Application Note

Anritsu envision : ensure

Programming the MT8870A using the .NET Driver

Universal Wireless Test Set MT8870A

Introduction

This application note explains how to program the Universal Wireless Test Set MT8870A using the .NET driver to measure an 802.11ac packet.
The details provided explain:
• Which reference DLLs to include in your .NET project
• How to discover and connect to an MT8870A on a network
• How to capture and measure an 802.11ac packet

.NET References

To use the .NET driver, you must have CombiView and NI VISA 5.0.2 or above installed with .NET 4.0 language support.

The following assemblies must be referenced from the "C:\Program Files\Anritsu\CombiView directory":
• MT8870x.Discovery.dll
• MT8870x.Driver.dll
• CombiView.Common.dll
• VXI_11_DiscoveryLib.dll

The following assemblies must be referenced from the "C:\Program Files\Anritsu\CombiView\Applets directory":
• Mt8870x.Driver.Srw.dll
• SRWireless.Common.dll

Note: For Win7 64 bit systems the Program Files folder is Program Files (x86)

The following assembly must also be referenced. Add this from the available references within Visual Studio after NI VISA has been installed:
• NationalInstruments.VisaNS

The following namespaces must also be added to the source file.

```
using Anritsu.MMD.Instrument;
using Anritsu.MMD.Mt8870x.Discovery;
using Anritsu.MMD.Mt8870x.Driver.Types;
using Anritsu.MMD.Mt8870x.Driver.Srw;
using Anritsu.MMD.SRWireless.Common;
using VXI_11_DiscoveryLib;
using NationalInstruments.VisaNS;
using System.Threading;
```

Connecting to an MU887000A/01A Module

If you know the GPIB address, IP address, or hostname of the MU887000A/01A module, you can connect to it using the VISA resource string. For example, for an MU887000A/01A module with an IP address of 192.168.1.3 and a raw socket port number of 56001, the VISA resource string would be "TCPIP::192.168.1.3::56001::SOCKET". The following code sample shows how to make a connection to this module from a host PC:

```
IMessageBasedSession session =
                ResourceManager.GetLocalManager().Open(
                "TCPIP0::192.168.1.3::56001::SOCKET")
                as MessageBasedSession;
Mt8870xSrw instrument = Mt8870xSrw.Create(session);
```

To connect to an MU887000A/01A module via GPIB, use a GPIB VISA resource string such as "GPIBx::y::INSTR", where "x" is the interface number and "y" is the GPIB address. For example, "GPIB0::4::INSTR".

Discovering MU887000A/01A Modules

If multiple MU887000A/01A modules are connected on an intranet, each with DHCP enabled, or to a host PC via a GPIB interface, you can scan for all modules and connect to the required one as identified by its serial number or physical slot number in the MT8870A mainframe. Follow the steps below to scan and connect to a MU887000A/01A module:

1. Find network cards available on the host PC.

```
//find network card
InterfaceInfo[] networkCards =
   VXI_11_Discovery.GetInterfaceInfo();
```

2. Create an InstrumentDiscovery object.

```
//Create InstrumentDiscovery object
InstrumentDiscovery instrumentDiscovery =
   InstrumentDiscovery.Create();
```

3. Scan for MU887000A/01A modules using the selected network card.

```
//VXI11 auto discovery using the first available network card
Identity[] identities = instrumentDiscovery.Discover(
                      "MT8870.", networkCards[0]).ToArray();
```

4. If multiple modules are available, identify the module you wish to use.
The InstrumentDiscovery scans through Ethernet then GPIB interfaces, and returns a collection of identities that include MT8870A and MU887000A/01A modules.

If an identity is a MT8870A, its Parent property returns null, and its Children property contains a collection of all the modules inserted in the mainframe. This identity can be cast into an MT8870xIdentity object to find detailed information about the mainframe such as the software options and licensing information.

If an identity is a module, its Parent property is set to the mainframe identity and its Children property is null. This identity object can then be cast into an MU887000xIdentity object to find detailed information about this module such as mainframe slot number, waveform file list, and FPGA version numbers.

This additional information is useful in identifying the correct module or mainframe.

In the following example, the model type and serial number are used to identify a particular module.

```
foreach (Identity instrumentFound in identities)
{
    if (instrumentFound.Model == "MU887000A")
    {
        if (instrumentFound.Serial == "123456789")
        {
            //Found the desired MU887000A module
            //add code here to connect to this module (step 5)
        }
    }
}
```

Likewise, you can use the Parent property and the MainframeSlotNumber to identify a module in a particular mainframe.

5. Get IMessageBasedSession and connect to the desired module.

```
IMessageBasedSession session =
   instrumentDiscovery.GetSession(instrumentFound);

// Create driver
Mt8870xSrw mt8870xSrw = Mt8870xSrw.Create(session);
```

General Settings

Make the following settings after connecting to an MU887000A/01A module:
• Select SCPI as the remote language mode.
• Select the short range wireless application (SRW).
• Set the input & output ports.

```
// Switch to SCPI mode
mt8870xSrw.System.Language =
   Anritsu.MMD.Mt8870x.Driver.Types.RemoteLanguageMode.SCPI;

// Select SRW application
mt8870xSrw.Instrument.Application =
   Anritsu.MMD.Mt8870x.Driver.Types.ApplicationType.
   ShortRangeWireless;

// Set port

Anritsu.MMD.Mt8870x.Driver.Types.PortConfiguration
portConfiguration = mt8870xSrw.Routing.PortConfiguration;
portConfiguration.InputPort = TestPort.PortThree;
portConfiguration.OutputPort = TestPort.PortFour;
mt8870xSrw.Routing.PortConfiguration = portConfiguration;
```

