Tektronix®

# 使用JEDEC標準JEP183A對 SiC MOSFET進行閾值電壓測試

應 用 摘 要


KEITHLEY
A Tektronix Company

## 介紹

寬能隙裝置在電力電子技術中是廣為人知的裝置。碳化矽 (SiC) 是一種發展潛力強大的材料，與矽基裝置相比，其在增益效率和功率密度方面具有優勢，可提供高電壓和高電流。閾值電壓 ($V_t$) 是功率 MOSFET ( 金屬氧化物半導體場效電晶體 ) 的關鍵裝置特性參數。然而，使用 SiC MOSFET 進行閾值電壓量測存在著可靠性問題，閾值電壓可能會根據先前的閘極偏壓而出現變化。閘極向上掃描與閘極反向掃描的比較圖中顯示了遲滯行為。這種閾值電壓變化也會影響其他裝置特性，例如漏電流和導通電阻 ($R_{dsOn}$)。JEDEC 標準 JEP183A 引入了量測 N 通道垂直結構 SiC MOSFET 裝置閾值電壓 ($V_t$) 的準則。

本應用摘要提供了 SiC MOSFET 的 SiC 裝置遲滯詳細資料，並提出如何根據 JEDEC 標準對閾值電壓進行可靠的量測。Keithley 提供多種 SourceMeter® 電源量測設備 (SMU) 來分析功率 MOSFET 裝置特性，包括閾值電壓測試。本應用摘要中將會介紹一個Test Script Processor (TSP®) 指令碼的範例，說明如何使用 2636B SourceMeter 測試閾值電壓標準。

Keithley 同時也生產高功率 SourceMeter 設備 (SMU) 以分析功率 MOSFET 的特性。其 2657A 高功率系統 SourceMeter 儀器可輸出高達 3 kV 的電壓，而 2651A 搭配 8010 測試夾具則可提供高達 50 A 的脈衝。Keithley 的 ACS 軟體和 KickStart 軟體可提供自動化的功率 MOSFET的 I-V 特性分析測試設定和執行程序。

## 閾值電壓量測中的 SiC 遲滯

量測閾值電壓 ($V_t$) 的方法之一是掃描閘極電壓並在固定偏壓下量測汲極電流。目前有多種方法可以根據傳輸曲線量測結果 ($V_g$ –$I_d$) 來定義閾值電壓。最常見的情況是，當汲極電流達到一定位準時，閘極電壓就是該電流位準下的閾值電壓。然而，SiC 裝置在向上 ( 負電壓到正電壓 ) 和向下 ( 正電壓到負電壓 ) 方向之間具有不同的特性。如圖 1 所示，在 $V_g$ -$I_d$ 資料中觀察到遲滯現象，並且根據閘極的起始電壓偏壓，兩種情況下的閾值電壓有所不同。

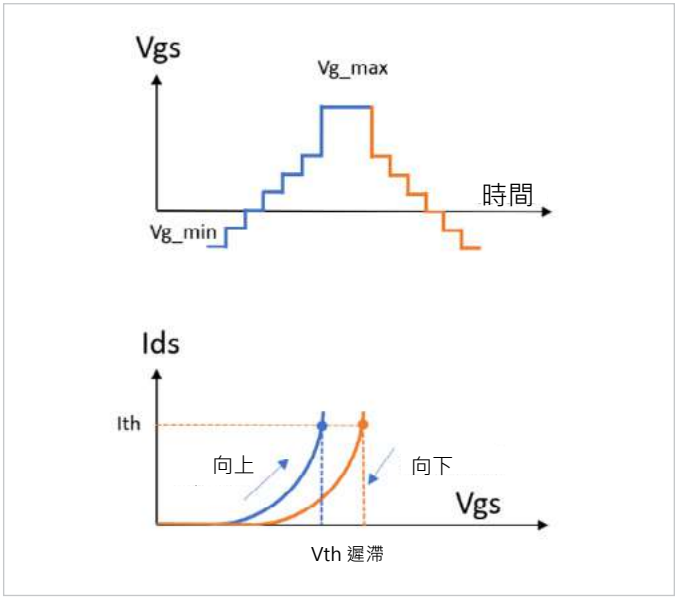



圖 1. SiC MOSFET 閾值電壓遲滯。

## SiC 和閘極氧化物之間的界面

電洞電荷捕獲是向上和向下方向掃描之間閾值電壓遲滯的主要原因。當在閘極終端上施加負電壓時，許多電洞 (SiC 層中帶正電荷的多數載流子 ) 被迫移動至閘極氧化物界面。其中一些會被負閘極偏壓捕獲，如圖 2 左上方所示。當閘極電壓向上掃描時，正閘極偏壓會將一些少數載流子作為電子拉入 SiC 層，使汲極電流如圖所示流動。此外，先前捕獲的電洞電荷會吸引更多電子，如圖 2 左下方所示。眾所周知，這些捕獲的電荷會導致更高的電流。

另一方面，當在閘極終端施加正電壓時，作為少數載流子的電子會移動到界面層，且正閘極偏壓會將先前捕獲的正電荷從氧化矽層推向 SiC 層，如圖 2 右上方所示。當執行向下閘極電壓掃描時，閘極電壓驅動的電子有助於使汲極電流位準低於向上方向的電流。

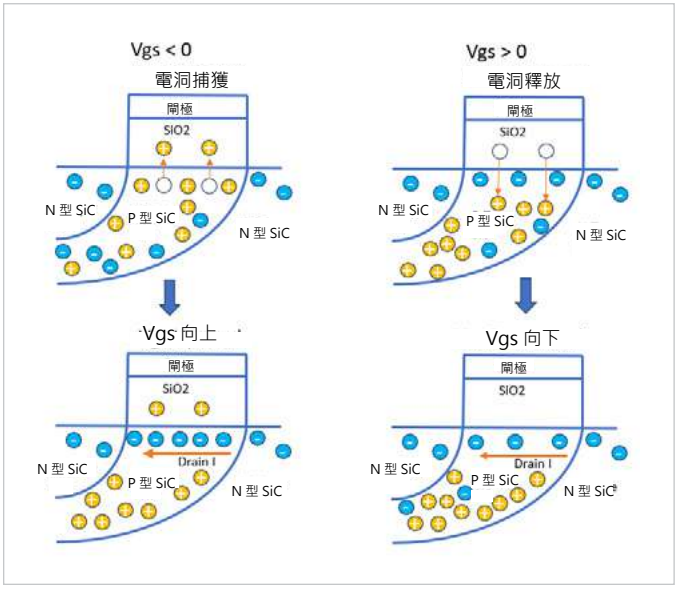



圖 2. 負閘極偏壓捕獲電洞載流子。

## 觀察到的 SiC 遲滯

圖 3 顯示了在示波器時域中，使用市售 SiC MOSFET 裝置搭配 1200 V 32 A 電源進行測試時，汲極電流遲滯與閘極電壓的結果。此測試有兩個步驟。第一步是在閘極上使用負預調節脈衝向上掃描，而第二步則是在閘極上使用正預調節脈衝向下掃描。測試的前半部從 –3V 的閘極電壓脈衝開始，然後增加閘極電壓 ( 黃色 )，同時保持汲極電壓 ( 紫色 ) 固定。汲極電流 ( 綠色 ) 從低位準洩漏開始，並隨著閘極偏壓的增加而上升。後半部則以最大正閘極偏壓 +15 V 開始，然後閘極電壓向下掃描。隨著閘極偏壓縮降低，汲極電流也會從峰值位準降低。即使在時域擷取中也可以觀察到兩個方向之間汲極電流位準的差異。這種差異表示 $V_g$-$I_d$ 曲線和閾值電壓間隙存在遲滯現象。

