Application Note AN208

Connecting the Rabbit via Dial-Up PPP

Introduction

This application note will briefly explain PPP and modem hardware, and describe how to use the Rabbit PPP and modem control libraries to handle a dial-up internet connection to an ISP.

Overview

The first section will discuss the basics of what PPP is. This is followed by a general discussion of modems and ISP negotiation, since the specifics of a dial-up PPP connection are very dependent on modem hardware and the ISP you are connecting to. Finally, the implementation of a PPP dial-up on the Rabbit will be covered.

PPP

Point-to-Point Protocol (PPP) is a versatile protocol for transferring data packets between two hosts over a full-duplex serial link. One of the most common uses of the PPP protocol is the transfer of IP packets between a remote host and an ISP over a modem connection.

Modems

The interface between a modem and a controller is either a true RS232 interface or a variation on RS232 that uses TTL voltage levels for all of the signals. The latter are used by board mounted modem modules. If an external modem is used, an RS232 transceiver chip must is needed to convert RS232 voltages to logic signals and vice versa. A full RS232 connection has 3 outputs and 5 inputs from the controllers point of view. In RS232 terminology, the controller is referred to as the DTE (Data Terminal Equipment). Conversely, modems and other peripherals are referred to as DCE's (Data Communications Equipment). There are single chips such as the Maxim MAX235 which have the correct combination of transmitters and receivers for a full DTE interface.

ISP

An Internet Service Provider (ISP) is a service that allows hosts to connect to the internet by dialing in with a modem and negotiating a PPP link with an ISP host that is directly connected to the internet and will route packets to and from the remote host. Unfortunately there is no universal standard for negotiating this connection with a particular ISP. Although the PPP protocol includes a variety of authentication methods, an ISP will often issue a password challenge in normal text mode before beginning PPP negotiation. This is partly an historical anomaly created during the transition from shell accounts on ISP machines to direct PPP connections, when the same systems were used for both types of services.

Hardware Implementation

The Rabbit implementation for PPP currently uses serial port C on the Rabbit chip. For directly connecting a serial line to the peer, the two serial data lines may be adequate for low speeds (9600 baud max). Higher speed connections through a direct line or modem will usually require flow control. Hardware flow control is implemented for the Rabbit PPP system. It follows the RS232 convention of using RTS and CTS lines. If a modem is used, the additional control signals for the RS232 standard should be connected. The modem control library defines default connections to the Rabbit as follows:

Table 1. Modem Pin Assignments

RS232 Signal	Rabbit Pin	Direction
DTR	PB6	OUT
RTS	PB7	OUT
CTS	PB0	IN
DCD	PB2	IN
RI	PB3	IN
DSR	PB4	IN
TD	PC2	OUT
RD	PC3	IN

Software Implementation

The first stage in dial-up PPP is to establish a modem connection with the ISP. The function **`ModemInit()`** opens the serial port detects if there is a modem connected and ready. It does this by sending “AT” to the modem a set number of times until it receives an “OK” response. This should work with any Hayes-compatible modem, which is the standard

today. At this point the modem is ready and commands can be sent to it using **ModemSend()**. Remember to include a carriage return "\r" at the end of each command sent.

The function **ModemExpect()** is used to wait for a character sequence to occur. Normally the first use of this in a program is to determine that the modem has connected. When a connection occurs, the modem will send a string along the lines of "CONNECT AT x" or something similar. **ModemExpect()** can be set to listen for this. Once connected, the ISP may either attempt PPP negotiation immediately, or request a user name and password first. In the latter case, a sequence of **ModemSend()** and **ModemRequest()** calls are used to handle this.

Eventually the ISP will begin PPP negotiation. At this stage **ModemClose()** should be called to shutdown normal serial operation. After calling **sock_init()** and doing any other necessary TCP/IP initialization, **PPPinit()** is called, followed by any necessary PPP initialization, and finally a call to **PPPstart()**. **PPPstart()** will return after PPP negotiation is complete, or if it has timed out. At this point a link has been established and IP packets can be sent and received normally.

As mentioned before, one of the difficulties with dial-up PPP is an ISP will try to authenticate the dialer before PPP negotiation. There are no real standards for doing this, so each ISP is potentially different. The best way to develop a correct sequence of **ModemSend()** and **ModemExpect()** commands is to connect to the ISP using a terminal program on a PC. You can then take note of the necessary sequence to start PPP negotiation.

Here is a hypothetical session as seen by a terminal program. Note, characters typed in and sent to the ISP or the modem are in bold.

```
AT
OK
ATDT5554545
OK
CONNECT 28800
Welcome to someisp.com
Login?rabbit
Password:Ilikecarrots
Logging in as rabbit
Start PPP $*($}}}}}$}$#$#${@#>>}}FF}}$}
```

From this session we could use **ModemSend()** and **ModemExpect()** to create a dial-up function like this:

```
int myDialUp()
{
   if(ModemOpen(57600) == 0)
   {
      return -1;
   }
   if(ModemInit() == 0)
   {
      return -2;
   }
   ModemSend("ATDT5554545\r");
   if (ModemExpect("OK", 2000) == 0))
   {
      return -3;      //something is wrong with the modem
   }
   //wait up to 30 seconds for modem to connect
   if(ModemExpect("CONNECT", 30000) == 0)
   {
      return -4;      //didn't connect to the ISP
   }
   if(ModemExpect("Login?", 5000) == 0)
   {
      return -5;
   }
   ModemSend("rabbit\r");
   if(ModemExpect("word:", 5000) == 0)
   {
      return -6;
   }
   ModemSend("Ilikecarrots\r");
   if(ModemExpect("PPP", 5000) == 0)
   {
      return -7;      //probably a failed login
   }
   ModemClose();
   sock_init();
   PPPinit(57600);
   PPPflowcontrolOn();
   return 1; //all done
}
```

As you can see, **ModemExpect()** will pick up any part of the received string. Clever use of this allows the initialization to be fairly generic, but subtle differences between ISP's will often require customized sequences such as this.

Tearing down the link must also be done in stages. First, a terminate request must be sent to the peer. This is done with **PPPshutdown()**. **PPPshutdown()** will return once an

acknowledgement has been sent by the peer, or after a time out period. This is followed by a call to **PPPclose**, which unloads the PPP serial driver. If the connection is via a modem, the modem must then be hung up. First the regular serial driver is reopened with **ModemOpen()**. **ModemHangup()** sends the hang up and reset commands to the modem. Finally, a call to **ModemClose()** shuts down the serial driver.

Using Cofunctions

Establishing a PPP connection over a modem is time-consuming. Depending on the baud rate negotiated by the modem, the whole process can take 30 seconds or more. Much of this time is spent by the controller waiting for a response from the other end. In a practical application where the controller has other tasks to perform, this may be unacceptable. For this, there are cofunction versions of all of the functions that wait for responses from the peer. There are still parts of the initialization process that create delays, but the effect is much smaller.

Summary

Dial-up PPP is a reliable and standardized method for low-speed internet connections. Giving an embedded system the ability to use dial-up PPP allows for internet connectivity in situations where ethernet is either not available or not necessary for the application.

REFERENCES

RFC1661, The Point-to-Point Protocol, http://www.ietf.org/rfc/rfc1661.txt

RFC1662, PPP in HDLC-like Framing, http://www.ietf.org/rfc/rfc1662.txt

James Carlson, PPP design and Debugging, Addison Wesley 1998

Rabbit Semiconductor
2932 Spafford Street
Davis, California 95616-6800
USA

Tel. (530)757-8400
FAX (530)757-8402

Web site: http://www.rabbitsemiconductor.com