Section 9

Speed scanning for increased test throughput

In this section:

Introduction

There are three different multiplex modules available for use with the DAQ6510. This application example demonstrates how each of the multiplexer modules can impact productivity by changing test time. The multiplexer modules all share the same basic code base for switching, scanning, and measuring. Any limits on system speed are the result of the relays in the multiplexer that switch the signals from the device under test (DUT) into the instrument.

The Model 7700 20-Channel Differential Multiplexer Module uses electromechanical relays which have low contact resistance and contribute only a minor offset potential (<1 Ω through the end of life and < 500 nV, respectively). This results in the most accurate readings of the modules but with a 3 ms relay closure time, the slowest scan time in comparison to other options.

The 7703 multiplexer module uses reed relays which have low contact resistance (<1 Ω through the end of life), but a higher contact potential (6 µV max) which contributes more signal offset and slightly less precise readings. The benefit of this module is shorter relay close time (less than 1 ms) which makes it approximately three times faster than the 7700.

The 7710 multiplexer module uses solid-state relays which have the highest contact resistance and contact potential of the three options (<5 Ω and <1 µV, respectively) and are therefore the least precise, however the 7710 has the overall speed advantage with a relay close time of less than 0.5 ms, making it twice as fast as the 7703 and at least six times faster than the 7700.

NOTE

For comprehensive information on relay types, topologies, and the benefits or drawbacks of each, go to the Keithley webpage on the Tektronix website (tek.com/keithley) and search for the Switching Handbook.

Equipment required

- One DAQ6510
- One Model 7700 20-channel differential multiplexer module
- One Model 7710 20-channel differential multiplexer module
- One computer setup for remote communication with the DAQ6510
- One device or component to be tested

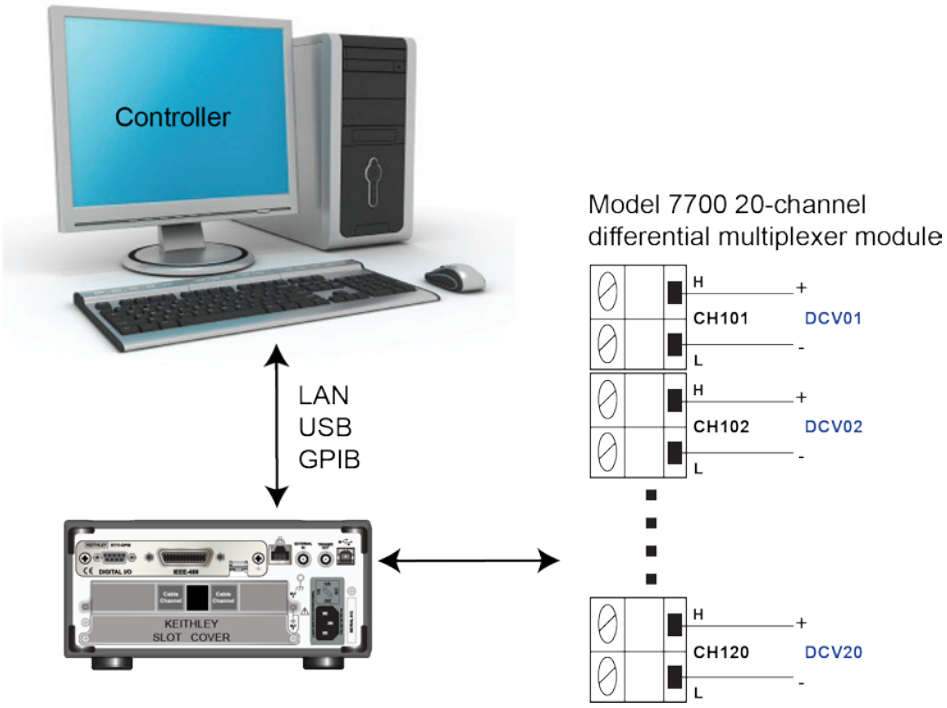
Device connections

This example assumes a setup optimized for the greatest speed where the DAQ6510 uses either the 7700 or 7710 20-channel differential multiplexer module multiple to monitor the following signals; the example code is the same for each.

The controlling computer can use either LAN, USB, or GPIB to reach scan and data transfer speeds comparable to those of the original factory setup. The GPIB interface requires an optional communications accessory.

See the following image for the connection example.

Figure 41: DAQ6510 speed scanning instrument and device connection



NOTE

The 7700 replaces the 7710 for the second run of the program code for comparison and assumes the same input signals to channels 101 to 120.