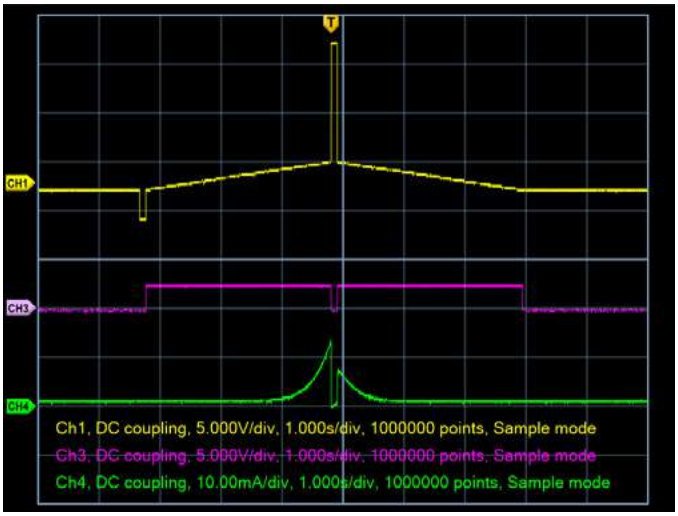
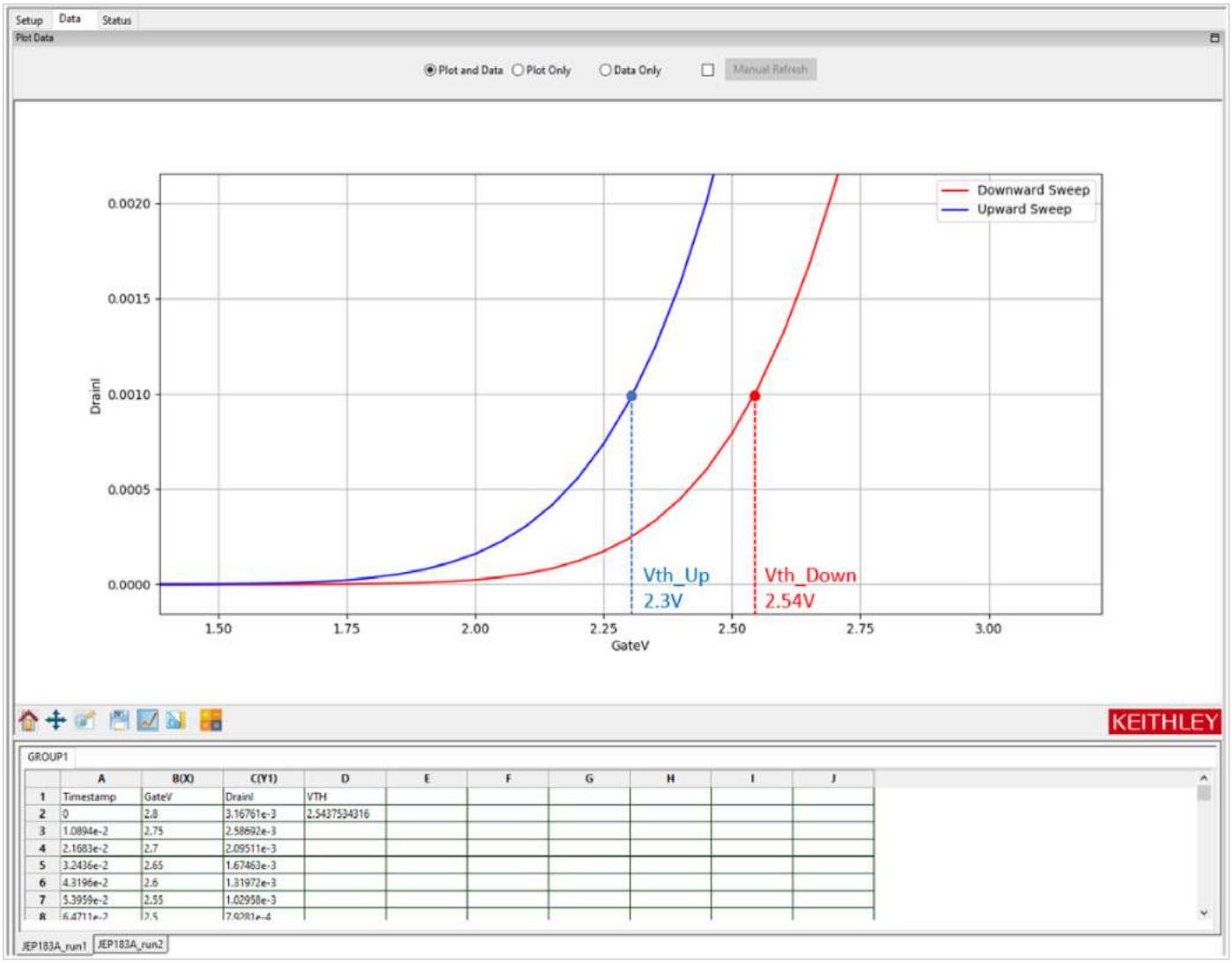



圖 3. 用於掃描方向比較的波形擷取 (CH1 對應 GateV、CH3 對應 DrainV、CH4 對應 DrainI)

