Section 7

Sampling temperature at a set time interval

In this section:

Introduction

This application example demonstrates how to use the Model DMM7510 to log temperature measurement data every minute over a 24-hour period.

It can be important for product quality during production or storage to know the ambient temperature of the testing environment. You can use the Model DMM7510 to set up a temperature monitoring system that samples temperatures at a fixed time interval for an extended period.

Equipment required

- One Model DMM7510
- 100-ohm RTD temperature probe (such as Tektronix TP750)
- One computer set up for remote communication with the Model DMM7510

Device connections

Use of a four-wire resistance temperature detector (RTD) reduces the effects of lead resistance on accuracy and provides high stability.

You can use either the front or rear input terminals for this test. Both front-panel and rear-panel connections are safety banana jacks. Note that you must use either the front terminals or the rear terminals — you cannot mix front and rear connections.

Make sure that the front-panel **TERMINALS** switch is set to the terminals you are using. **F** displayed next to the switch indicates that the instrument is reading the front-panel terminals. **R** indicates that the instrument is reading the rear-panel terminals.

Make the following connections:

- Connect the red test leads to the SENSE HI and INPUT HI terminals.
- Connect the black test leads to the SENSE LO and INPUT LO terminals.

The physical connections for the front and rear panels are shown in the following figures.

Figure 27: 4-wire RTD probe front-panel connections

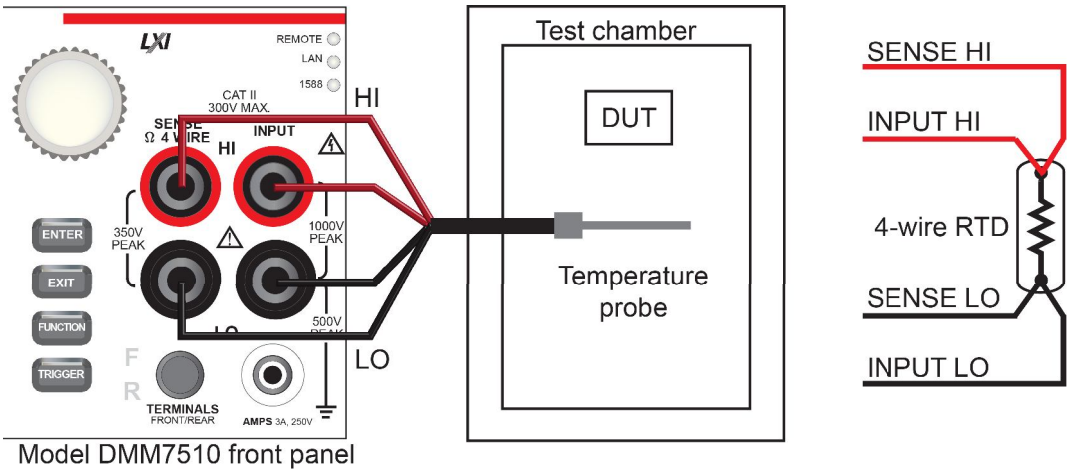
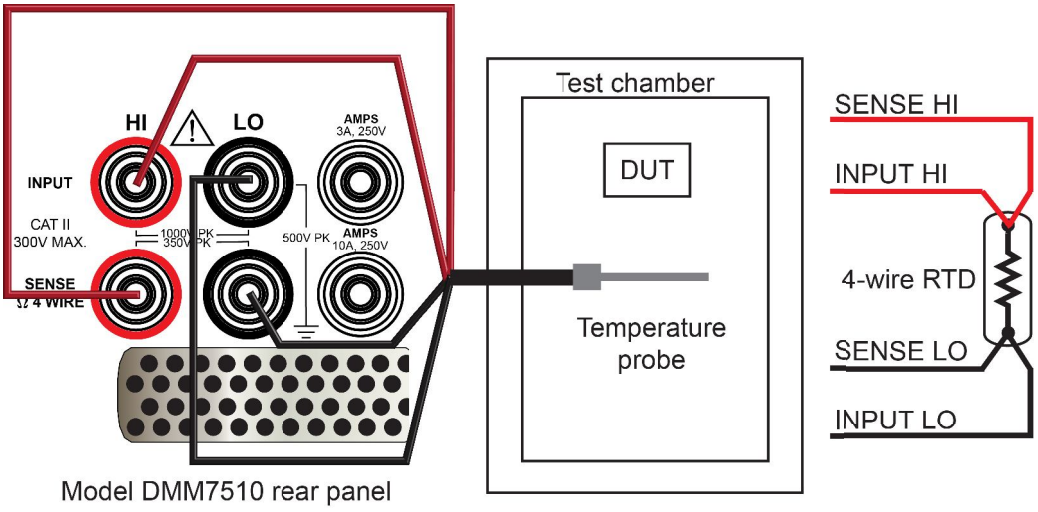