⚠ WARNING

To prevent electric shock, test connections must be configured such that the user cannot come in contact with test leads or any device under test (DUT) that is in contact with the conductors. It is good practice to disconnect DUTs from the instrument before powering the instrument. Safe installation requires proper shields, barriers, and grounding to prevent contact with test leads.

There is no internal connection between protective earth (safety ground) and the LO terminals of the DAQ6510. Therefore, hazardous voltages (more than 30 V_{rms}) can appear on LO terminals. This can occur when the instrument is operating in any mode. To prevent hazardous voltage from appearing on the LO terminals, connect the LO terminal to protective earth (safety ground) if your application allows it. You can connect the LO terminal to the chassis ground terminal on the front panel or the chassis ground screw terminal on the rear panel. Note that the front-panel terminals are isolated from the rear-panel terminals. Therefore, if you are using the front-panel terminals, ground to the front-panel LO terminal. If using the rear-panel terminals, ground to the rear panel LO terminal. Failure to follow these guidelines can result in injury, death, or instrument damage.

Speed scanning for increased production test throughput

This application demonstrates how to configure the DAQ6510 to execute scans at the best scan speed. You will eliminate certain measurement options that add to overall test time. The scan completion times of the 7700 and 7710 20-channel differential multiplexer modules will be compared, and you will see the speed improvement attributed to solid-state relays over electromechanical relays used in some multiplexer modules.

For this application, you will:

- Use the sample code (either SCPI or TSP) to issue commands that:
  - Fix the DCV measurement range for all channels and remove the delay time introduced through autoranging.
  - Eliminate the autozero feature to remove extra correction measurements from being performed.
  - Set the display digits to a low resolution and turn off the front panel to bypass any delays that would come from updates/refreshes.
  - Turn off channel statistics calculation so that the instrument puts its processing power into data acquisition and transmission.
  - Disable line synchronization.
  - Execute a 20-channel scan with a count of 1000 for a total sample count of 20,000 readings.
  - Incrementally extract the most current scan measurements until complete and either save them to a file or print them to the computer display.
- Evaluate the time elapsed.

Using SCPI commands

This sequence of SCPI commands executes a 20-channel scan 1000 times and saves the data to the controlling computer.

You may need to make changes so that this code will run in your programming environment. In the table, the SCPI commands have a light gray background. The light green shaded code represents pseudocode that will vary depending on the programming environments you use.

Send the following SCPI commands for this example application:

	Commands	Descriptions
Pseudocode	`int scanCnt = 1000`	• Create a variable to hold the scan count
	`int sampleCnt`	• Create a variable to hold the full sample count (total number of readings)
	`int chanCnt`	• Create a variable to hold the channel count
	`int actualRdgs`	• Create a variable to hold the actual reading count
	`string rcvBuffer`	• Create a string buffer to hold extracted readings
	`timer1.start()`	• Start a timer to help capture elapsed time
DAQ6510	`*RST`	• Put the instrument in a known state
	`FORM:DATA ASCII`	• Format data as an ASCII string
	`ROUT:SCAN:COUN:SCAN scanCnt`	• Apply the scan count
	`FUNC 'VOLT:DC',(@101:120)`	• Set function to DCV
	`VOLT:RANG 1, (@101:120)`	• Set the fixed range at 1 V
	`VOLT:AVER:STAT OFF, (@101:120)`	• Disable background statistics
	`DISP:VOLT:DIG 4, (@101:120)`	• Front panel shows only four significant digits
	`VOLT:NPLC 0.0005, (@101:120)`	• Set fastest NPLC possible
	`VOLT:LINE:SYNC OFF, (@101:120)`	• Turn off line sync
	`VOLT:AZER:STAT OFF, (@101:120)`	• Turn off auto zero
	`CALC2:VOLT:LIM1:STAT OFF, (@101:120)`	• Turn off limit tests
	`CALC2:VOLT:LIM2:STAT OFF, (@101:120)`	
	`ROUT:SCAN:INT 0`	• Set trigger interval between scans to 0s
	`TRAC:CLE`	• Clear the reading buffer
	`DISP:LIGH:STAT OFF`	• Turn the display off
	`ROUT:SCAN:CRE (@101:120)`	• Set the scan list
	`chanCnt = ROUTe:SCAN:COUNt:STEP?`	• Query the channel count
Pseudocode	`sampleCnt = scanCnt * chanCnt`	• Calculate the number of readings taken