圖 4 顯示了 ACS 軟體中雙向掃描閘極偏壓的結果。在汲極電流為 1 mA 時，閘極電壓在兩個方向上皆有 0.24 V 的間隙。此閾值電壓變化可能會影響漏電流和導通電阻 ($R_{dsOn}$)。閾值電壓也可能對可靠性測試中的長期不穩定性產生影響。JEP183A 標準提出了一個新準則來量測閾值電壓的相容性。該準則是在閘極掃描量測之前加入預調節脈衝，以從氧化矽界面釋放任何捕獲的電洞電荷並向下掃描。JEP183A 標準也提出了三種不同的 SiC 閾值量測方法。Keithley SMU 和軟體可以支援這三種方法。Keithley TSP 指令碼可用於任何應用，並且可以嵌入 ACS-Standard 和 ACS-BASIC 軟體中，讓使用者能夠享受 ACS 圖形化介面的優勢，因此無需修改程式碼。TSP 允許我們對 SMU 進行程式設計，以完成預調節脈衝和掃描，而無需複雜的同步程式設計。




| | A | B(X) | C(Y1) | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Timestamp | GateV | DrainI | VTH | | | | | | |
| 2 | 0 | 2.8 | 3.16761e-3 | 2.5437534316 | | | | | | |
| 3 | 1.0894e-2 | 2.75 | 2.58692e-3 | | | | | | | |
| 4 | 2.1683e-2 | 2.7 | 2.09511e-3 | | | | | | | |
| 5 | 3.2436e-2 | 2.65 | 1.67463e-3 | | | | | | | |
| 6 | 4.3196e-2 | 2.6 | 1.31972e-3 | | | | | | | |
| 7 | 5.3959e-2 | 2.55 | 1.02958e-3 | | | | | | | |
| 8 | 6.4711e-2 | 2.5 | 7.9281e-4 | | | | | | | |

圖 4. 觀察到的 SiC MOSFET 閾值電壓遲滯。

## 閘極上的掃描偏壓和固定汲極

第一種方法是將閘極偏壓設定為掃描，並將汲極偏壓設定為固定。測試指令碼嵌入在 ACS 軟體的 STM ( 指令碼測試模組 ) 中，其設定如圖 5 所示。圖 6 中顯示的波形與軟體中程式設計的波形相同。兩個 SMU 通道分別連接到閘極和汲極終端。如果功率 SiC 裝置是 TO-247 型封裝，則 8010 Keithley 高功率裝置測試夾具將是連接至該裝置的最佳選擇。為了符合標準，應在閘極終端上施加預調節脈衝，同時保持汲極和源極終端接地。如圖 7 所示，測試結果應類似典型的傳輸曲線 ($V_g$ –$I_d$)。ACS 軟體的優點是其可根據原始 I-V 資料提供閾值電壓計算。對於本範例中使用的裝置，汲極電流為 1 mA 時閘極電壓為 2.57 V。

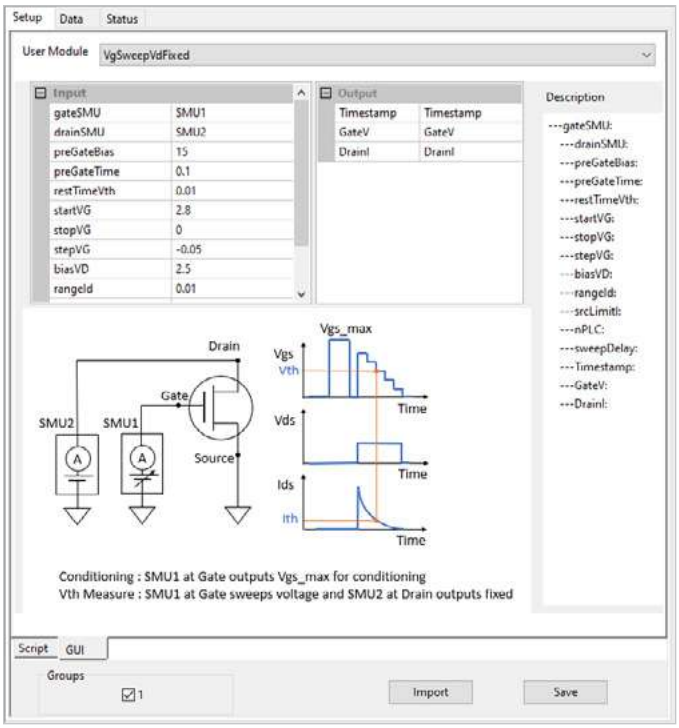



圖 5. 方法一：在固定汲極下掃描閘極。

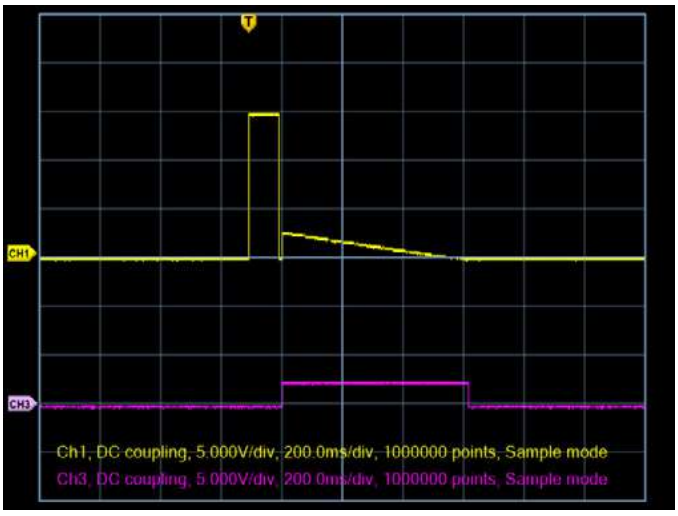



圖 6. 在固定汲極下掃描閘極的波形。

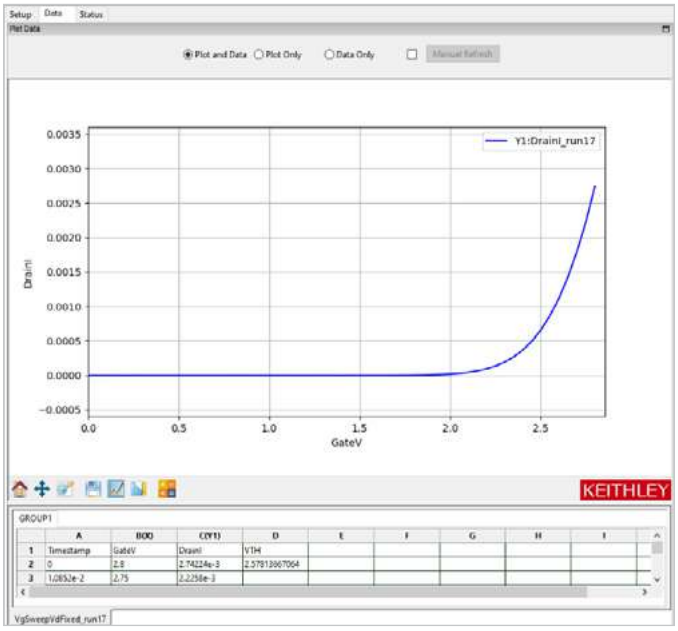



圖 7. 在固定汲極下掃描閘極的結果。

## 閘極和汲極上的掃描偏壓

第二種方法是設定要掃描的閘極和汲極偏壓。為了執行此方法，您可透過下列三個選項來配置量測儀器。

1. 設定單一 SMU 並使用切換矩陣將 SMU 路由至預調節脈衝和掃描所需的終端
2. 設定兩個獨立的 SMU 來掃描閘極和汲極
3. 設定一個 SMU 來執行掃描，並設定第二個 SMU 來執行預調節脈衝