Making Measurements on an 802.11ac Packet

1. Clear segments.

```
//Clear all segments
mt8870xSrw.Config.SRWireless.Segment.Clear();
```

2. Add and configure a segment.

```
//Add a new segment
mt8870xSrw.Config.SRWireless.Segment.Append(
   WirelessStandard.WLan_AC);

//set up frequency, level, MCS, triggers, etc
mt8870xSrw.Config.SRWireless.Frequency = 5290e6;
mt8870xSrw.Config.SRWireless.Power = 10;   //10dBm
mt8870xSrw.Config.SRWireless.Wlan80211ac.McsIndex =
   Wlan80211acMcsIndex.Mcs9;
mt8870xSrw.Config.SRWireless.Trigger = SegmentTrigger. Level;
mt8870xSrw.Config.SRWireless.TriggerLevel = -20;    //-20dB
mt8870xSrw.Config.SRWireless.TriggerDelay = -30e-6; //30us
pre-trigger
mt8870xSrw.Config.SRWireless.Settling = 0;

//set capture duration and the number of packets to analyse
mt8870xSrw.Config.SRWireless.Duration = 1e-3;     //1ms
mt8870xSrw.Config.SRWireless.NumberOfPackets = 2;
```

3. Configure measurements.
To maximize test speed, each measurement result item can be turned on or off individually.
For example, graphical results can be turned off if they are not required. In the following example, power, EVM, and spectral mask pass/fail results are enabled.

```
//select measurements
mt8870xSrw.Config.SRWireless.Select.Wlan.TransmitPower =
true;
mt8870xSrw.Config.SRWireless.Select.Wlan.Evm = true;
mt8870xSrw.Config.SRWireless.Select.Wlan.TransmitSpectrumGrap
hical = false;
mt8870xSrw.Config.SRWireless.Select.Wlan.TransmitSpectrumNume
ric = true;
```

For power and spectral measurements, you must specify the part of the packet on which the measurement will be performed. Up to eight gates can be assigned to different measurements.
The position of the measurement gates can be defined by entering the start positions and the total gate width, or by selecting 'full packet'. It is worth noting that gate 1 type defaults to full packet and cannot be changed. Gates 2 to 7 can be defined by the user. In the following example, Gate 2 is enabled and assigned to the power and spectral measurement.

```
//assign gates
mt8870xSrw.Config.SRWireless.Gates.Gate2.Enabled = true;
mt8870xSrw.Config.SRWireless.Gates.Gate2.GateType =
   SegmentGateType.Packet;
mt8870xSrw.Config.SRWireless.Wlan.TransmitPower.GateAssignmen
ts.AddGate(2);
mt8870xSrw.Config.SRWireless.Wlan.Spectrum.GateAssignments.Ad
dGate(2);
```

4. Start the measurement and wait for completion.

```
//-------------------------------------------
// Make measurements
//-------------------------------------------
mt8870xSrw.Initiate.ShortRangeWireless();

// Wait for complete
while (!mt8870xSrw.MeasurementComplete())
{
    Thread.Sleep(10);
}
```

5. Get measurement results.
The MT8870A provides summary results as well as results for individual packets. Whenever possible, the summary results are averaged across the number of packets measured.

```
//-------------------------------------------
// Get measurement results
//-------------------------------------------
// Get power result
WlanTransmitPowerResult power =
   mt8870xSrw.Fetch.SRWireless.Summary.Wlan.TransmitPower(1);
// Check to see if power measurement is successful
if (power.NumberOfGates >= 1)
{
    // Print Gate1 average and peak power results
    Console.Write("{0:N}       {1:N}       ",
                power.GateResults[0].AveragePower,
                power.GateResults[0].PeakPower);
}
// Get EVM result
WlanOfdmEvmResult evm =
mt8870xSrw.Fetch.SRWireless.Summary.Wlan.Ofdm.EvmNumeric(1);
// print EVM result in db and %
Console.Write("{0:N}       {1:N}",
                evm.EvmAvgdB, evm.EvmAvgPct);
// Get Spectral mask result
WlanTransmitSpectrumResult result =
mt8870xSrw.Fetch.SRWireless.Summary.Wlan.TransmitterSpectrum.
Numeric(1);
// Check to see if spectral measurement is successful
if (result.NumberOfGates > 0)
{
    // Print spectral mask pass/fail status
    Console.WriteLine("        {0}",
(result.GateResults[0].SpectralMaskLimitPass) ? "PASS" :
"FAIL");
}
```

If multiple packets are measured, you can query results for each individual packet.

```
// Get the number of segments
int numberOfSegments = mt8870xSrw.Fetch.SRWireless.
CaptureInfo.NumberOfSegments;

// Get the number of packets in the first segment
int numberOfPackets = mt8870xSrw.Fetch.SRWireless.
CaptureInfo.SegmentInfo[0].NumberOfPackets;

Console.WriteLine("Number of Segments: {0}",
numberOfSegments);
Console.WriteLine("Number of Packets : {0}", numberOfPackets);
// Print the results for each packet in the first segment
for (int packetIndex = 1; packetIndex <= numberOfPackets;
packetIndex++)
{
   WlanOfdmEvmResult result = mt8870xSrw.Fetch.SRWireless.
   Packet.Wlan.Ofdm.EvmNumeric(1, packetIndex);

   // print EVM result in db and %
   Console.Write("{0:N}      {1:N}        {2:N}",
      packetIndex, result.EvmAvgdB, result.EvmAvgPct);
}
```

Note: "CaptureInfo" reports only the number of packets in the capture buffer that are defined by “mt8870xSrw.Config.SRWireless. NumberOf Packets”. The capture duration is not automatically adjusted to match the specified number of packets.

Further Assistance

For further support, contact your Anritsu sales or support centre.