RABBIT PRODUCT MANUAL

Network Samples Overview

NOTE: There are more networking sample programs provided that are specific to each model of Rabbit device. These model-specific samples are not detailed in this document; please see your product's User's Manual for information on these samples.

This document contains a summary of some of the TCP/UDP/IP sample programs provided for the Rabbit. It can help you find a sample that might be a close match to an application you want to develop. Each sample program is briefly described, along with keywords for the topic of each program. Almost all of the samples listed in this file are generic—that is, they should run on Ethernet, Wi-Fi, or PPP connections.

Some of the samples listed in this document call the TCP functions directly, and some use application libraries for protocols such as FTP and HTTP. There are also samples for the µC/OS II multi-tasking environment ported by Rabbit Semiconductor.

The sample programs are divided into several broad categories:

- Generic TCP
- Generic UDP
- Generic Internet/ICMP
- Wi-Fi
- Point-to-Point Protocol (PPP)
- RabbitWeb
- SNMP
- µC/OS-II

If you know a protocol, you can just search this text file. Each entry starts with the filename of the sample program, followed on the next line by keywords relevant to the sample. After that is a summary of the program, highlighting its functions and/or features.

Table of Contents

Generic TCP Samples

This group of programs contains TCP samples that work on any Ethernet or Wi-Fi board. They can be modified to use PPP instead. These samples are generally the higher protocols, such as HTTP, telnet, and SMTP.

`Samples\Tcpip\active.c`

Keywords: TCP, http

Retrieves a web page from an HTTP server. All addresses are compiled in. Demonstrates using **sock_established()** and **sock_bytesready()** to make sure the socket is connected.

`Samples\Tcpip\echo.c`

Keywords: TCP, listen

This program demonstrates the **tcp_listen()** call. A basic server, that when a client connects, echoes back to them any data that they send. Data also displayed on Stdout. Can be configured to listen on different TCP ports.

`Samples\Tcpip\multi_echo.c`

Keywords: TCP, listen

Like `echo.c`, this sample shows how to construct an echo server that simply sends back any data that it receives. This sample, however, also shows how to have multiple echo servers operating on multiple TCP ports and even on multiple network interfaces (e.g., both Ethernet and PPP).

`Samples\Tcpip\nist_time.c`

Keywords: TCP daytime, RFC867, RFC868

Accesses the current time from a well-known “daytime server.” Decodes the time and updates the Rabbit’s RTC with the time.

`Samples\Tcpip\serialexa.c`

Keywords: TCP, telnet, serial

This program directs all of the data from a serial port to a TCP port and vice versa. Demonstrates buffering and timeouts. Can handle multiple connections through a state machine it uses.

`Samples\Tcpip\state.c`

Keywords: TCP, HTTP server

This program demonstrates building a state machine-based Internet server. The example is a simple web server (it doesn't use HTTP.LIB). Study the usage of **sock_dataready()** and **sock_fastwrite()** here when implementing your own server.

`Samples\Tcpip\tcp_time.c`

Keywords: TCP, UDP, time, RFC868

Demonstrate how to talk to one or more "Time Protocol" servers (see RFC868 at **http://www.faqs.org/rfcs/rfc868.html**). Code includes both client and server. Permits Rabbit to get a reliable wall-clock time from the network. Uses TCP/UDP to connect to a sequence of Time servers. The list of servers is specified in an initialized array. You might like to use this as a basis for adding a reliable wall-clock time. Note that UDP is used to query the servers, but the target allows both TCP and UDP queries to its server sockets. This sample also demonstrates the extended UDP functions and waiting for DNS name-to-IP resolving (some of the finer details or DCRTCP).

`Samples\Tcpip\tcp_timemulti.c`

Keywords: TCP, UDP, time, RFC868

This demo is same as tcp_time.c, except that multiple interfaces may be specified. Demonstrate how to talk to one or more "Time Protocol" servers (see RFC868 at **http://www.faqs.org/rfcs/rfc868.html**). Code includes both client and server. Permits Rabbit to get a reliable wall-clock time from the network. Uses TCP/UDP to connect to a sequence of Time servers. The list of servers is specified in an initialized array. You might like to use this as a basis for adding a reliable wall-clock time. Note that UDP is used to query the servers, but the target allows both TCP and UDP queries to its server sockets. This sample also demonstrates the extended UDP functions and waiting for DNS name-to-IP resolving (some of the finer details or DCRTCP).