本應用摘要將重點放在選項 2 和選項 3。圖 8 顯示了選項 2 的組態，其中兩個 SMU 分別連接至閘極和汲極終端。圖 9 中的波形與軟體中程式設計的波形相同。根據建議的標準，應在閘極終端上施加預調節脈衝，同時保持其他終端接地。閘極 SMU 和汲極 SMU 應同步工作，以確保閘極和汲極處於相同電位。ACS 軟體負責模型配置。如圖 10 所示，測試結果應類似典型的傳輸曲線 ($V_g$ –$I_d$)。在本例中，汲極電流為 1 mA 時，閘極電壓為 2.57 V。與僅掃描閘極的第一種測試方法相比，此測試中的值可能略有不同，因為條件不相同。然而，此裝置顯然具有與第一種情況相同的閾值電壓。

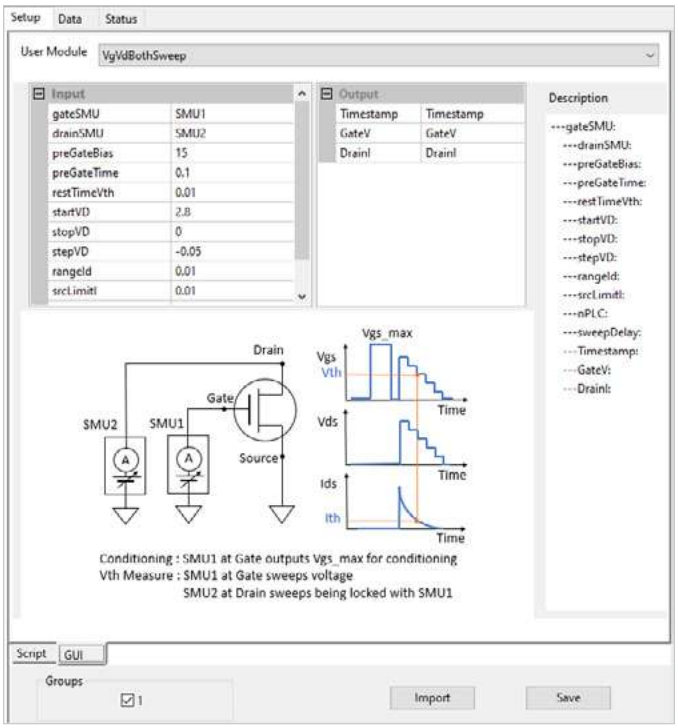



圖 8. 方法二，選項 2：使用單獨的 SMU 掃描閘極和汲極。

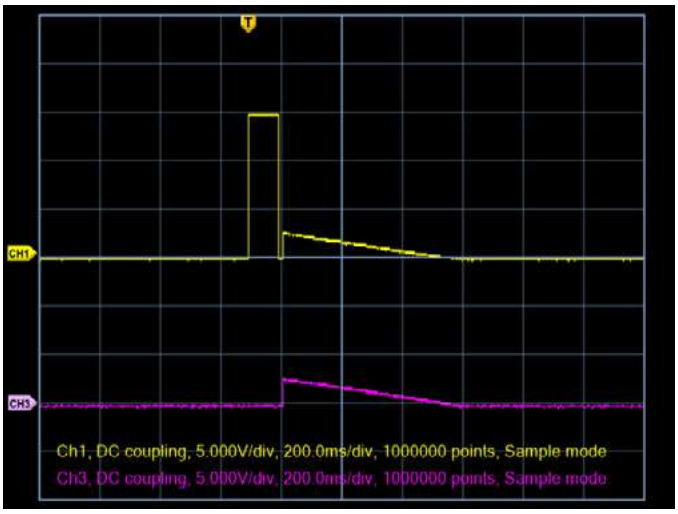



圖 9. 使用單獨的 SMU 掃描閘極和汲極的波形。

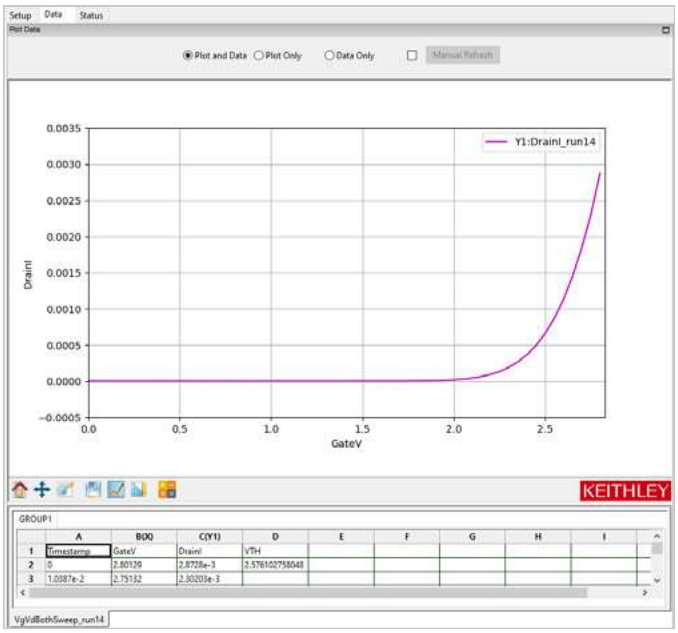



圖 10. 使用單獨的 SMU 掃描閘極和汲極的結果。

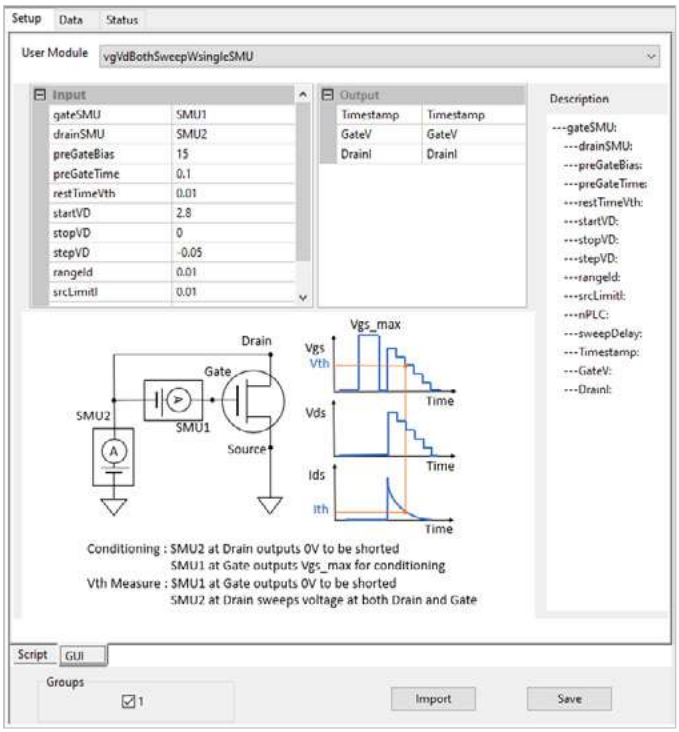



圖 11. 方法二、選項 3：使用相同 SMU 掃描閘極和汲極。

先前的設定要求兩個 SMU 緊密同步才能獲得正確的結果。第一種設定的好處是將相同的 SMU 掃描路由到每個終端，確保終端可接收到相同的輸入。不過，這涉及矩陣控制的額外程式設計和額外昂貴的硬體。而第二種設定則可以修改為僅使用其中一個 SMU 執行掃描，並使用第二個 SMU 執行預調節脈衝，進而降低同步要求。我們可以透過將閘極 SMU 的低端連接至汲極 SMU 的高終端來達成此設定，如圖 11 所示。