Figure 28: 4-wire RTD probe rear-panel connections



⚠ 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 Model DMM7510. 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 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.

Sample temperatures at a specific time interval

This application demonstrates how to use the Model DMM7510 to measure temperature at fixed time intervals using commands sent over a remote interface. You can make the measurement using SCPI code or TSP code. For information about setting up remote communications, see Remote communications interfaces (on page 3-1).

For this application, you will:

- Reset the instrument.
- Configure the instrument to measure a temperature using a RTD type PT100.
- Enable detection of open leads and offset compensation for more accurate readings.
- Set up a timer to make a temperature measurement every 60 seconds for 1440 times (24 hours) using a trigger model.

Using SCPI commands

This sequence of SCPI commands makes temperature readings every minute for 24 hours.

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-brown shaded code represents pseudocode that will vary depending on the programming environment you use.

Send the following commands for this example application:

Command	Description
`*RST`	• Reset the Model DMM7510
`MeasCount = 1440` `MeasInterval = 60`	• Define variables to make a measurement every 60 seconds for 24 hours (1440 times)

`:SENSe:FUNCtion "TEMPerature"`
`:SENSe:TEMPerature:TRANsducer FRTD`
`:SENSe:TEMPerature:RTD:FOUR PT100`
`:SENSe:TEMPerature:ODETector ON`
`:SENSe:TEMPerature:NPLCycles 1`
`:SENSe:TEMPerature:OCOMpensated ON`
`:DISPlay:SCReen PLOT`
`:TRIGger:TIMer1:DELay MeasInterval`
`:TRIGger:TIMer1:STARt:STIMulus NOTify1`
`:TRIGger:TIMer1:STARt:GENerate OFF`
`:TRIGger:TIMer1:STATe ON`
`:TRIGger:BLOCk:WAIT 1, DISPlay`
`:TRIGger:BLOCk:NOTify 2, 1`
`:TRIGger:BLOCk:MEASure 3, "defbuffer1"`
`:TRIGger:BLOCk:WAIT 4, TIMer1`
`:TRIGger:BLOCk:BRANch:COUNt 5, MeasCont, 2`
`:INIT`
	• Set the instrument to measure temperature
	• Set the transducer type 4-wire RTD
	• Se the RTD to type PT100
	• Enable the detection of open leads
	• Set NPLC to 1
	• Enable offset compensation
	• Switch to the GRAPH swipe screen
	• Set up timer 1 with delay time of `MeasInterval`
	• Set the notify 1 event to trigger the start of the timer
	• Generate a timer event once when the timer delay elapses
	• Enable the timer
	• Set trigger model block 1 to wait for front-panel TRIGGER key press
	• Set block 2 to send NOTIFY1 to start the timer
	• Set block 3 to make a measurement and store it in `defbuffer1`
	• Set block 4 to wait for the timer 1 event after the delay elapses
	• Set block 5 to go to block 2 `MeasCount` times
	• Initiate the trigger model

Using TSP

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 Instruments website. 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 *Model DMM7510 Reference Manual*.