`Samples\Tcpip\Bsd\bsd.c`

Keywords: resolve, TCP, connecting, name service, domain name

Opens a port and waits for connections. Demonstrates some of the informational routines in the TCP stack, including set/gethostname(), set/getdomainname(), getsockname(), and getpeername(). Also how to format for printing the IP address.

`Samples\Tcpip\ftp\ftp2fat.c`
Keywords: FTP client, FAT

Downloads a file from a remote FTP server and saves it to the FAT filesystem on the Rabbit. Requires hardware with mass storage (NAND flash, serial flash, SD card, etc.).

`Samples\Tcpip\Ftp\ftp_client.c`
Keywords: FTP client

Demonstration of the use of the ftp_client library to download and upload files. Uses both "standard" mode and "data handler" mode. Optionally, also includes FTP server library making it do the same things as the FTP_SERVER.C sample program.

`Samples\Tcpip\Ftp\ftp_fat.c`
Keywords: FTP server, FAT

Demonstration of a simple FTP server, using the ftp library, that allows file uploads and deletion. This sample uses the FAT library for storage.

`Samples\Tcpip\Ftp\ftp_server.c`
Keywords: FTP server, sspec

Demonstrates the Rabbit FTP server library. Two files are given to user “anonymous” and three files for user “foo” who can see the first two files, but not the other way around. One file is created from **xstring** text. Uses **sspec_addxmemfile()** to add compiled in files, and **sauth_adduser()** to create authorized users, joined together with **sspec_setuser()**.

`Samples\Tcpip\Http\authentication.c`
Keywords: HTTP, authentication

This sample shows how to use HTTP authentication (both basic mode and digest mode) with the HTTP (web server) library. Authentication allows restricting access to content based on username/password credentials.

`Samples\Tcpip\Http\cgi.c`
Keywords: HTTP, CGI

Demonstrates a CGI function with the HTTP server library. **HTTPSPEC_FUNCTION** means a function will be creating the whole page dynamically. This program outputs a simple counter, incremented each time the CGI page is accessed. The function **http_date_str()** is used in the header, and **cgi_sendstring()** sends the whole response to the browser. The counter is not referenced using SSI.

`Samples\Tcpip\Http\cgi_concurrent.c`

Keywords: HTTP, CGI

This program demonstrates some more complex CGI programming. In particular, it demonstrates how to use the user data area within each HTTP state structure, as well as how to share access to a resource. These techniques allow for multiple concurrent CGI processes.

`Samples\Tcpip\Http\flashlog.c`

Keywords: HTTP, Forms, SSPEC, CGI, Flash memory

The program logs users that request a controller's page. The log can be viewed online. Uses CGI pages to make entries, clear and display the access log. A form to alter the date uses **sspec_addform()** to construct it dynamically. Form inputs are programmed with a maximum width and a specific valid numeric range for each field.

`Samples\Tcpip\Http\form1.c`

Keywords: HTTP, Forms, SSPEC

Demonstrates dynamically building adding pages to the HTTP server. These pages have forms with two dependent variables. Software validates this, and rejects incorrect values. See the RabbitWeb software module for a better way to solve this problem.

`Samples\Tcpip\Http\form2.c`

Keywords: HTTP, Forms, SSPEC, authentication

An example of a simple HTTP form generation and parsing program with password protection of the form. This is similar to form1.c, but with the addition of authentication.

`Samples\Tcpip\http\http2fat.c`

Keywords: HTTP client, FAT

Downloads a file from a remote HTTP (web) server and saves it to the FAT filesystem on the Rabbit. Requires hardware with mass storage (NAND flash, serial flash, SD card, etc.).

`Samples\Tcpip\http\http client.c`

Keywords: HTTP client

Demonstrates use of the HTTP client library to open a connection to a remote HTTP (web) server. Parses headers and can optionally set the Rabbit’s RTC to match the server’s clock.

`Samples\Tcpip\Http\http_upld.c`
Keywords: HTTP server, CGI, upload

Demonstrates how to upload a file to the HTTP server through a CGI function. This program simply displays the uploaded information on the stdio window.

