Agilent E5070B/E5071B ENA Series RF Network Analyzers

Control over LAN using Microsoft Excel

Second Edition

Agilent Technologies

No. 16000-95012

August 2002

Notices

The information contained in this document is subject to change without notice.

Agilent Technologies Japan, Ltd.

Component Test PGU-Kobe

1-3-2, Murotani, Nishi-ku, Kobe, Hyogo, 651-2241 Japan

Sample Program

The customer shall have the personal, non-transferable rights to use, copy, or modify SAMPLE PROGRAMS in this manual for the customer’s internal operations. The customer shall use the SAMPLE PROGRAMS solely and exclusively for their own purposes and shall not license, lease, market, or distribute the SAMPLE PROGRAMS or modification of any part thereof.

Agilent Technologies shall not be liable for the quality, performance, or behavior of the SAMPLE PROGRAMS. Agilent Technologies especially disclaims any responsibility for the operation of the SAMPLE PROGRAMS to be uninterrupted or error-free. The SAMPLE PROGRAMS are provided AS IS.

AGILENT TECHNOLOGIES DISCLAIMS ANY IMPLIED WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Agilent Technologies shall not be liable for any infringement of any patent, trademark, copyright, or other proprietary right by the SAMPLE PROGRAMS or their use. Agilent Technologies does not warrant that the SAMPLE PROGRAMS are free from infringements of such rights of third parties. However, Agilent Technologies will not knowingly infringe or deliver software that infringes the patent, trademark, copyright, or other proprietary right of a third party.

Sample Application Programs

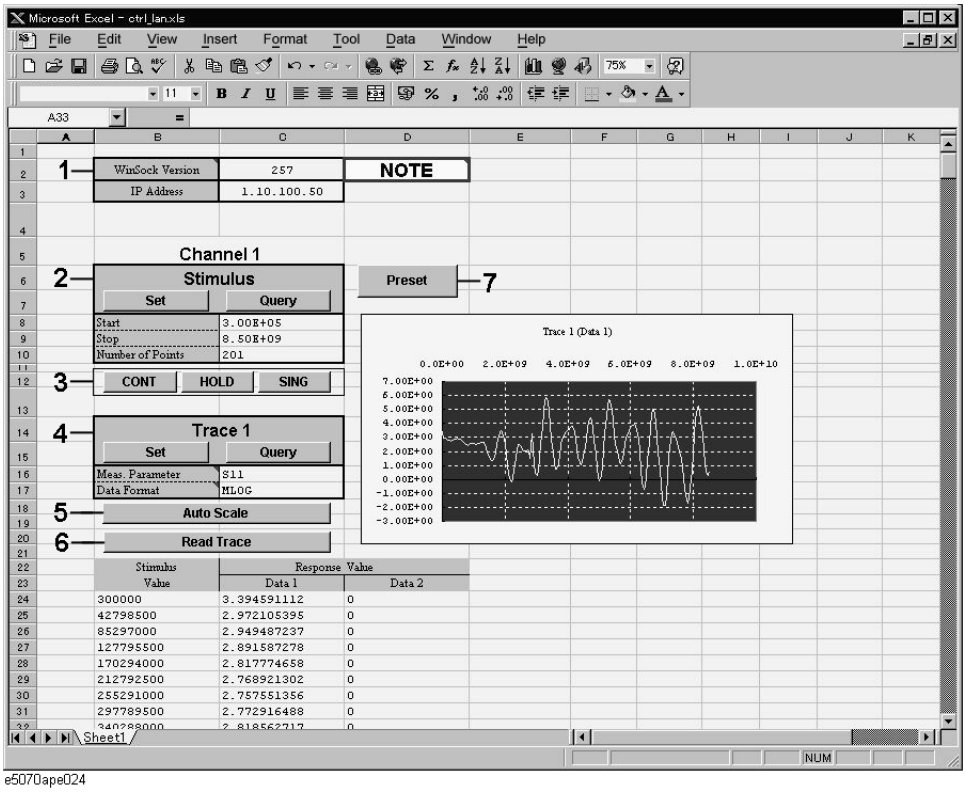
Controlling over LAN

This section describes how to control the E5070B/E5071B using WinSock API in the Windows environment, with a sample program written in Visual Basic (VBA macro). You can find the source file of this program, named ctrl_lan.xls (Microsoft Excel file), on the sample program disk.