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

By default, the Model DMM7510 is configured to use 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 temperature 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:

```
--Reset the instrument to the default settings
reset()
--Establish variables to make a measurement every 60 seconds 1440 times (24 hours)
MeasCount = 1440
MeasInterval = 60
--Set to measure temperature
dmm.measure.func = dmm.FUNC_TEMPERATURE
--Configure to measure 4-wire RTD
dmm.measure.transducer = dmm.TRANS_FOURRTD
--Enable the detection of open leads
dmm.measure.opendetector = dmm.ON
--Select 4-wire RTD type to be a PT100 sensor
dmm.measure.fourrtd = dmm.RTD_PT100
--Set the number of power line cycles to 1
dmm.measure.nplc = 1
--Enable offset compensation for more accurate readings
dmm.measure.offsetcompensation.enable = dmm.ON
--Display the GRAPH swipe screen
display.changescreen(display.SCREEN_GRAPH_SWIPE)

--Set up timer 1
trigger.timer[1].reset()
--Timer delays each time the timer is triggered
trigger.timer[1].delay = MeasInterval
--Set the notify 1 event to trigger the start of timer
trigger.timer[1].start.stimulus = trigger.EVENT_NOTIFY1
--Generate the timer event once when the timer delay elapses
trigger.timer[1].start.generate = trigger.OFF
--Enable the timer to perform delay operation
trigger.timer[1].enable = trigger.ON
```

```
--Wait for TRIGGER key press from front panel to start trigger model
trigger.model.setblock(1, trigger.BLOCK_WAIT, trigger.EVENT_DISPLAY)
--Send notify to start the timer
trigger.model.setblock(2, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY1)
--Make a measurement and store the reading in default buffer
trigger.model.setblock(3, trigger.BLOCK_MEASURE, defbuffer1)
--Wait for timer delay to elapse
trigger.model.setblock(4, trigger.BLOCK_WAIT, trigger.EVENT_TIMER1)
--Go to block 2 for the number of times set by MeasCount
trigger.model.setblock(5, trigger.BLOCK_BRANCH_COUNTER, MeasCount, 2)

--Initiate trigger model and wait until finished
trigger.model.initiate()
display.changescreen(display.SCREEN_USER_SWIPE)
display.clear()
--Display a message on the USER swipe screen to indicate test in progress
display.settext(display.TEXT1, "Test in progress...")
waitcomplete()
--Display a message on the user swipe screen to indicate test completion
display.clear()
display.settext(display.TEXT1, "Test complete.")
--Print the temperature readings and the corresponding timestamps
printbuffer(1,defbuffer1.n,defbuffer1)
printbuffer(1,defbuffer1.n,defbuffer1.relativetimestamps)
```

Test results

The following figures show a sample graph and final test measurement for this application.

Figure 29: Model DMM7510 graph of temperature measurements

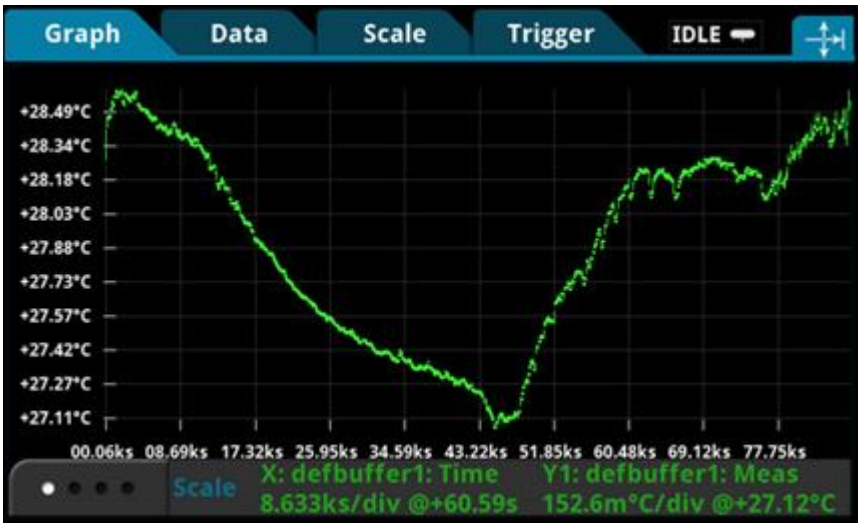


Figure 30: Model DMM7510 final temperature measurement