`Samples\Tcpip\Http\httpupld2.c`
Keywords: HTTP server, CGI, upload

Demonstrate the HTTP file upload facility. This is the same demo as httpupld.c, except that it generates the HTML response in the CGI function itself (instead of using http_switchCGI() to respond with a pre-configured HTML file).

`Samples\Tcpip\Http\multiweb.c`
Keywords: HTTP server

This sample shows a web server running on multiple network interfaces simultaneously, such as Ethernet and PPP.

`Samples\Tcpip\Http\post.c`
Keywords: HTTP server, CGI, form post

Maintain two string fields posted from a form. Includes logic to decode the form URL sent back on submit.

`Samples\Tcpip\Http\post2.c`
Keywords: HTTP server, CGI, form post, SSPEC

Extends the **http\post.c** example to also control LEDs on the prototype board. Once the user registers themselves, a cookie is sent to the browser. Whenever the user changes an LED, a user-specific log entry is created. This tracks (records) the user's actions based on their cookie. Uses the **SSPEC_RESOURCETABLE** to statically define the web site's contents.

`Samples\Tcpip\Http\post2a.c`
Keywords: HTTP server, CGI, form post, ZSERVER

This is an alternate version of **http\post2.c** that uses the **ZSERVER.LIB** functionality to build the server table. Comparison with **post2.c** shows how a program can be converted from using **SSPEC_RESOURCETABLE** to **ZSERVER.LIB** functionality. This uses the **http\post.c** style form submission, and the **ssi.c** style dynamic pages to build a fully functional and audited controller. At the user's first access the page, they enter their name into a form, which is then stored in a HTTP cookie. This is used to later build an audit trail of what changes each user makes.

`Samples\Tcpip\Http\refresh.c`

Keywords: HTTP server, SSI variable, auto-refresh

Sets up JavaScript to keep on refreshing a web page. The page has an SSI variable that increments on each refresh. Refresh done with:

```
<BODY onLoad=window.setTimeout("location.href='index.shtml'",1000)>
```

`Samples\Tcpip\Http\ssi.c`

Keywords: HTTP server, dynamic web page, SSI

Uses SSI (server-side includes) to display the state of four virtual lights. After the user hits a (web page) button, the form submits to update the LED states. This program simulates the LED's. See your device's user manual for a program that can change actual LED's on your device.

`Samples\Tcpip\Http\ssi2.c`

Keywords: HTTP server, dynamic web page, SSI, peer name

The same lights and buttons from http\ssi.c, with an audit log of who made what changes, encoded as a hash of the remote user's IP address using getpeername(). The audit log is a circular queue.

`Samples\Tcpip\Http\ssi2_fat.c`

Keywords: HTTP server, dynamic web page, SSI, peer name, FAT

This is the same as ssi2.c, except that the FAT filesystem is used to contain the web pages and images.

`Samples\Tcpip\Http\static.c`

Keywords: HTTP server, static web page

A very basic web server example with static pages. This program completely initializes the library, outputting a basic static web page. Contains lots of comments explaining the defines and settings.

`Samples\Tcpip\Http\static2.c`

Keywords: HTTP server, static web page, xmemory file, FTP server

Basic web server example that serves its "index.html" page from xmemory. It can be uploaded with FTP.

`Samples\Tcpip\Http\upld_fat.c`

Keywords: HTTP server, FAT, upload

Demonstrate the HTTP file upload facility, using the default handlers. This sample uses the FAT filesystem so that there is somewhere we can save the uploaded file.

`Samples\Tcpip\Http\zimport.c`

Keywords: HTTP server, zimport

This program uses the ZIMPORT.LIB library to compress web pages that are served by the HTTP server. It demonstrates a couple of different ways in which compressed files can be used with the HTTP server.

`Samples\Tcpip\Pop\parse_extra.c`

Keywords: POP client, read E-mail

Reads and downloads E-mail from a POP3 server. Defines **POP_PARSE_EXTRA** to displays messages in a nice format, separating header fields from the message body.

`Samples\Tcpip\Pop\pop.c`

Keywords: POP client, read E-mail

Reads and downloads E-mail from a POP3 server. Displays messages.

`Samples\Tcpip\Smtp\smtp.c`

Keywords: send E-mail, SMTP

Sends an E-mail using a root memory string. Also supports SMTP authentication.

