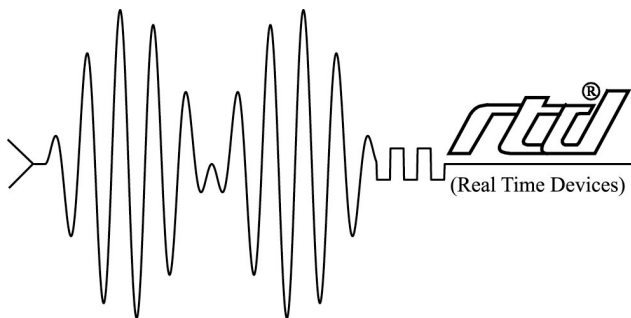


APPLICATION NOTE

A socket-based networking program for RTD CPU modules



RTD Embedded Technologies, Inc.

"Accessing the Analog World"®

SWM-64000015
Rev. A

APPLICATION NOTE
A socket-based networking program
for
RTD CPU modules



RTD Embedded Technologies, INC.

103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087

FAX: +1-814-234-5218

E-mail

sales@rtd.com

techsupport@rtd.com

web site

<http://www.rtd.com>

Revision History

Rev. A

Application note created on 15.04.2004

Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.

All rights reserved

Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Introduction

This application note reviews a networking demonstration program package designed for RTD CPU modules with Ethernet interface. The program package contains two example programs: a client and a server application. The programs are demonstrating how can be interconnected two computers on a LAN/WAN network if they are running Windows operating system. The connection established on an IP-based system, the server application creates a socket with a specific port number on the server computer, allowing to a client application on a remote computer to connect in. On this communication channel the client and the server computers are able to exchange data on a fast and reliable way.

Theoretical basics

The current Windows versions are supporting Windows Sockets 2. A "socket" is an endpoint of communication: an object through which your application communicates with other Windows Sockets applications across a network. Sockets generally exchange data with other sockets in the same "communication domain," which uses the Internet Protocol Suite.

Two socket types are available:

Stream sockets: a data flow without record boundaries: a stream of bytes. Streams are guaranteed to be delivered, correctly sequenced and unduplicated.

Datagram sockets: a record-oriented data flow that is not guaranteed to be delivered and may not be sequenced as sent or unduplicated.

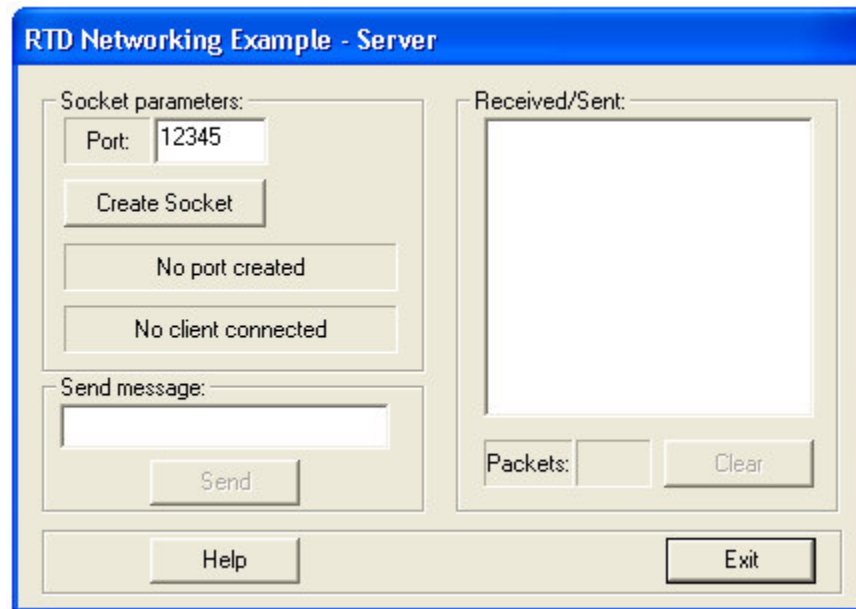
The purpose of Windows Sockets is to abstract away the underlying network so that you do not have to be knowledgeable about that network and so your application can run on any network that supports sockets.