Using VBA macro

Opening ctrl_lan.xls in Microsoft Excel display a screen as shown in Figure 1.

Figure 1 **ctrl_lan.xls**



e5070ape024

For how to use each element in Figure 1, refer to the following description.

We begin describing the part 1. Enter the version number of WinSock API in the cell right side of “Winsock Version.” The version number is obtained by multiplying 256 by the major version then adding the minor version. For example, when the version of your Winsock API is 1.1, the version number is obtained as follows: 256×1+1=257. Enter the IP address of the E5070B/E5071B in the cell right side of “IP Address.” This VBA macro will not work properly without appropriate values in the two cells.

In the part 2, sweep range (start and stop points) and number of measurement points are set. Clicking the button labeled as “Set” executes setting operation as specified with the setting table, while clicking the button labeled as “Query” retrieves the current settings of the

E5070B/E5071B.

The part 3 is dedicated to setting the trigger mode.

The part 4 set the measurement parameters and data format for trace 1 in channel 1. Clicking the button labeled as ‘Set” executes setting operation as specified with the setting table, while clicking the button labeled as “Query” retrieves the current settings of the E5070B/E5071B.

In the part 5, clicking the button labeled as “Auto Scale” executes auto scaling for trace 1 in channel 1.

Clicking the button labeled as “Read Trace” in the part 6 retrieves the formatted data of trace 1 in channel 1 and displays it in tabular and graphical formats.

Clicking the button labeled as “Preset” executes the presetting operation.

Description of operation in VBA macro

Here described is operation of the VBA macro, focusing on the part related to controlling with WinSock API.

In order to use WinSock API, you must declare functions and define variables with a definition file of WinSock API, as shown in Example 1.