`Samples\Tcpip\Smtp\smtp_dh.c`

Keywords: send E-mail, SMTP

A small program that uses the SMTP library to send an e-mail. This makes use of the smtp_data_handler() function to generate message data on-the-fly.

`Samples\Tcpip\Smtp\smtpxmem.c`

Keywords: send E-mail, SMTP

Sends an E-mail, taking the message body from xmemory. Also supports SMTP authentication

`Samples\Tcpip\Telnet\rxsample.c`

Keywords: telnet server, serial

Starts up a listen on TCP port 23. When a connection is established, all writes from the remote host are sent to Stdout (displayed in the Dynamic C IDE). This sample doesn't provide a way to send data to the remote host.

`Samples\Tcpip\Telnet\vserial.c`

Keywords: telnet server, virtual serial driver

Uses **VSERIAL.LIB** to create a bidirectional stream between a telnet port and a serial port on the Rabbit. It extends the application, **TcpIp\telnet\rxsample.c**, from just output to a serial port to bidirectional communication. Connection will be re-established after it's closed

`Samples\Tcpip\zserver\fat_serve.c`

Keywords: ZServer, FAT, HTTP

Sample program to demonstrate use of ZSERVER.LIB and FAT filesystem functionality. This is the 2nd of a 2-part sample. The first part (which you should have run already) is FAT_SETUP.C. This sample sets up a web server which accesses the files which were copied to the FAT filesystem by the previous sample.

`Samples\Tcpip\zserver\fat_serve2.c`

Keywords: ZServer, FAT, HTTP

Sample program to demonstrate use of ZSERVER.LIB and FAT filesystem functionality. This is the 2nd of a 2-part sample. The first part (which you should have run already) is FAT_SETUP.C. This sample sets up a web server which accesses the files which were copied to the FAT filesystem by the previous sample. It is similar to FAT_SERVE.C, except that access permissions have been added.

`Samples\Tcpip\zserver\fat_setup.c`

Keywords: ZServer, FAT, HTTP

Sample program to demonstrate use of ZSERVER.LIB and FAT filesystem functionality. This is the first of a 2-part sample. The other part (run next) is FAT_SERVE.C, or you can run FAT_SERVE2.C (which adds access control), or you can run samples\tcpip\http\ssi2_fat.c which adds FAT access to the ssi2.c sample. This sample copies some #ximported files into the FAT filesystem. You should make sure that the FAT filesystem is already formatted (see the FAT samples for how to do this).

`Samples\Tcpip\zserver\filesystem.c`

Keywords: ZServer, FAT, HTTP, FTP

This program demonstrates how to use the features of the ZServer.lib library. This library is an abstraction layer for TCP/IP servers (in particular, the HTTP and FTP servers) that allows those servers to manage files in xmem, root memory, or the FAT filesystem. This sample shows a lot of the run-time capabilities of the ZServer library.

`Samples\zconsole\userblock_tcpipconsole.c`

Keywords: ZConsole, TCP, user block, HTTP

This example of a ZCONSOLE.LIB program demonstrates how the console configuration information can be stored in the user block.   This sample also demonstrates the use of the extended TCP/IP configuration options, including setting multiple name servers, enabling DHCP and ping configuration, and handling multiple interfaces.

Generic UDP Samples

This group of programs contains UDP samples that work on any Ethernet board.

`Samples\Tcpip\addp.c`

Keywords: UDP

Demonstrates ADDP.LIB, an implementation of Digi's Advanced Device Discovery Protocol. The programs in the Utilities/ADDP directory can discover and change the network configuration of a Rabbit running addp.c. Works with Ethernet and Wi-Fi only – not compatible with PPP.

`Samples\Tcpip\multicast.c`

Keywords: UDP, multicast, IGMP

This sample program sends and receives heartbeat packets via multicast UDP sockets. Multiple sockets can be configured to operate at once. IGMP can be enabled so that the multicast datagrams can be routed across subnets. Multicasting is useful when information needs to be transmitted to a group of other devices. All of these devices can be listening on a single multicast group.