在預調節脈衝期間，連接汲極的 SMU 設定為零伏特，作為低端短路。在脈衝後，汲極 SMU 執行掃描量測，且閘極 SMU 必須輸出零電壓偏壓，以建立從汲極 SMU 到閘極終端的短路。則汲極和閘極即會具有與單一汲極 SMU 相同的偏壓。圖 12 顯示了每個 SMU 的實際波形。圖 13 顯示測試結果類似典型的傳輸曲線 ($V_g$–$I_d$)。當汲極電流為 1 mA 時，閘極電壓為 2.57 V。此值與先前設定的閾值電壓完全相同，由兩個單獨的 SMU 執行掃描。

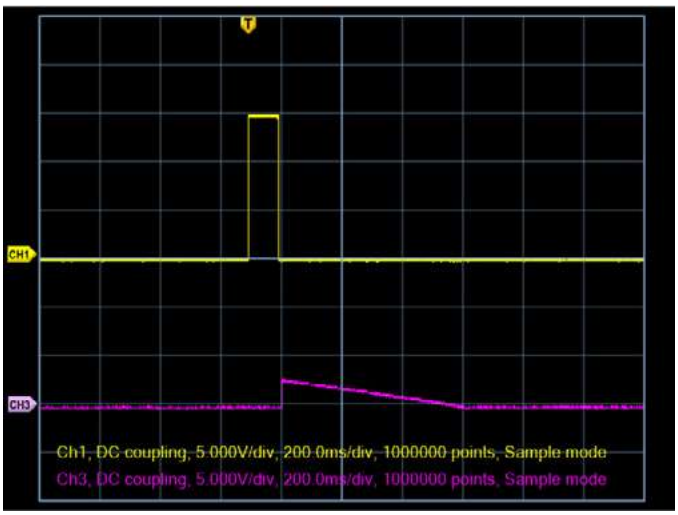



圖 12. 使用相同 SMU 掃描閘極和汲極的波形。

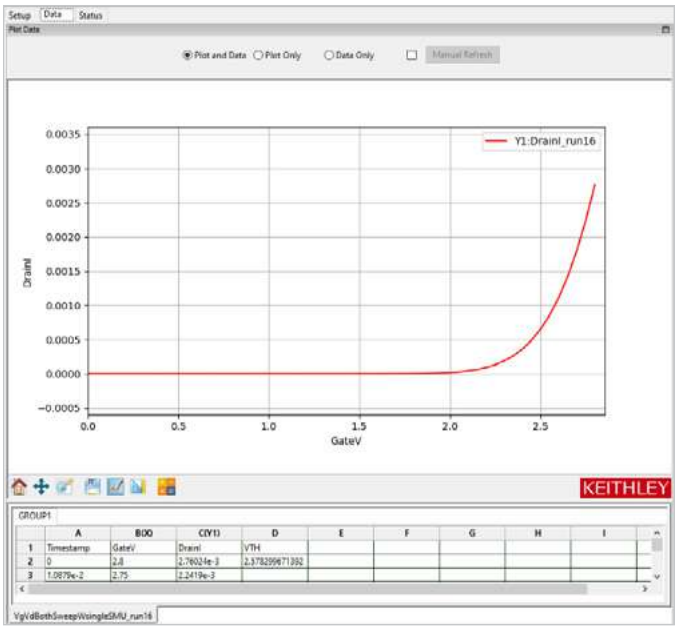



圖 13. 使用相同 SMU 掃描閘極和汲極的結果。

## 閘極和汲極上的固定電流偏壓

JEP183A 標準中的第三種方法是為閘極和汲極設定固定電流偏壓。此方法使用與第二種方法的第三個選項相同的儀器組態。連接兩個 SMU，第一個連接至閘極，第二個連接至汲極，且閘極的 LO 終端連接至汲極的 HI 終端，如圖 14 所示。在預調節脈衝期間，連接汲極的 SMU 設定為零伏特，以建立低端對源極的短路。當汲極 SMU 執行閾值電壓量測時，閘極 SMU 必須輸出零電壓偏壓，以建立從汲極 SMU 到閘極終端的短路。閘極和汲極必須施加相同的固定電流。或者，透過連接外部切換系統，在預調節脈衝期間掀將 SMU 路由到正確的終端，然後再路由到電流偏壓，也可以達到相同的效果。但這會增加程式碼和連接的額外複雜性，這就是為什麼此方法具有優勢的原因。無需外部切換矩陣即可向閘極和汲極提供相同的電流。

圖 15 中顯示的實際波形與軟體中程式設計的波形相同。第三種方法無法提供典型的傳輸曲線 ($V_g$–$I_d$)，而是透過固定汲極電流下的電壓量測直接產生閾值電壓。此範例在 1 mA 汲極電流下會產生 2.57 V 的閾值電壓，這與所有其他測試案例結果一致。