In our Client/Server application we are using stream sockets and applying the MFC support of the Microsoft Visual Studio C/C++ compiler. The Microsoft Foundation Class Library (MFC) supports programming with the Windows Sockets API by supplying two classes (Csocket and CAsyncSocket). CSocket provides a high level of abstraction to simplify your network communications programming.

Realization of the application

Our networking application uses a client/server model as communication context.

Server:



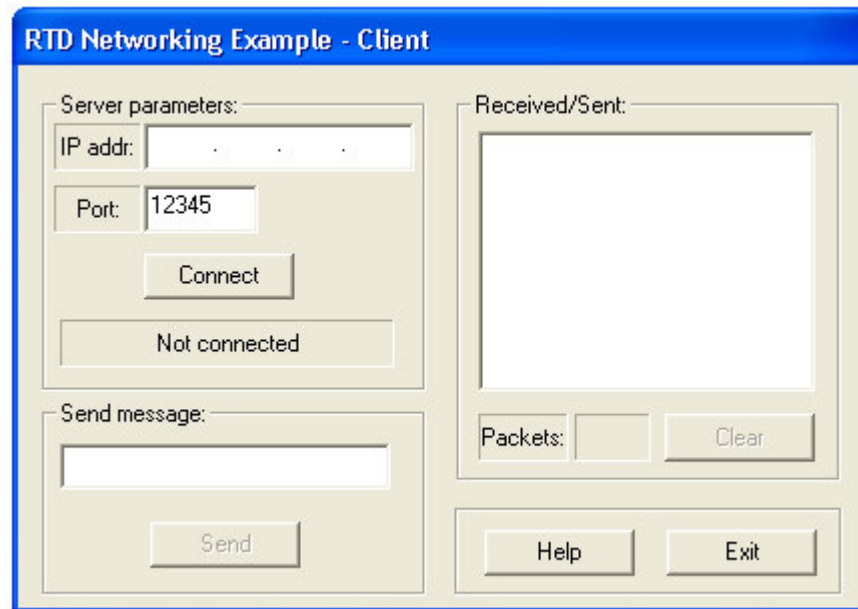
After starting the Server and give a PORT number, by clicking the 'Create Socket' button a socket is created for listening the incoming connect requests using the following code in the member function **OnListen()**:

```
// initialize Windows sockets
if(AfxSocketInit(NULL)==0)
MessageBox("Socket Init Failed","Error",MB_OK);
// Create STREAM type Windows socket and attach it
if(sock_SBase.Create(PORT)==0)
MessageBox("Socket Create Failed","Error",MB_OK);
// Start listen for incoming connection requests
if(sock_SBase.Listen()==0)
MessageBox("Socket Listen Failed","Error",MB_OK);
```

When a new connect request is received from a client, the member function **OnAccept** in the listening socket class is notified by the system.

In the OnAccept function, a new socket is created for the new client. We use the reference to this object through the program for sending or receiving messages. The system reads the incoming message in the member function **OnReceive** in the client socket and processes it.

Client:



After you start the Client and enter the IP address and PORT number of the server running the server socket, a socket is created using the following code in the member function **OnConnect()**:

```
// initialize Windows sockets
if(AfxSocketInit(NULL)==0)
MessageBox("Socket Init Failed","Error",MB_OK);

// Create STREAM type Windows socket and attach it
if(sock_CBase.Create()==0)
{
    sprintf(szBuf,"Socket Create Failed %d",GetLastError());
    MessageBox(szBuf,"Error",MB_OK);
}

// Connect at IP address
if(!sock_CBase.Connect(ip, PORT))
MessageBox("Connect failure","Error",MB_OK);
```

Please make sure that the PORT number is the same in the client and server application.
The system reads the incoming messages in the client socket class **OnReceive** member function.

The applications are using a special structure for the communication in the server → client transfer:

```
struct SCAN
{
char    scanline[256];
        BYTE counter;
        BOOL newdata; // used by client only
};
```

The scanline member is for general purpose data transmission. The counter counts the number of transmitted communication packets.

Additionally, the client application sends the local messages in a simple text format to the server.

Notes

The programs are demonstrating the client/server type networking. The communication data structure can be changed according to the special requirements.

The server application is written to accept one client connection. The base socket class of the server responds to the client connection requests with the **OnAccept** function. In this function a new socket class is generated for the client. If multiple client connection is required, the **OnAccept** function should be changed.