`Samples\Tcpip\sntp_time.c`

Keywords: UDP, time, RFC2030

Demonstrate how to talk to one or more SNTP servers (see RFC2030 at http://www.faqs.org/rfcs/rfc2030.html) in order for the Rabbit to get a reliable wall-clock time from the network. This sample based on tcp_time.c, except that we use the more modern and well supported SNTP protocol. The results are much more accurate, typically within a few 10's of ms.

`Samples\Tcpip\Tftp\tftp.c`

Keywords: TFTP server (Trivial FTP)

Implements a TFTP server that can send one file and receive into another area. (The file system is not used here.) Eliminates all TCP buffers (to reduce RAM footprint). Although the code stores the file in a fixed-sized block of root memory, commented code shows how to store it in xmemory.

`Samples\Tcpip\Tftp\tftpclnt.c`

Keywords: TFTP client (Trivial FTP)

Uses TFTP to download one file from the server, then upload the same file. The file is stored into xmemory. Requires a remote host to act as the TFTP server.

`Samples\Tcpip\Udp\udp_cli.c`
Keywords: UDP send

Uses **udp_send()** to transmit a “heart beat” datagram once a second. Can be sent to a specific host, or broadcast to the whole local network.

`Samples\Tcpip\Udp\udp_echo_dh.c`
Keywords: UDP, data handler

A UDP example, that receives packets from any host on UDP port 7, then echoes the received packet back to the sender. This demonstrates use of a UDP data handler, which is the fastest way to implement UDP-based request/response servers.

`Samples\Tcpip\Udp\udp_srv.c`
Keywords: UDP receive

Uses **udp_recv()** to collect heartbeat datagrams. Since it doesn’t use the extended UDP receive function, the first client to send a packet “sets” which host subsequent datagrams will be received from.

Generic Internet/ICMP Samples

This group of programs include examples of printing the Ethernet MAC address, starting up a DHCP client, resolving a domain name using the DNS library, and the ICMP ping feature. They don't use application-level protocols such as HTTP or FTP.

`Samples\Tcpip\dhcp.c`

Keywords: DHCP, debugging

Configures the network interface automatically via DHCP. Note that you must have a DHCP server available for this sample to work. This sample has a number of debugging and diagnostic features as well, which makes it a very good test program. For example, you can change the network configuration, ping other devices, access a web server, and display configuration information.

`Samples\Tcpip\dhcp_bootfile.c`

Keywords: DHCP, TFTP

Configures the network interface automatically via DHCP and also downloads a bootfile via TFTP if available. This bootfile may be specified by the DHCP server.

`Samples\Tcpip\md5_test.c`

Keywords: MD5

Simple demonstration of MD5 hashing library. MD5 takes any amount of data and produces a 16-byte hash value of it.

`Samples\Tcpip\netrand.c`

Keywords: random number generator

Demonstrates how to use the arrival of packets to add randomness to the entropy table for a random number generator. This can improve the "randomness" of the random number generator.

`Samples\Tcpip\virtualeth.c`

Keywords: virtual Ethernet

Demonstrates the use of virtual Ethernet interfaces. These are multiple logical interfaces on a single physical Ethernet interface. This could be used to, for instance, replace a number of network devices with a single network device. The advantage of this method is that the system that communicates with these devices would not need to have its configuration or code changed.

`Samples\Tcpip\bsd\bsd.c`

Keywords: getpeername(), sethostname(), setdomainname(), getsockname()

The Rabbit TCP stack does not use the UNIX sockets API. However, it does support some of the naming function calls. This program opens a TCP port and provides information about which hosts connect to it.

`Samples\Tcpip\DNS\dns.c`

Keywords: DNS, resolve

Demonstration of how to look up an IP address through a DNS server. This sample uses the blocking resolve() function.

`Samples\Tcpip\DNS\dns2.c`

Keywords: DNS, resolve

Manually goes through of process of resolving a network name into an IP address. Uses **resolve_name_check()** for non-blocking operation. Uses **MY_DOMAIN** for the default domain (for not fully qualified domain names). This function looks up multiple host names simultaneously. It is organized around a state machine, with separate state information maintained for each request.

`Samples\Tcpip\display_mac.c`

Keywords: MAC, Ethernet

Displays the board's Ethernet MAC (Media Access Connection) "station" address, which is unique to the board. Uses a packet driver function to return the MAC. Rabbit uses a range of addresses; resellers might want to change it to a value inside their own range. Not the same as the IP, which can be bound to different network hardware addresses. The program then loops calling **tcp_tick()**, so you can ping it from another computer.