Example 1 **Definition file of WinSock API**

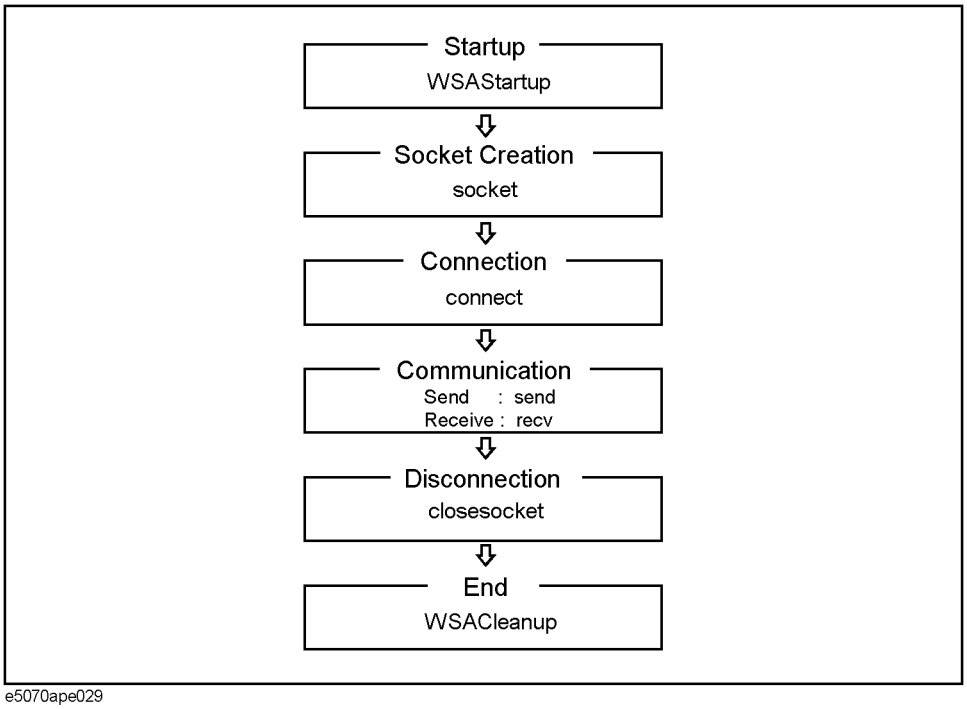
```
'This is the Winsock API definition file for Visual Basic

'Setup the variable type 'hostent' for the WSAStartup command
Type Hostent
  h_name As Long
  h_aliases As Long
  h_addrtype As String * 2
  h_length As String * 2
  h_addr_list As Long
End Type
Public Const SZHOSTENT = 16


'Set the Internet address type to a long integer (32-bit)
Type in_addr
   s_addr As Long
End Type


'A note to those familiar with the C header file for Winsock
'Visual Basic does not permit a user-defined variable type
'to be used as a return structure.  In the case of the
'variable definition below, sin_addr must
'be declared as a long integer rather than the user-defined
'variable type of in_addr.
Type sockaddr_in
   sin_family As Integer
   sin_port As Integer
   sin_addr As Long
   sin_zero As String * 8
End Type

Public Const WSADESCRIPTION_LEN = 256
Public Const WSASYS_STATUS_LEN = 128
Public Const WSA_DescriptionSize = WSADESCRIPTION_LEN + 1
```

```
Public Const WSA_SysStatusSize = WSASYS_STATUS_LEN + 1

'Setup the structure for the information returned from
'the WSAStartup() function.
Type WSAData
   wVersion As Integer
   wHighVersion As Integer
   szDescription As String * WSA_DescriptionSize
   szSystemStatus As String * WSA_SysStatusSize
   iMaxSockets As Integer
   iMaxUdpDg As Integer
   lpVendorInfo As String * 200
End Type

'Define socket return codes
Public Const INVALID_SOCKET = &HFFFF
Public Const SOCKET_ERROR = -1

'Define socket types
Public Const SOCK_STREAM = 1          'Stream socket
Public Const SOCK_DGRAM = 2           'Datagram socket
Public Const SOCK_RAW = 3             'Raw data socket
Public Const SOCK_RDM = 4             'Reliable Delivery socket
Public Const SOCK_SEQPACKET = 5       'Sequenced Packet socket


'Define address families
Public Const AF_UNSPEC = 0            'unspecified
Public Const AF_UNIX = 1              'local to host (pipes, portals)
Public Const AF_INET = 2              'internetwork: UDP, TCP, etc.
Public Const AF_IMPLINK = 3           'arpanet imp addresses
Public Const AF_PUP = 4               'pup protocols: e.g. BSP
Public Const AF_CHAOS = 5             'mit CHAOS protocols
Public Const AF_NS = 6                'XEROX NS protocols
Public Const AF_ISO = 7               'ISO protocols
Public Const AF_OSI = AF_ISO          'OSI is ISO
Public Const AF_ECMA = 8              'european computer manufacturers
Public Const AF_DATAKIT = 9           'datakit protocols
Public Const AF_CCITT = 10            'CCITT protocols, X.25 etc
Public Const AF_SNA = 11              'IBM SNA
Public Const AF_DECnet = 12           'DECnet
Public Const AF_DLI = 13              'Direct data link interface
Public Const AF_LAT = 14              'LAT
Public Const AF_HYLINK = 15           'NSC Hyperchannel
Public Const AF_APPLETALK = 16        'AppleTalk
Public Const AF_NETBIOS = 17          'NetBios-style addresses
Public Const AF_MAX = 18              'Maximum # of address families


'Setup sockaddr data type to store Internet addresses
Type sockaddr
  sa_family As Integer
  sa_data As String * 14
End Type
Public Const SADDRLEN = 16


'Declare Socket functions

Public Declare Function closesocket Lib "wsock32.dll" (ByVal s As Long)
As Long

Public Declare Function connect Lib "wsock32.dll" (ByVal s As Long, addr
As sockaddr_in, ByVal namelen As Long) As Long
```

```
Public Declare Function htons Lib "wsock32.dll" (ByVal hostshort As Long)
As Integer

Public Declare Function inet_addr Lib "wsock32.dll" (ByVal cp As String)
As Long

Public Declare Function recv Lib "wsock32.dll" (ByVal s As Long, ByVal
buf As Any, ByVal buflen As Long, ByVal flags As Long) As Long

Public Declare Function recvB Lib "wsock32.dll" Alias "recv" (ByVal s As
Long, buf As Any, ByVal buflen As Long, ByVal flags As Long) As Long

Public Declare Function send Lib "wsock32.dll" (ByVal s As Long, buf As
Any, ByVal buflen As Long, ByVal flags As Long) As Long

Public Declare Function socket Lib "wsock32.dll" (ByVal af As Long, ByVal
socktype As Long, ByVal protocol As Long) As Long

Public Declare Function WSAStartup Lib "wsock32.dll" (ByVal
wVersionRequired As Long, lpWSAData As WSAData) As Long

Public Declare Function WSACleanup Lib "wsock32.dll" () As Long

Public Declare Function WSAUnhookBlockingHook Lib "wsock32.dll" () As
Long

Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory"
(hpvDest As Any, hpvSource As Any, ByVal cbCopy As Long)
```