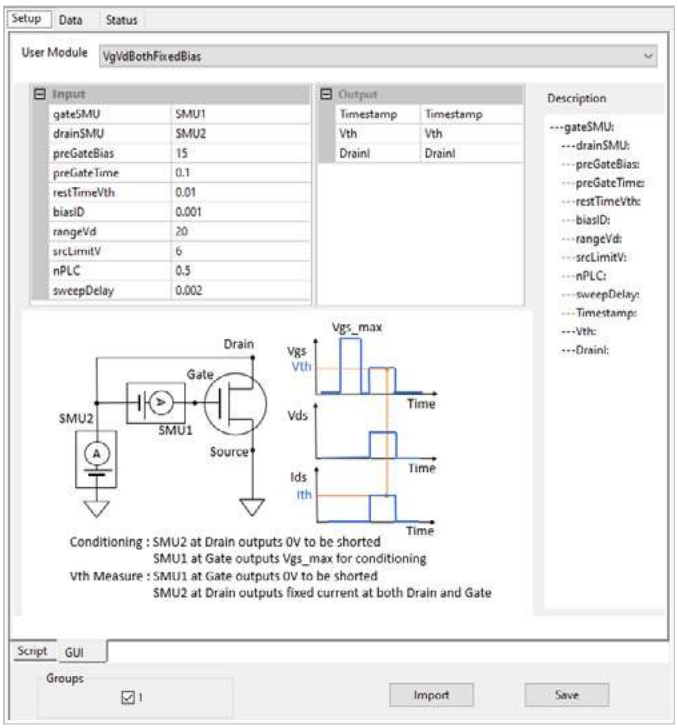



圖 14. 方法三：使用相同 SMU 的閘極和汲極固定電流偏壓。

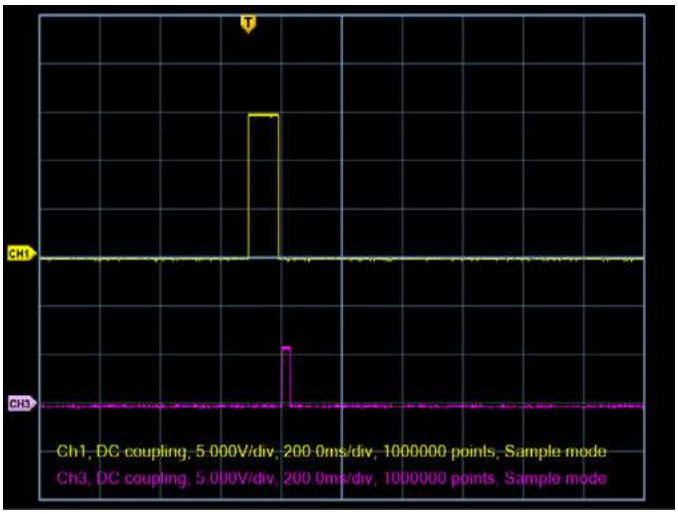



圖 15. 使用相同 SMU 的閘極和汲極固定電流偏壓波形。

## 結論

Keithley 提供了 SourceMeter 儀器和軟體來進行 JEP183A 標準提出的 SiC MOSFET Vt 量測。Keithley TSP 指令碼模組可用於任何應用，並嵌入 ACS 軟體以方便資料視覺化。為了使單一通道 SMU 的閘極和汲極終端短路，需要外部切換系統，或需要額外的 SMU。但一部具有兩個通道的 SMU 即可提供 SiC 閾值電壓解決方案，而無需任何額外的切換或 SMU 儀器。