`Samples\Tcpip\ping.c`

Keywords: ICMP, ping, resolve

Uses compiled in addresses for the host and the target computer to send an ICMP ping request to. Does this once a second using **_ping()** to handle the ICMP transmission.

`Samples\Tcpip\icmp\pingme.c`
`Samples\Tcpip\icmp\pingyou.c`

Keywords: ICMP, ping, resolve

The program **pingyou.c** will send out a series of ICMP ping requests. The program **pingme.c** just sets up the TCP stack and then calls **tcp_tick()**, which gives time to the TCP stack to handle the ping requests.

Wi-Fi Samples

The programs in this section have been designed for Rabbit devices with an integrated Wi-Fi networking interface. The Dynamic C TCP/IP stack works unchanged for Wi-Fi devices, but Wi-Fi does have some additional configuration compared to Ethernet. Note that almost all of the other samples in this document will also work for Wi-Fi devices.

`Samples\WiFi\WiFiDHCPorStatic.c`

Keywords: Wi-Fi, DHCP, configuration

This sample program demonstrates the runtime selection of a static IP configuration or DHCP. Please see the DHCP.C for a larger example of using DHCP with your application.

`Samples\WiFi\WiFiMultipleAPs.c`

Keywords: Wi-Fi, configuration

This code demostrates changing AP's using WEP keys. It associates the module with the first AP (AP_0 defined below) with WEP key KEY0 (defined below). After associating, it waits for a predefined time period, and then pings the Ethernet address of the AP (AP_ADDRESS_0). Next, it associates with the second AP and pings its Ethernet address (AP_1, KEY1, AP_ADDRESS_1). Then back to the first AP, etc.

`Samples\WiFi\WiFiPingYou.c`

Keywords: Wi-Fi, ping, ad-hoc

This sample program will send out a series of 'pings' to another computer on the network. This is similar to the samples\tcpip\icmp\pingyou.c sample, except it uses an ad-hoc Wi-Fi network.

`Samples\WiFi\WiFiScan.c`

Keywords: Wi-Fi, scan

This code demostrates how to scan Wi-Fi channels for SSID's using the ioctl WIFI_SCAN function call. ioctl WIFI_SCAN takes a while to complete, so it calls a callback function when it is done. The callback function is specified using an ioctl WIFI_SCANCB function call.

`Samples\WiFi\WiFiScanAssociate.c`

Keywords: Wi-Fi, scan, configuration

This code demostrates how to scan Wi-Fi channels for SSID's using the ioctl WIFI_SCAN function call. It also allows the user to choose an SSID from the scan list with which to associate. Once association has completed, then another device can ping the Wi-Fi device assuming that the other network parameters have been correctly configured.

`Samples\WiFi\Regulatory\region_compiletime.c`

Keywords: Wi-Fi, configuration, regulatory

This program demonstrates how you can setup your Wi-Fi device at compile time to run in a given region to meet power and channel requirements.

`Samples\WiFi\Regulatory\region_multi_domain.c`

Keywords: Wi-Fi, configuration, regulatory, 802.11d

This program demonstrates how the multi-domain option (802.11d) can be used for configuring your device to meet regional regulations. The demo also includes doing pings, which will indicate that the Wi-Fi device has successfully received country information from your access point.

`Samples\WiFi\Regulatory\region_runtime_ping.c`

Keywords: Wi-Fi, configuration, regulatory

This program demonstrates how the region setting can be set at runtime for configuring the Wi-Fi device to meet regional regulations. It also shows how you can save and retrieve the region setting from non-volatile memory.

Point-to-Point Protocol (PPP) Samples

These applications demonstrate PPP (and PPP over Ethernet) features.

`Samples\Ppp\pppoe_test.c`

Keywords: PPP, PPPoE, PAP, DHCP, LCP, dynamic IP, ISP, ximport

Connects using Point-to-point over Ethernet protocol on a serial port. Rabbit will obtain its IP address from the server. Requires setting the PAP username and password.

