Programming 34970A for high channel count and high throughput with Visual Basic

Overview

This application note describes how to program one or more 34970A instruments to extend the channel count of the 34970A, and to achieve high throughput. Since each 34970A has its own DMM and control circuitry, multiple 34970A's may be used to perform data logging simultaneous (parallel processing) for high channel count at high throughput. With ten 34970A's, 480 channels of measurement can be performed in less than 1 second. The time in seconds for a scan of multiple instruments is approximately t = 300 + 65(n) where n it the number of instruments, t the time in milli seconds. This assumes using a 600MHz PC, delay on, DC volts, resolution of 4 ½ digits and the 34902A module.

This application note discusses how to program the 34970A for scanning, datalogging and monitoring. It includes Microsoft Visual Basic 6.0 code examples and Excel code examples for download. This application note demonstrates how to:

- Control multiple 34970A's with MS Visual Basic 6.0
- Set up a function with range and resolution
- Close or Open a relay
- Monitor the readings of the 34970A
- Set thermocouples to an external temperature reference

Available for download:

- Example code using the Agilent ISDK for Visual Basic 6.0
- Example VBA code for Excel 97 or later

Difficulty

The Visual Basic and Excel programming examples discussed are of intermediate difficulty. They require knowledge of the Visual Basic 6.0 development environment, the use of referenced objects, and instrument I/O. Some knowledge of the 34970A instrument and the intended application is assumed.

Operation

To extend the 34970A relay count and achieve a high throughput, several 34970A's, each with an internal DMM, are run simultaneously for parallel processing. Operating multiple 34970A instruments simultaneously is not possible using the BenchLink Data Logger software. To achieve multiple processing the instruments must be programmed for the functions, initialized for a scan and then the data read into the user's environment or file. This application discusses how to program the 34970A for scanning. There are code examples for Visual Basic and Excel.

Most of the sample code uses the Agilent I/O objects that are part of the Agilent ISDK for Visual Basic. It is used here because it is easy to use in Excel VBA, readily available, and makes visible the instrument commands with a minimum of extraneous text.

Agilent Technologies

Programming the 34970A

To program the 34970A requires a GPIB interface. The examples included use the Agilent I/O Objects in **MS Visual Basic 6.0**. The Agilent I/O Object library will work for the GPIB card or USB-to-GPIB converter from either Agilent or National Instruments. For a minimum of problems, GPIB is recommended for both speed and reliability. All the samples assume GPIB.

The Agilent I/O Objects used in these samples can be loaded to your PC by loading the **IntuiLink for DC Power Supplies** found at www.agilent.com/find/intuilink. Once you find the **IntuiLink** page, select **Software, Firmware and Drivers** on the bottom of the page, then select **IntuiLink for Basic DC Power Supplies.** To use GPIB, you must have a GPIB card installed and working prior to using this software. The Agilent I/O objects are also a part of the **Agilent** ISDK for VB6.0 that is available from the Agilent Development Network at www.agilent.com/find/adn Go to the Agilent Development Network for the complete Instrument Software Development Kit (ISDK).

Connecting

Making a connection requires a reference to **Agilent I/O Manager.** From the **Project, References…** menu in VB select **Agilent I/O Manager** to create the reference for your project. The code to establish a connection looks like this:

```
Dim A_34970A As AgtIOServer
Dim io_mgr As AgilentIOUtilsLib.AgtIOManager

Set io_mgr = New AgtIOManager
Set A_34970A = io_mgr.ConnectToInstrument("GPIB::9")
```

Where `GPIB::9 is the GPIB address 9`

Once the above code is executed the object A_34970A can be used to talk to the instrument. This code demonstrates how to retrieve the instrument error:

```
Dim Reply As String
A_34970A.Output "Syst:Error?"
A_34970A.Enter Reply
```

When using multiple instruments it is useful to have each instrument as part of an array. With the array you can step through the instruments with a For Next loop, or add additional instruments as needed. One way to do this is:

```
Dim MUX_IO(1 To 10) As AgtIOServer
Dim io_mgr As AgilentIOUtilsLib.AgtIOManager

Set io_mgr = New AgtIOManager
Set MUX_IO(1) = io_mgr.ConnectToInstrument("GPIB::9")
Set MUX_IO(2) = io_mgr.ConnectToInstrument("GPIB::10")
```

For large number of instruments a **Collection** of AgtIOServers for multiple 34970A's simplifies adding and removing additional instruments.