Basic control flow with WinSock API is shown in Figure 2.

Figure 2 **Control flow with WinSock API**



e5070ape029

Procedures in each step in Figure 2 are described below.

Startup

The procedure corresponding to Startup is StartIt (Example 2). StartIt launches and initialize WinSock API with **WSAStartup** in WinSock API, whose version is in the part 1 of Figure 1. The function WSAStartup should be always used when initiating WinSock. This function takes version number (input) and launching information (output) as its parameters.

Example 2 **StartIt**

```
Sub StartIt()

    Dim StartUpInfo As WSAData

    'Version 1.1 (1*256 + 1) = 257
    'version 2.0 (2*256 + 0) = 512

    'Get WinSock version
    Sheets("Sheet1").Select
    Range("C2").Select
    version = ActiveCell.FormulaR1C1

    'Initialize Winsock DLL
    x = WSAStartup(version, StartUpInfo)

End Sub
```

Socket Creation and Connection

The procedure for Socket Creation and Connection is OpenSocket (Example 3). OpenSocket makes a connection to an instrument associated with the IP address specified with the input parameter Hostname. It uses a socket of the port specified with the input parameter PortNumber. Each functional part of OpenSocket is described below.

In (1), the inet_aadr function of WinSock API is used to convert an IP address delimited by "." to an Internet address.

In (2), a new socket is created with **socket** function of WinSock API and its socket descriptor is obtained. If an error occurs, the control returns to the main program with a message. socket function takes parameters for an address family (input), a socket type (input), and a protocol number (input).

In (3), the socket address is specified. Note that htons, which is used for specifying the port number, is a function of WinSock API. The function converts a 2-byte integer from the Windows byte order (little endian) to the network byte order (big endian).

In (4), a connection to the E5070B/E5071B is made using **connect** function of WinSock API. If an error occurs, the control returns to the main program with a message. connect function takes parameters for a socket descriptor (input), a socket address (input), and size of the socket address (input).

Example 3 **OpenSocket**

```
Function OpenSocket(ByVal Hostname As String, ByVal PortNumber As Intege
r) As Integer

    Dim I_SocketAddress As sockaddr_in
    Dim ipAddress As Long

    ipAddress = inet_addr(Hostname)                     '...........(1)

    'Create a new socket
    socketId = socket(AF_INET, SOCK_STREAM, 0)          '
    If socketId = SOCKET_ERROR Then                     '
        MsgBox ("ERROR: socket = " + Str$(socketId))    '...........(2)
        OpenSocket = COMMAND_ERROR                      '
        Exit Function                                   '
    End If                                              '

    'Open a connection to a server

    I_SocketAddress.sin_family = AF_INET                '
    I_SocketAddress.sin_port = htons(PortNumber)        '...........(3)
    I_SocketAddress.sin_addr = ipAddress                '
    I_SocketAddress.sin_zero = String$(8, 0)            '

    x = connect(socketId, I_SocketAddress, Len(I_SocketAddress))  '
    If socketId = SOCKET_ERROR Then                               '
        MsgBox ("ERROR: connect = " + Str$(x))                    '..(4)
        OpenSocket = COMMAND_ERROR                                '
        Exit Function                                             '
    End If                                                        '

    OpenSocket = socketId

End Function
```

Communication

The procedure corresponding to Communication is SendCommand (Example 4). SendCommand transmits a message (SCPI command) specified with the input parameter “command” to the E5070B/E5071B using **send** function of WinSock API. send function takes parameters for a socket descriptor (input), a message to be transmitted (input), message length (input) and a flag (input).