`Samples\Ppp\ppp_answer.c`

Keywords: PPP, dynamic IP, LCP, PAP, AT commands

Waits for a phone call and explicitly answers it, then authenticates to the caller (i.e. the ISP calls the Rabbit, great for remote sensor stations). It obtains an IP address from the caller (great for remote sensor stations). While connected, the web server is available. Loops around to wait for another call.

`Samples\Ppp\modem_test.c`

Keywords: PPP, dynamic IP, SMTP, PAP

Dials an ISP and authenticates. Once connected, an E-mail is sent to the defined SMTP server. The “From,” “To” and “Body” portions are hard-coded (compiled in). Then it hangs up. This test is repeated three times.

`Samples\Ppp\modem_server.c`

Keywords: PPP, LCP, PAP, web server

Dials an ISP and authenticates to it. The Rabbit obtains an IP address. While connected, the web server is available.

`Samples\zconsole\pppconsole.c`

Keywords: ZConsole, PPP, LCP, PAP, web server

This sample program demonstrates the PPP configuration commands of ZCONSOLE.LIB. This sample also uses the user block for storing configuration information.

RabbitWeb Samples

These programs all make use of the RabbitWeb extensions for the HTTP server library. These extensions ease the chore of creating a web interface for your device with complex error checking, error reporting, and error correction. It is particularly useful in creating configuration interfaces or monitoring interfaces.

`Samples\Tcpip\rabbitweb\arrays.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of arrays in the RabbitWeb HTTP enhancements. This sample shows how to register array variables for use in your web pages, as well as how to use them within your web pages.

`Samples\Tcpip\rabbitweb\auth.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of authentication features for variables in the RabbitWeb HTTP enhancements. These authentication features allow you to restrict which users can update which variables.

`Samples\Tcpip\rabbitweb\ethernet_to_serial.c`

Keywords: HTTP, RabbitWeb

Uses the RabbitWeb HTTP enhancements to configure a simple Ethernet-to-serial converter. Each serial port can be associated with a specific TCP port. The Rabbit will listen on each of these TCP ports for a connection, which will then be associated with a specific serial port. Data will then be shuttled between the serial and Ethernet connections. Note that this sample can also be used for Wi-Fi interfaces to make a Wi-Fi-to-serial converter.

`Samples\Tcpip\rabbitweb\humidity.c`

Keywords: HTTP, RabbitWeb

This is a simple RabbitWeb sample based on a humidity monitor. It demonstrates some simple #web variable registration along with the authentication features.

`Samples\Tcpip\rabbitweb\selection.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of the selection variables feature in the RabbitWeb HTTP enhancements. Selection variables allow for easy implementation of pull-down menus in web pages.

`Samples\Tcpip\rabbitweb\structures.c`

Keywords: HTTP, RabbitWeb

Shows the use of structures in the RabbitWeb HTTP enhancements. This sample shows how to register structure variables for use in your web pages, as well as how to use them within your web pages.

`Samples\Tcpip\rabbitweb\updating.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of the updating() feature in the RabbitWeb HTTP enhancements. This feature allows a web page to behave differently based on whether it is being loaded as a result of a user update or not. For example, this program redirects the web browser to a success page after the user has updated information in the web interface.

`Samples\Tcpip\rabbitweb\web.c`

Keywords: HTTP, RabbitWeb

Demonstrates the basic use of the #web registration statement in the RabbitWeb HTTP enhancements. This statement registers global variables in the C program with the web server such that ZHTML scripting tags can be used to access them.

`Samples\Tcpip\rabbitweb\web_error.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of the WEB_ERROR() feature, along with displaying these errors in web pages, in the RabbitWeb HTTP enhancements. WEB_ERROR() can be used to display an error message specific to the error in the user input.

`Samples\Tcpip\rabbitweb\web_update.c`

Keywords: HTTP, RabbitWeb

Demonstrates the use of the #web_update feature in the RabbitWeb HTTP enhancements, which allows the program to be notified when certain variables have acquired new values.

SNMP Samples

This group of samples requires the SNMP (Simple Network Management Protocol) module which is available for purchase from Rabbit. The SNMP module implements SNMP version 1 and is used to manage and monitor network devices. SNMP is often used for network routers, switches, and printers. The SNMP module provides API-level access to building a MIB (Management Information Base).