Closing a Relay

You can close any relay if the module is not part of a scan list. To clear a Scan List, send the command `Route:Scan (@).` Close one or several relays with this command:

```
MUX_IO(1).Output "Route:Close (@301:305,309)"
```

The syntax 301:305 acts on channels 301-305. Open the relays with the `Route:Open` command. To open all the relays except channel 99 (Bank Switch) use `"Syst:Cpon All".` To be sure all relay operations are complete use this code and throw away the returned value.

```
MUX_IO(1).Output "Rout:Done?"
MUX_IO(1).Enter Reply
```

Using the relays this way is useful for routing signals and controlling the individual relays. For taking measurements you should set up the function or multiple functions, the scan list and then take a reading.

Setting up a Function

When setting up a function that we want to monitor or scan at a later time use the Config command. This command lets you set up several channels at a time. For this application once the channels are set up use the Monitor command to read the channels. The following code sets up channels 01,02,03 in slot 3, range 10000 ohms with an integration time of 10 line cycles (6 digit resolution). Once the Configure command is send, the Sense command can be used to refine the setting. The Sense command is optional. Without the Sense command, the DMM uses default values of the Configure command.

```
cmd = "Configure:Resistance 1E4, (@301,302,303)"
MUX_IO(1).Output cmd
cmd = "Sense:Res:NPLC 10, (@301,302,303)"
MUX_IO(1).Output cmd
```

Scanning and Scan list

A Scan List is a list of channels that are scanned when scanning is initiated. The configure command sets up a Scan List, but only for the functions in the one configure command. Send a Scan List command to include the channels of multiple Configure commands, each with a different list of channels and function. The Scan List must include all the channels you want to scan when a scan is initiated. It may include all or part of the channels that are configured.

The following code example is from the EZ Scan example. It represents a complete but simple scan of two different functions.

```
Dim readings() As Double
Dim result As String
Dim DAQ As AgtIOServer
Dim io_mgr As AgilentIOUtilsLib.AgtIOManager

Set io_mgr = New AgtIOManager
Set DAQ = io_mgr.ConnectToInstrument("GPIB0::9")

With DAQ
    ' Stop any ongoing scan and reset
    .Output "Abort"
    .Output "*RST"

    ' configure for DC and Resistance
    .Output "Conf:volt:DC Auto,(@101,102);:Volt:DC:NPLC 0.02,(@101,102)"
    .Output "Conf:Res Auto,(@106:108);:Res:NPLC 0.02,(@106:108)"

    ' Set the scan list
    .Output "Route:Scan (@101,102,106:108)"

    ' initialize and wait for the scan to complete
    .Output "INIT"
    .Output "*OPC?"
    .Enter result

    ' Get the data from the instrument
    .Output "Fetch?"
    .Enter readings
```

```
End With
```

Monitor

To monitor a single channel in the Scan List first set the channel to monitor, set monitor on, and finally get the reading.

```
Dim reading As Double
Dim cmd As String
cmd = "Route:Monitor (@301)"
With DAQ
   .Output cmd
   .Output "Route:Monitor:State ON"
   .Output "Route:Monitor:Data?"
   .Enter reading
End With
```

Temperature with External Reference

Making temperature measurements with the 34970A is straightforward. Set the temperature function like any other function such as DC volts. The temperature sensor (thermistor, thermocouple, etc.) is connected to the channel on the scan list.

When making thermocouple temperature measurements, it may be desirable to terminate the thermocouples at an external terminal block. The reason might be for easier access, because the external terminal block provides better isothermal characteristics for high accuracy thermocouple measurements, or to provide a quick disconnect through a connector. Under these conditions the thermocouple measurements require an external reference temperature.

