Writing Test Code

Complete the following steps to use Semiconductor Test Library in your test code:

1. Create a new TSMSessionManager object and any other local variables that are necessary for your test.
2. Use the TSMSessionManager object to query the session for the target pins.
3. Configure the instrumentation connected to the target pins and configure the relay modules.
4. Source and/or measure the signals.
5. Burst the patterns required to configure the DUT.
6. Calculate and/or publish the required test results.
7. Repeat step 4 and 6 as necessary for your test.
8. Clean up and restore the instrument state(s) after finishing the test.

Example C# Code Snippet of Workflow

```
public static void WorkFlowExample(
    ISemiconductorModuleContext semiconductorModuleContext,
    string[] sumPinNames,
    string[] digitalPinNames,
    string relayConfigBeforeTest,
    string relayConfigAfterTest,
    string patternName)
{
    var sessionManager = new TSMSessionManager(semiconductorModuleContext);
    double voltageLevel = 3.3;
    double currentLimit = 0.01;
    double settlingTime = 0.001;
    var measureSettings = new DCPowerMeasureSettings()
    {
        ApertureTime = 0.001,
        Sense = DCPowerMeasurementSense.Remote
    };

    var smuPins = sessionManager.DCPower(sumPinNames);
    var digitalPins = sessionManager.Digital(digitalPinNames);

    smuPins.ConfigureMeasureSettings(measureSettings);
    semiconductorModuleContext.ApplyRelayConfiguration(relayConfigBeforeTest,
waitSeconds: settlingTime);

    smuPins.ForceVoltage(voltageLevel, currentLimit);
    PreciseWait(timeInSeconds: settlingTime);
    var currentBefore = smuPins.MeasureAndPublishCurrent(publishedDataId: "CurrentBefore");
    digitalPins.BurstPatternAndPublishResults(patternName, publishedDataId:
"PatternResults");
    PreciseWait(timeInSeconds: settlingTime);
    var currentAfter = smuPins.MeasureAndPublishCurrent(publishedDataId: "CurrentAfter");

    var currentDifference = currentBefore.Subtract(currentAfter).Abs();
    semiconductorModuleContext.PublishResults(currentDifference,
publishedDataId: "CurrentDifference");

    smuPins.ForceVoltage(voltageLevel: 0, currentLimit: 0.001);
    PreciseWait(timeInSeconds: settlingTime);
    semiconductorModuleContext.ApplyRelayConfiguration(relayConfigAfterTest,
waitSeconds: settlingTime);
}
```

Namespace NationalInstruments.SemiconductorTestLibrary.Common

Classes

NISemiconductorTestException

Defines a specific exception for NI Semiconductor Test Library.

ParallelExecution

Defines utility methods for handling parallel execution.

Publish

Defines measurement results publish methods.

SitePinInfo

Defines an object that contains the information for an individual site-pin pair. Such as, the site number, pin name, instrument channel string used by the driver, instrument model, etc.

SystemSettings

Defines system settings.

Utilities

Provides general helper methods.

Enums

MeasurementType

Defines whether to measure voltage or current.

Namespace NationalInstruments.SemiconductorTestLibrary.TestStandSteps

Classes

CommonSteps

s

SetupAndCleanupSteps

Defines entry points for semiconductor setup and cleanup steps.

Structs

DMMMeasurementSettings

Defines DMM measurement settings.

Enums

SetupAndCleanupSteps.NIInstrumentType

Defines NI instrument types the NI Semiconductor Test Library supports.