`Samples\Tcpip\snmp\mibtest.c`

Keywords: SNMP, MIB

This demo shows only the MIB (Management Information Base) part of the SNMP suite. It allows you to interactively add and delte objects, and examine the MIB tree.

`Samples\Tcpip\snmp\snmp1.c`

Keywords: SNMP, MIB

This sample demonstrates how to use the SNMP library. It defines a number of read-only and read-write managed variables which can be browsed and manipulated from an SNMP browser. It also demonstrates the use of SNMP traps.

SSL/TLS Samples

These samples show how to use the Rabbit SSL/TLS module. The SSL module provides support for HTTPS, which is HTTP over SSLv3 and TLSv1 (TLS is the successor to SSL). Essentially, this means that the software provides for a secure web site where all communication between the web browser and web server is encrypted. To use these samples, you will first need to create a certificate for your web server. See the SSL module documentation as well as the SSL certificate generator utility in Utilities\SSL_Utilities\certificate.exe.

`Samples\Tcpip\ssl\https\ssl_authentication.c`

Keywords: SSL, TLS, HTTP, HTTPS, authentication

This sample is a modification of authentication.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. This sample shows how to use HTTP authentication (both basic mode and digest mode) with the HTTP (web server) library. Authentication allows restricting access to content based on username/password credentials.

`Samples\Tcpip\ssl\https\ssl_cgi.c`

Keywords: SSL, TLS, HTTP, HTTPS, CGI

This sample is a modification of cgi.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. Demonstrates a CGI function with the HTTP server library. **HTTPSPEC_FUNCTION** means a function will be creating the whole page dynamically. This program outputs a simple counter, incremented each time the CGI page is accessed. The function **http_date_str()** is used in the header, and **cgi_sendstring()** sends the whole response to the browser. The counter is not referenced using SSI.

`Samples\Tcpip\ssl\https\ssl_form.c`

Keywords: SSL, TLS, HTTP, HTTPS, Forms, SSPEC

This sample is a modification of form1.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. Demonstrates dynamically building adding pages to the HTTP server. These pages have forms with two dependent variables. Software validates this, and rejects incorrect values. See the RabbitWeb software module for a better way to solve this problem.

`Samples\Tcpip\ssl\https\ssl_ssi.c`
Keywords: SSL, TLS, HTTP, HTTPS, dynamic web page, SSI

This sample is a modification of ssi.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. Uses SSI (server-side includes) to display the state of four virtual lights. After the user hits a (web page) button, the form submits to update the LED states. This program simulates the LED's. See your device's user manual for a program that can change actual LED's on your device.

`Samples\Tcpip\ssl\https\ssl_static.c`
Keywords: SSL, TLS, HTTP, HTTPS, static web page

This sample is a modification of static.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. A very basic web server example with static pages. This program completely initializes the library, outputting a basic static web page. Contains lots of comments explaining the defines and settings.

`Samples\Tcpip\ssl\https\ssl_zimport.c`
Keywords: SSL, TLS, HTTP, HTTPS, zimport

This sample is a modification of zimport.c (in the HTTP samples folder) to do HTTP authentication over HTTPS, secured by SSL. This program uses the ZIMPORT.LIB library to compress web pages that are served by the HTTP server. It demonstrates a couple of different ways in which compressed files can be used with the HTTP server.

µC/OS II Samples

This group of programs are µC/OS-II samples. Be aware that a 2K byte stack is required for the single thread that makes calls to the network libraries.

`Samples\Tcpip\Ucos\ucos2.c`

Keywords: TCP, uCOS, telnet, serial

A sample data-gathering application that combines a task reading from the serial port with another task to report this data to the user over the network, with associated local messaging and processing. Using µC/OS tasks, creates a reader and a writer task. One side controls a serial port, and the other works with a TCP socket. Data is forwarded and read from the other side. A semaphore regulates access to shared data (i.e. the buffers between the two tasks). Currently, at most one task can call any socket functions, and that task’s stack must be at least 2K in size.

`Samples\UCos-II\ucostcpip.c`

Keywords: TCP, uCOS, HTTP

Example of using TCP/IP and uC/OS together. In this example uC/OS is monitoring the network link and reflecting the link light on the "LEDs" on the development board. At the same time, this program has a ECHO server and HTTP server running. This program can be easily modified to use actual LEDs on a protoboard rather than stdio messages.