Example 4 **SendCommand**

```
Function SendCommand(ByVal command As String) As Integer

    Dim strSend As String

    strSend = command + vbCrLf

    count = send(socketId, ByVal strSend, Len(strSend), 0)

    If count = SOCKET_ERROR Then
        MsgBox ("ERROR: send = " + Str$(count))
        SendCommand = COMMAND_ERROR
        Exit Function
    End If

    SendCommand = NO_ERROR

End Function
```

The procedure corresponding to a receiving part of communication is RecvAscii (Example 5) and other functions. RecvAscii receives a message as ASCII format and stores it in the dataBuf output parameter. Maximum length of the message is specified with the maxLength input parameter. Each functional part of RecvAscii is described below.

In (1), a message (a response to a query for SCPI command) is received from the E5070B/E5071B as a series of characters using **recv** function of WinSock API. If an error occurs, the control returns to the main program with a message. recv function takes parameters for a socket descriptor (input), a message to be received (input), message length (input) and a flag (input).

In (2), it is determined whether each received character is LF (ASCII code: 10). When it is LF, receiving is terminated adding NULL (ASCII code: 0) to the end of dataBuf string and the control returns to the main program.

In (3), number of the last characters that was read out is added to the count value for checking a number of received characters, and append the characters to the end of dataBuf string.

Example 5 **RecvAscii**

```
Function RecvAscii(dataBuf As String, ByVal maxLength As Integer) As
Integer

    Dim c As String * 1
    Dim length As Integer

    dataBuf = ""
    While length < maxLength
        DoEvents
        count = recv(socketId, c, 1, 0)                '
        If count < 1 Then                              '
            RecvAscii = RECV_ERROR                     '............(1)
            dataBuf = Chr$(0)                          '
            Exit Function                              '
        End If                                         '

        If c = Chr$(10) Then                           '
           dataBuf = dataBuf + Chr$(0)                 '............(2)
           RecvAscii = NO_ERROR                        '
           Exit Function                               '
        End If                                         '

        length = length + count                        '............(3)
        dataBuf = dataBuf + c                          '
    Wend

    RecvAscii = RECV_ERROR

End Function
```

Disconnection

The procedure corresponding to Disconnection is CloseConnection (Example 6). CloseConnection disconnects communication and removes a socket using **closesocket** function of WinSock API. closesocket function takes a parameter for a socket descriptor (input).

Example 6 **CloseConnection**

```
Sub CloseConnection()

    x = closesocket(socketId)

    If x = SOCKET_ERROR Then
        MsgBox ("ERROR: closesocket = " + Str$(x))
        Exit Sub
    End If

End Sub
```

End

The procedure corresponding to End is EndIt (Example 7). EndIt disconnects WinSock API using **WSACleanup** function of WinSock API. The function WSACleanup should be always used when terminating WinSock.

Example 7 **EndIt**

```
Sub EndIt()

    'Shutdown Winsock DLL
    x = WSACleanup()

End Sub
```

Example of control

The E5070B/E5071B can be controlled by executing the above procedures in order, following the control flow in Figure 2. This is demonstrated by the procedure autoscale (a procedure which is executed when the Auto Scale button is clicked) as described in Example 8.

Example 8 **autoscale**

```
Sub autoscale()
'
' auto scaling
'
    Call StartIt
    Call get_hostname
    x = OpenSocket(Hostname$, ScpiPort)

    x = SendCommand(":DISP:WIND1:TRAC1:Y:AUTO")

    Call CloseConnection
    Call EndIt

End Sub
```

NOTE

When you execute more than one command by connecting and disconnecting a socket for every command, the sequence of execution may change.

- ❏ Connection → Command 1 → Command 2 → Disconnection

  Commands 1 and 2 are always executed in this sequence.

- ❏ Connection → Command 1 → Disconnection → Connection → Command 2 → Disconnection

  These commands may be in the sequence of Command 2 → command 1.