DAQ6510	`INIT`	• Initiate the scan
Pseudocode	`for i = 1, i < sampleCnt`	• Setup a loop from one up to sampleCnt, but leave the incrementing of i for later
	`delay 500`	• Delay for 500 ms to allow readings to accumulate
DAQ6510	`actualRdgs = TRACe:ACTual?`	• Query the actual readings captured
	`rcvBuffer = "TRACe:DATA? i, actualRdgs, "defbuffer1", READ`	• Query the readings available from i to the value of actualRdgs
Pseudocode	`WriteReadings("C:\myData.csv", rcvBuffer)`	• Write the extracted readings to a file, myData.csv, on the local computer
	`i = actualRdgs + 1`	• Increment i for the next loop pass
	`end for`	• End the for loop
	`timer1.stop()`	• Stop the timer
	`timer1.stop – timer1.start`	• Calculate the elapsed time
DAQ6510	`DISP:LIGH:STAT ON100`	• Turn the display back on

Using TSP commands

NOTE

The following TSP code is designed to be run from Keithley Instruments Test Script Builder (TSB). TSB is a software tool that is available from the Keithley webpage on the Tektronix website (tek.com/keithley). You can install and use TSB to write code and develop scripts for TSP-enabled instruments. Information about how to use TSB is in the online help for TSB and in the "Introduction to TSP operation" section of the *DAQ6510 Reference Manual*.

To use other programming environments, you may need to make changes to the example TSP code.

By default, the DAQ6510 uses the SCPI command set. You must select the TSP command set before sending TSP commands to the instrument.

To enable TSP commands:

1. Press the **MENU** key.
2. Under System, select **Settings**.
3. For Command Set, select **TSP**.
4. At the prompt to reboot, select **Yes**.

This sequence of TSP commands makes a series of voltage measurements. After the code executes, the data is displayed in the Instrument Console of Test Script Builder.

Send the following commands for this example application:

```
-- Set up variables to be referenced during the scan
scanCnt = 1000
sampleCnt = 0
chanCnt = 0
actualRdgs = 0
rcvBuffer = ""
-- Get the initial timestamp for end-of-run comparison
local x = os.clock()
-- Reset the instrument and clear the buffer
reset()
defbuffer1.clear()
-- Set up reading buffer format and establish scan count
format.data = format.ASCII
scan.scancount = scanCnt
-- Configure the scan channels for the Slot 1 card
channel.setdmm("101:120", dmm.ATTR_MEAS_FUNCTION, dmm.FUNC_DC_VOLTAGE)
channel.setdmm("101:120", dmm.ATTR_MEAS_RANGE, 1)
channel.setdmm("101:120", dmm.ATTR_MEAS_RANGE_AUTO, dmm.OFF)
channel.setdmm("101:120", dmm.ATTR_MEAS_AUTO_ZERO, dmm.OFF)
channel.setdmm("101:120", dmm.ATTR_MEAS_DIGITS, dmm.DIGITS_4_5)
channel.setdmm("101:120", dmm.ATTR_MEAS_NPLC, 0.0005)
channel.setdmm("101:120", dmm.ATTR_MEAS_APERTURE, 8.33333e-06)
channel.setdmm("101:120", dmm.ATTR_MEAS_LINE_SYNC, dmm.OFF)
channel.setdmm("101:120", dmm.ATTR_MEAS_LIMIT_ENABLE_1, dmm.OFF)
channel.setdmm("101:120", dmm.ATTR_MEAS_LIMIT_ENABLE_2, dmm.OFF)
-- Dim the display...
display.lightstate = display.STATE_LCD_OFF
-- Generate the scan...
scan.create("101:120")
scan.scaninterval = 0.0
```

```
chanCnt = scan.stepcount
-- Calculate the overall sample count and use it to size the buffer
sampleCnt = scanCnt * chanCnt
defbuffer1.capacity = sampleCnt
-- Start the scan...
trigger.model.initiate()
-- Loop to capture and print readings
i = 1
while i  <= sampleCnt do
   delay(0.5)
   myCnt = defbuffer1.n
   -- NOTE: Can be supplemented or replaced by writing to USB
   printbuffer(i, myCnt, defbuffer1.readings)
   i = myCnt + 1
end
-- Turn the display back on...
display.lightstate = display.STATE_LCD_50
-- Output the elapsed time to the user
print(string.format("Elapsed Time: %2f\n", os.clock() - x))
```

Test results

Factory run times are as follows. The test time for an equivalent setup using the Model 7703 has also been provided so that you can see what is achievable with the reed relays.

Differential Multiplexer Module Setup	Outcome
7710: 20-channels for 1000 scans at 20,000 readings	Approximate test duration: 19.77 s at 1052 rdgs/s
7703: 20-channels for 1000 scans at 20,000 readings	Approximate test duration: 43.12 s at 465 rdgs/s
7700: 20-channels for 1000 scans at 20,000 readings	Approximate test duration: 3 m 38.93 s at 91 rdgs/s

The 7710 multiplexer module provides readings to the DAQ6510 faster and has the speed advantage.