on the PC are ideally set up (i.e., with hardware flow control availability) for communicating with the transmitting modem. Due to the nature of the hardware design on the WMI for the serial ports, software flow control was used with XON/XOFF headers.

XON/XOFF are two defined control characters in the ASCII character set. When the transmitter is sending out data, the XON character is appended to the character stream. As soon as data transmission is done the transmission of an XOFF character implies to the receiver that transmission is complete. The receiver would respond similarly if it needed to talk back to the transmitter.

![Figure 4.7: PC setup for modem](image)

![Figure 4.8: WMI setup for modem communications](image)
When the message data was transported to the WMI setup I observed that a connection was established between the ISP and the WMI. However, the WMI did not receive the "login/password" prompt. Upon consulting the protocol documentation I interpreted that I may need to implement the Password Authentication Protocol. Implementing a skeletal version of the PAP did not result in any login/password prompts from the peer either. I suspected that the peer might be waiting for LCP Configure-Request packets and worked on implementing this portion of the Link Control Protocol. This attempt proved to be futile as well.

```
7E 7D 23 C0 21 7D 21 7D 21 7D FF 7D 34 7D 22 7D 26 7D 7F 7D 2A 7D 7F 7D 7F
7D 7F 7D25 7D 26 7D 39 7C 6D 7D 27 7D 22 7D 28 7D 22 5E 2C
```

**Sample message received from ISP**

```
7E 03 C0 21 01 01 DF 14 02 06 5F 0A 5F 5F 05 06 19 7C 6D 07 02 08 02 5E 2C
```

**Sample message being sent**

**Figure 4.9: Interpretation of PPP message**

### 4.11 Analysis

Having consulted other sources of information and after discussions with other technical professionals, I could not come to a satisfactory answer to why the login/password prompt was not being sent by the peer. I noticed that the RTS, CTS, DTR, CD lines (the hardware handshaking signal lines) on the WMI were hardwired to either the 5V rail or to circuit common. I suspected, comparing physically the differences between the COM ports on a PC and the serial ports on the WMI, that this might be a possible cause for the "silence" of the peer. It was also possible that the peer was sending messages to the modem connected to the WMI. But when the modem tries to retransmit this data to the WMI, the modem software hangs because it did not detect a DTR/DSR, CD or RTS/CTS signal from the WMI (because these are hardwired). I also observed that during the PC-ISP setup the transmit/receive LEDs on the modem were blinking to indicate data transfer while I received the "login/password" prompt. However, the 'receive' LED was not blinking when I used WMI-ISP setup. I did not probe the voltages at the input and output of the modem at that time since I was assuming it was a software problem.

These are some of the causes that need to be looked into in greater detail to resolve the login/password issue. The internal 6805 code on the WMI should also be studied (with help from Grayson Electronics Inc. ®) to see if it is capable of supporting both software and hardware flow control schemes. Once this is accomplished, implementation of the protocol should be mechanical.

During the last few months of my work on this project, my primary focus was to implement the protocol as much as possible so that I would resolve the login/password issue at a later stage with more understanding. It is with these
intentions that code was written to implement the Password Authentication Protocol and parts of the Link Control Protocol configuration options. These segments were successfully tested on the PC-ISP development bed.

I implemented the PPP layers as stand-alone software since I did not intend to complicate this software further by merging it with the message processing application. At the time I was running into roadblocks with the PPP implementation on the WMI, the IVDS project ran out of funding. My work on this project stopped shortly thereafter with the PPP implementation left incomplete.

4.12 Summary

The point to point protocol (PPP) is a standard for communicating over the Internet between two parties at the physical layer. Having gone through the majority of the RFC, I observed that even though the standard does not force us to implement all the options it lists, it does expect the implementation to handle all choices should the other peer require them. This can further increase the complexity of the software.

The level of PPP software implemented on the WMI is able to successfully dial the Internet service provider (ISP), interpret and send out LCP negotiation packets to the ISP. The link connection is broken by the ISP after a time out period. Since this was not a major step towards the PPP goal, it was not demonstrated to the faculty or the sponsor.