The external reference temperature is used to calculate the temperature measurements with thermocouples when the thermocouples are terminated outside the 34901A or 34902A MUX card. To make a measurement under these conditions, the first channel (01) of the first module in the 34970A is connected to a thermistor or thermocouple and the function is set using the configure command. Next the thermocouple function is set with the required scan list. Before setting the external reference, the scan list must be set and it must include the temperature reference channel. Finally the channels with the thermocouple functions are set to external temperature reference. The code with the proper sequence looks like this.

```
Dim readings() As Double
Dim result As String
Dim scanList As String

With DAQ
    ' Stop any ongoing scan
    .Output "Abort"
    ' reset instrument
    .Output "*RST"

    ' configure First channel of first module to thermistor 10k ohm
    ' This will measure the reference Junction temperature
    .Output "Conf:Temp THER,10000,(@101);:Unit:Temp C,(@101)"

    ' configure the channels for the thermocouples
    .Output "Conf:Temp TC,J,(@103);:Unit:Temp C,(@103)"
    .Output "Conf:Temp TC,T,(@102);:Unit:Temp C,(@102)"

    ' configure any other channels
    .Output "Conf:volt:DC 10,(@111:113);:Volt:DC:NPLC 1,(@111:113)"
```

```
        ' set the scan list for all channels including the
        ' reference junction temperature
        scanList = "(@101,102,103,111:113)"
        .Output "Rout:Scan " & scanList

        ' Set the thermocouple channel to the external reference
        ' note only the thermocouples are in the scan list
        scanList = "(@102,103)"
        .Output "Temp:Tran:TC:RJun:Type EXT," & scanList

        ' initialize and wait for the scan to complete
        .Output "INIT"
        .Output "*OPC?"
        .Enter result

        ' Get the data from the instrument
        .Output "Fetch?"
        .Enter readings
    End With
```

Example Visual Basic Code

The .zip file available with this Technical Note contains example software in Visual Basic 6.0. When extracting the examples into the root of the C:\ directory the examples will be placed into the directory:

C:\Program Files\Agilent\IntuiLink\34970A\Examples\VB 60.

Scan_Monitor – Demonstrates how to monitor the 34970A, how to read the modules, and how to read the scan list of channels.

Ext_TempRef – Demonstrates how to program the 34970A using an external reference junction with thermocouple measurements.

MatrixSwitch – A graphical user interface for a Matrix Card that demonstrates how to open and close a relay, how to read the state of a relay and how to read the module type installed in the 34970A.

MUX_Switch –A graphical user interface for the 34901A MUX Card that demonstrates how to open and close a relay, how to read the state of a relay and how to read the module type installed in the 34970A.

EZ_Scan – Demonstrates how to set the instrument to perform a scan.

This directory contains several different programs. All are functionally the same, except they are implemented using different I/O in **Visual Basic 6.0**. The Example programs demonstrate the scan described above in **Programming the 34970A.**

ISDK – uses the General Purpose set of ActiveX control and IO objects installed with the **IntuiLink for Power Supplies** www.agilent.com/find/intuilink . These objects are part of the Agilent Instrument Software Development Kit (ISDK) for VB 6.0 that is available from the **Professional Subscription of ADN** at www.agilent.com/find/adn

NI_488 – Uses the National Instrument NI-488 syntax that comes with the NI GPIB interface device.

SICL – Uses the Agilent SICL syntax that comes with the Agilent GPIB interface device.

VISA – Uses the industry standard VISA syntax that comes with many GPIB interface cards.

VISA-COM – Uses the Agilent VISA objects that are part of the Agilent IO library version M.

Excel –The ISDK Visual Basic 6.0 example in **EZ_Scan** is duplicated in Excel. Load the Excel file 34970A_scanExample.xls in directory:

C:\program files\Agilent\IntuiLink\34970A\Examples\Excel.

To see the example code click on Alt-F11.