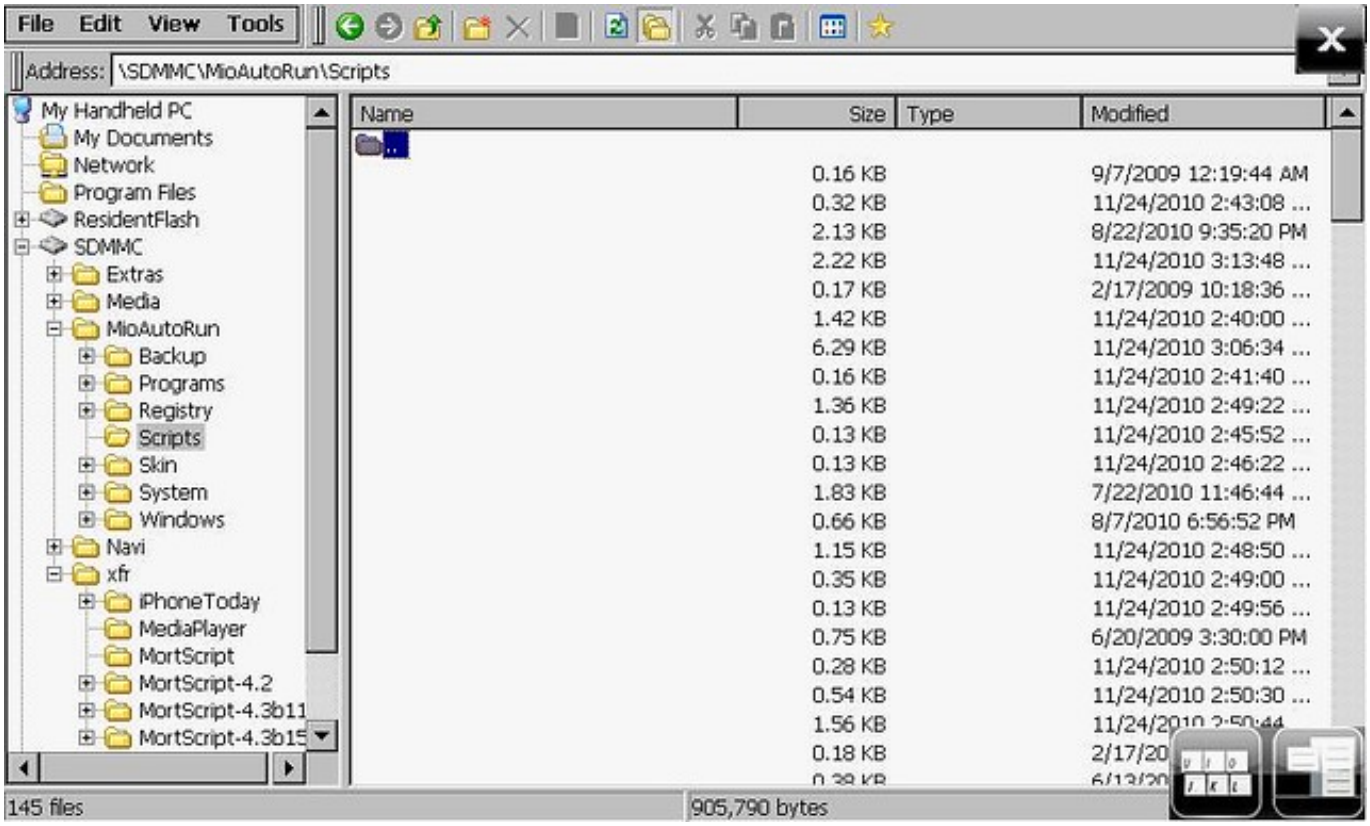
GPSScan Debug.exe



-d "net2.net1.net1.0.port=12345-7" getpeername debug.exe -d "net2.net1.net1.0.port=12345-7" These return the same values for both. So in summary, if I do not call getpeername explicitly, the server port does not change, but the client port does. The application is on.NET 4.0 A: It's a subtle difference that is actually present from.NET 3.5 onwards. If you start by calling GetTcpListenSocket before calling GetTcpClientSocket (GetTcpListenSocket waits until the socket is actually called with GetTcpClientSocket), you get the correct behavior (where the same port is used both on the client and on the server side): Socket listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp); listener.Bind(new IPEndPoint(IPAddress.Any, 12345)); listener.Listen(5); Socket client = listener.Accept(); Debug.Assert(client.LocalEndPoint.Port == 12345); listener.Close(); client.Close(); On the other hand, if you start by calling GetTcpClientSocket first (i.e. on a separate thread) before calling GetTcpListenSocket (GetTcpClientSocket returns immediately and GetTcpListenSocket is called later, on the same thread), then the same port is used on the server side as well as on the client side: In this example, the server port has the value 12345 on both client and server side (instead of 12345-7) because in the second example GetTcpListenSocket (that starts the thread that called GetTcpClientSocket) is not yet called when GetTcpClientSocket is called. If you call the Socket.Accept() method after GetTcpListenSocket, then 82157476af