## 附錄：$V_t$ 測量的 TSP 程式碼

```
--[[
##############################################################################

Script File: SiC_Vth_Measurement.tsp

    ********************************************************************
    *** Copyright Tektronix, Inc.                                    ***
    *** See www.tek.com/sample-license for licensing terms.          ***
    ********************************************************************

Description:
       This script is example code, which creates (and subsequently calls) a
       single function that can be used with the Model 2636B SourceMeter unit
       to make new threshold voltage measurement. There are three different methods
       are introduced as the JEDEC standard proposed. All the measurements include
       conditioning pulse prior to the execution. Upon completion, the data is
       printed to the Test Script Builder Instrument Console in a format that is
       suitable for copying and pasting into Microsoft Excel for graphing and
       analysis.

Required Equipment:  1 Model 2636B System SourceMeter instrument

Note:  The function does not perform any error checking.  It is the user's
       responsibility to specify settings that are compatible with the
       instrument model being used, and with its power envelope.

Note:  It is the user's responsibility to follow all safety guidelines given in
       the instrument's Reference Manual.  This is especially critical if
       voltages in excess of 42VDC will be present in the test circuits.  Such
       voltage levels are hazardous.

Function created by this script:
       For sweeping gate voltage and fixing drain voltage
       * newVTH = newVthSiC_JEP183()
       * newVTH.vgSweepVdFixed(15,100e-3,10e-3,2.8,0, -0.05,2.5,1e-3, 0.1, 0.5, 0.002)

       For sweeping both gate voltage and drain voltage
       * newVTH = newVthSiC_JEP183()
       * newVTH.vgVdBothSweep(15, 100e-3, 10e-3, 2.8, 0, -0.05, 10e-3, 7e-3, 0.1, 0.002)

       For sweeping both gate voltage and drain voltage with single SMU
       * newVTH = newVthSiC_JEP183()
       * newVTH.vgVdBothSweepWsingleSMU(15, 100e-3, 10e-3, 2.8, 0, -0.05, 10e-3, 7e-3, 0.1,0.002)

       For fixing both gate current and drain current
       * newVTH = newVthSiC_JEP183()
       * newVTH.vgVdBothFixedBias(-4, 100e-3, 10e-3, 5e-3, 20, 5, 1, 50e-3)

##############################################################################
--]]
```

```
function newVthSiC_JEP183()
	local self = { }

	local function initSMU()
		local srcRangeI = 0.1
		local srcRangeV = 20
		local srcLimitI = 0.1
		local srcLimitV = 20
		local measRangeI = 0.1
		local measRangeV = 20
		local nPLC = 0.1

		reset()

		setGateAutoSourceRangeI(0)
		setGateAutoSourceRangeV(0)
		setGateAutoMeasureRangeI(0)
		setGateAutoMeasureRangeV(0)
		setGateSourceSettling(128)
		setGateMeasAutoZero(1)
		setGateSrcFunction(1)
		setGateSrcRangeV(srcRangeV)
		setGateSrcLimitI(srcLimitI)
		setGateMeasRangeI(measRangeI)
		setGateSrcRangeI(srcRangeI)
		setGateSrcLimitV(srcLimitV)
		setGateMeasRangeV(measRangeV)
		setGateMeasNPLC(nPLC)

		setDrainAutoSourceRangeI(0)
		setDrainAutoSourceRangeV(0)
		setDrainAutoMeasureRangeI(1)
		setDrainAutoMeasureRangeV(0)
		setDrainSourceSettling(128)
		setDrainMeasAutoZero(1)
		setDrainSrcFunction(1)
		setDrainSrcRangeV(srcRangeV)
		setDrainSrcLimitI(srcLimitI)
		setDrainMeasRangeI(measRangeI)
		setDrainSrcRangeI(srcRangeI)
		setDrainSrcLimitV(srcLimitV)
		setDrainMeasRangeV(measRangeV)
		setDrainMeasNPLC(nPLC)
	end


	function self.vgSweepVdFixed(preGateBias, preGateTime, breakTimeVth,
startVG,stopVG,stepVG,biasVD,rangeId,srcLimitI,nPLC,sweepDelay)
		setDrainMeasRangeI(rangeId)
		drainBuffer = smub.nvbuffer1
		drainBuffer.clear()
		drainBuffer.appendmode = 1
		drainBuffer.collecttimestamps = 1

		local steps
		steps = math.ceil((stopVG - startVG)/stepVG) + 1
		local biasV = {}

		setDrainMeasNPLC(nPLC)
		setDrainSrcLimitI(srcLimitI)
		setDrainSourceDelay(0)
		setDrainMeasDelay(0)

		setGatePowerOnOff(1)
		setDrainPowerOnOff(1)
```

```
		setGateSrcLevelV(preGateBias)
		delay(preGateTime)
		setGateSrcLevelV(0)
		delay(breakTimeVth)

		setDrainSrcLevelV(biasVD)
		for index = 1,steps do
			biasV[index] = startVG + (stepVG*(index-1))
			setGateSrcLevelV(biasV[index])
			delay(sweepDelay)
			smub.measure.i(drainBuffer)
		end

		setGatePowerOnOff(0)
		setDrainPowerOnOff(0)

		print("No\tTime\tGateV\tDrainI")
		for index=1,drainBuffer.n do
			print(index, drainBuffer.timestamps[index],biasV[index],drainBuffer[index])
		end


		initSMU()
	end

	function self.vgVdBothSweep(preGateBias, preGateTime, breakTimeVth, startVD,stopVD,stepVD,
rangeId, srcLimitI,nPLC,sweepDelay)

		setDrainMeasRangeV(rangeId)
		drainIBuffer = smub.nvbuffer1
		drainIBuffer.clear()
		drainIBuffer.appendmode = 1
		drainIBuffer.collecttimestamps = 1

		drainVBuffer = smub.nvbuffer2
		drainVBuffer.clear()
		drainVBuffer.appendmode = 1
		drainVBuffer.collecttimestamps = 1

		local steps
		steps = math.ceil((stopVD - startVD)/stepVD) + 1


		setDrainMeasNPLC(nPLC)
		setDrainSrcLimitI(srcLimitI)
		setDrainSourceDelay(0)
		setDrainMeasDelay(0)

	measureCompleteBlender = trigger.blender[2]
	measureCompleteBlender.clear()
	measureCompleteBlender.orenable  = false
	measureCompleteBlender.stimulus[1]      = smua.trigger.PULSE_COMPLETE_EVENT_ID
	measureCompleteBlender.stimulus[2]      = smub.trigger.PULSE_COMPLETE_EVENT_ID

	sourceBlender = trigger.blender[1]
	sourceBlender.clear()
	sourceBlender.orenable           = true
	sourceBlender.stimulus[1]  = measureCompleteBlender.EVENT_ID
	sourceBlender.stimulus[2]  = trigger.generator[1].EVENT_ID


		sweepDelayTimer = trigger.timer[1]
		sweepDelayTimer.count            = 1
		sweepDelayTimer.delay            = sweepDelay
```

```
		sweepDelayTimer.passthrough       = false
		sweepDelayTimer.stimulus   = smua.trigger.SOURCE_COMPLETE_EVENT_ID


		for i=1, 2 do
			if i == 1 then
				configuringSMU = smua
			else
				configuringSMU = smub
			end

			configuringSMU.trigger.source.linearv(startVD, stopVD, steps)
			configuringSMU.trigger.source.limiti           = srcLimitI
			configuringSMU.trigger.source.action           = configuringSMU.ENABLE
			configuringSMU.trigger.source.stimulus  = sourceBlender.EVENT_ID

			configuringSMU.trigger.measure.action          = configuringSMU.ENABLE
			configuringSMU.trigger.measure.iv(configuringSMU.nvbuffer1, configuringSMU.
nvbuffer2)
			configuringSMU.trigger.measure.stimulus = sweepDelayTimer.EVENT_ID
			configuringSMU.trigger.endpulse.action  = configuringSMU.SOURCE_HOLD
			configuringSMU.trigger.endsweep.action  = configuringSMU.SOURCE_IDLE
			configuringSMU.trigger.endpulse.stimulus       = configuringSMU.trigger.
MEASURE_COMPLETE_EVENT_ID

			configuringSMU.trigger.count                   = steps
		end


		setGatePowerOnOff(1)
		setDrainPowerOnOff(1)

		setGateSrcLevelV(preGateBias)
		delay(preGateTime)
		setGateSrcLevelV(0)
		delay(breakTimeVth)

		smub.trigger.initiate()
		smua.trigger.initiate()
		trigger.generator[1].assert()

		waitcomplete()

		setGatePowerOnOff(0)
		setDrainPowerOnOff(0)

		print("No\tTime\tGateV\tDrainI")
		for index=1,drainIBuffer.n do
			print(index, drainBuffer.
timestamps[index],drainVBuffer[index],drainIBuffer[index])
		end

		initSMU()
	end

	function self.vgVdBothSweepWsingleSMU(preGateBias, preGateTime, breakTimeVth,
startVD,stopVD,stepVD, rangeId, srcLimitI,nPLC,sweepDelay)

		setDrainMeasRangeV(rangeId)
		drainBuffer = smub.nvbuffer1
		drainBuffer.clear()
		drainBuffer.appendmode = 1
		drainBuffer.collecttimestamps = 1

		local steps
```

```
		steps = math.ceil((stopVD - startVD)/stepVD) + 1
		local biasV = {}

		setDrainMeasNPLC(nPLC)
		setDrainSrcLimitI(srcLimitI)
		setDrainSourceDelay(0)
		setDrainMeasDelay(0)

		setGatePowerOnOff(1)
		setDrainPowerOnOff(1)

		setGateSrcLevelV(preGateBias)
		delay(preGateTime)
		setGateSrcLevelV(0)
		delay(breakTimeVth)


		for index=1,steps do
			biasV[index] = startVD + (stepVD*(index-1))
			setDrainSrcLevelV(biasV[index])
			delay(sweepDelay)
			smub.measure.i(drainBuffer)
		end
		setGatePowerOnOff(0)
		setDrainPowerOnOff(0)
		print("No\tTime\tGateV\tDrainI")
		for index=1,drainBuffer.n do
			print(index, drainBuffer.timestamps[index],biasV[index],drainBuffer[index])
		end

		initSMU()
	end


	function self.vgVdBothFixedBias(preGateBias, preGateTime, breakTimeVth, biasId, rangeVd,
srcLimitV, nPLC,measDelay)


		setDrainMeasRangeV(rangeVd)
		drainBuffer = smub.nvbuffer1
		drainBuffer.clear()
		drainBuffer.appendmode = 1
		drainBuffer.collecttimestamps = 1


		setDrainMeasNPLC(nPLC)
		setDrainSrcLimitV(srcLimitV)
		setDrainSourceDelay(0)
		setDrainMeasDelay(0)

		setGatePowerOnOff(1)
		setDrainPowerOnOff(1)

		setGateSrcLevelV(preGateBias)
		delay(preGateTime)
		setGateSrcLevelV(0)
		delay(breakTimeVth)

		setDrainSrcFunction(0)

		setDrainSrcLevelI(biasId)
		delay(measDelay)
		smub.measure.v(drainBuffer)

		setGatePowerOnOff(0)
```

```
		setDrainPowerOnOff(0)

		print("No\tTime\tBiasI\tMeasV")
		for index=1,drainBuffer.n do
			print(drainBuffer.timestamps[index],biasId,drainBuffer[index])
		end

		initSMU()
	end

	function self.vgDualSweepVdFixed(preGateBiasUP, preGateBiasDOWN,preGateTime,
startVG,stopVG,stepVG,biasVD, rangeId, srcLimitI,nPLC,sweepDelay)

		setDrainMeasRangeI(rangeId)
		drainBuffer = smub.nvbuffer1
		drainBuffer.clear()
		drainBuffer.appendmode = 1
		drainBuffer.collecttimestamps = 1

		local steps
		steps = ((stopVG - startVG)/stepVG) + 1
		local biasV = {}

		setDrainMeasNPLC(nPLC)
		setDrainSrcLimitI(srcLimitI)
		setDrainSourceDelay(0)
		setDrainMeasDelay(0)

		setGatePowerOnOff(1)
		setDrainPowerOnOff(1)
		-- sweep for upward
		setGateSrcLevelV(preGateBiasUP)
		delay(preGateTime)

		setDrainSrcLevelV(biasVD)
		local biasVg
		for index=1,steps+1 do
			biasVg =  startVG + (stepVG*(index-1))
			setGateSrcLevelV(biasVg)
			table.insert(biasV,biasVg)
			delay(sweepDelay)
			smub.measure.i(drainBuffer)
		end
		setDrainSrcLevelV(0)

		-- sweep for downward
		setGateSrcLevelV(preGateBiasDOWN)
		delay(preGateTime)
		setDrainSrcLevelV(biasVD)

		for index=1,steps+1 do
			biasVg = stopVG -(stepVG*(index-1))
			setGateSrcLevelV(biasVg)
			table.insert(biasV,biasVg)
			delay(sweepDelay)
			smub.measure.i(drainBuffer)
		end

		setDrainSrcLevelV(0)

		setGatePowerOnOff(0)
		setDrainPowerOnOff(0)

		print("No\tTime\tGateV\tDrainI")
		for index=1,drainBuffer.n do
```

```
                print(index, drainBuffer.timestamps[index],biasV[index],drainBuffer[index])
            end
            initSMU()
        end


        local function defineFunction()
          setGateSrcRangeI = makesetter(smua.source,  'rangei')
          setGateSrcRangeV = makesetter(smua.source,  'rangev')
          setGateSrcLimitI = makesetter(smua.source,  'limiti')
          setGateSrcLimitV = makesetter(smua.source,  'limitv')
          setGateSrcLevelI = makesetter(smua.source,  'leveli')
          setGateSrcLevelV = makesetter(smua.source,  'levelv')
                setGateSrcFunction = makesetter(smua.source,  'func')
          setGatePowerOnOff = makesetter(smua.source, 'output')
                setGateSourceSettling = makesetter(smua.source, 'settling')
          setGateMeasRangeV = makesetter(smua.measure, 'rangev')
          setGateMeasRangeI = makesetter(smua.measure, 'rangei')
          setGateMeasNPLC  = makesetter(smua.measure, 'nplc')
                setGateMeasAutoZero = makesetter(smua.measure,'autozero')
          setGateAutoSourceRangeI = makesetter(smua.source, 'autorangei')
          setGateAutoSourceRangeV = makesetter(smua.source, 'autorangev')
          setGateAutoMeasureRangeI = makesetter(smua.measure, 'autorangei')
          setGateAutoMeasureRangeV = makesetter(smua.measure, 'autorangev')

          setDrainSrcRangeI = makesetter(smub.source,  'rangei')
          setDrainSrcRangeV = makesetter(smub.source,  'rangev')
          setDrainSrcLimitI = makesetter(smub.source,  'limiti')
          setDrainSrcLimitV = makesetter(smub.source,  'limitv')
          setDrainSrcLevelI = makesetter(smub.source,  'leveli')
          setDrainSrcLevelV = makesetter(smub.source,  'levelv')
                setDrainSrcFunction = makesetter(smub.source,  'func')
          setDrainPowerOnOff = makesetter(smub.source, 'output')
                setDrainSourceSettling = makesetter(smub.source, 'settling')
                setDrainSourceDelay = makesetter(smub.source, 'delay')
          setDrainMeasRangeV = makesetter(smub.measure, 'rangev')
          setDrainMeasRangeI = makesetter(smub.measure, 'rangei')
          setDrainMeasNPLC  = makesetter(smub.measure, 'nplc')
                setDrainMeasAutoZero = makesetter(smub.measure,'autozero')
                setDrainMeasDelay = makesetter(smub.measure, 'delay')
          setDrainAutoSourceRangeI = makesetter(smub.source, 'autorangei')
          setDrainAutoSourceRangeV = makesetter(smub.source, 'autorangev')
          setDrainAutoMeasureRangeI = makesetter(smub.measure, 'autorangei')
          setDrainAutoMeasureRangeV = makesetter(smub.measure, 'autorangev')
    end

    defineFunction()
    initSMU()

       return self
end

newVTH = newVthSiC_JEP183()

newVTH.vgDualSweepVdFixed(-3,15, 100e-3,0,2.8,0.05,1,20e-3, 0.1,0.5,0.002)
--newVTH.vgSweepVdFixed(15,100e-3,10e-3,2.8,0,-0.05,2.5,10e-3, 0.1, 0.5, 0.002)
--newVTH.vgVdBothSweep (15,100e-3,10e-3,2.8,0,-0.05,    10e-3, 0.1, 0.5, 0.002)
--newVTH.vgVdBothSweepWsingleSMU(15, 100e-3, 10e-3, 2.8, 0, -0.05, 10e-3, 7e-3, 0.1, 0.002)
--newVTH.vgVdBothFixedBias(-4, 100e-3, 10e-3, 5e-3, 20, 5, 1, 50e-3)
```