Making Microsecond Pulse and AC Measurements with the S530 Parametric Test System by Integrating 4200A-SCS Parameter Analyzer Applications

APPLICATION NOTE

KEITHLEY
A Tektronix Company

Tektronix®

Introduction

Keithley's S530 Parametric Test Systems (**Figure 1**) are designed for the collection of parametric data on the semiconductor fab production floor for process control monitoring or wafer acceptance testing. These systems can be configured to handle a broad range of devices and technologies. The tests that can be defined and written include basic transistor characterization tests (threshold voltage, I_{on}), capacitance measurements, electrical critical dimension (ECD), ring oscillator measurements, etc.

Figure 1. S530 Parametric Test System.

An S530 system includes DC instruments, a switch matrix, and a 4200A-SCS Parameter Analyzer, all installed in a cabinet. The user-configurable 4200A-SCS (**Figure 2**) can contain source measure units (SMUs), a capacitance voltage unit (CVU), a pulse generator unit (PGU), and a pulse measure unit (PMU). The 4200A-SCS is designed primarily for semiconductor characterization labs and is provided with many projects and tests for a variety of different applications. The 4200A-SCS system can also be controlled by an external computer using a program called KXCI (Keithley eXternal Communication Interface); this is how the S530 controls the 4200A-SCS.

Figure 2: 4200A-SCS Parameter Analyzer.

When configured into an S530 system, the 4200A-SCS is mostly used to run C-V tests and pulses for flash applications. This application note describes how to incorporate and use other tests from the 4200-SCS Parameter Analyzer in the S530 system, including a wide variety of other pulsed measurements, such as very fast pulsed I-V measurements using the Keithley Test Environment (KTE).

Use cases

The 4200A-SCS comes with a library of tests for semiconductor device characterization. Many of the tests are not available on the S530 KTE systems, but they can be integrated into the system. Some examples include:

1. **Tests that use pulse measure units (PMU).** These instruments provide the capability to force voltage and measure currents with pulse widths as short as 20–100 ns. These tests can be used for characterizing non-volatile memory (NVM) or power devices. The S530's standard capability is limited by its 150 µs pulse width, due to the limits in speed of SMU instruments.

2. **Tests that can perform parallel (up to four channels) C-V measurements.** This may be useful for parallel C-V measurements to achieve higher throughput. The application uses the same frequency for all four units.

3. **Non-volatile memory (NVM) characterization tests.** These tests were developed for PRAM, FeRAM, ReRAM, and flash technologies. Minimum hardware configuration: two remote pulse modules (RPMs) connected to the 4200A-SCS SMUs and the PMU.

Communication between Systems

Figure 3 shows the communication between the S530 and 4200A-SCS systems. The S530 KTE software installation must have the 'kxcicom' library installed to support communication between these systems. Test requests from the KTE software on the S530 are sent to the 4200A-SCS using the system's internal Ethernet connection. The 4200A-SCS receives this request through its KXCI interface, executes this test, and sends data back to the S530.

The 4200A-SCS includes the Keithley User Library Tool (KULT), which is used to develop and debug the user modules. User modules control the instruments and are executed by the S530 via the KXCI interface. A user library includes one or more user modules. Users can employ any of the many user modules that come in the library, can modify an existing user module, or can create new user modules.

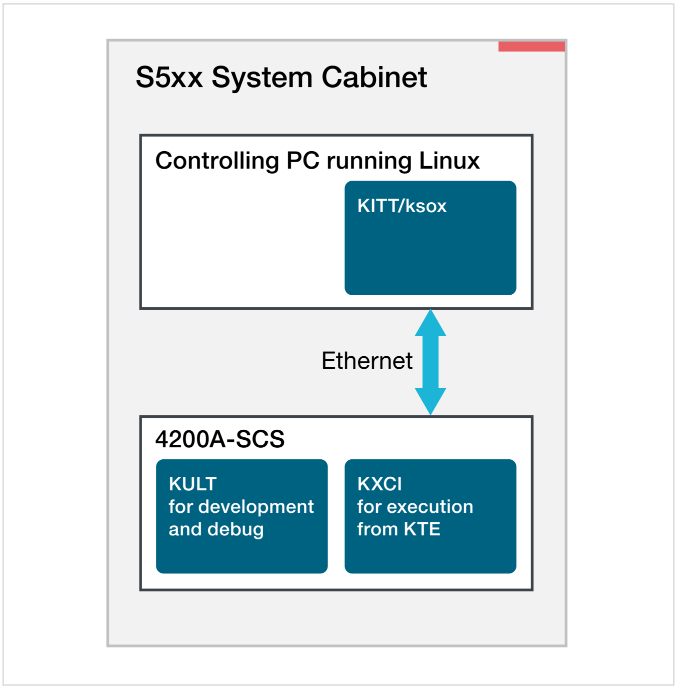

S5xx System Cabinet
Controlling PC running Linux
KITT/ksox
Ethernet
4200A-SCS
KULT for development and debug
KXCI for execution from KTE

Figure 3. Communication between the S530 Parametric Test System and the 4200A-SCS Parameter Analyzer.

Integrating the 4200A-SCS user libraries into the KTE System

Follow these steps to integrate the user libraries into the KTE system:

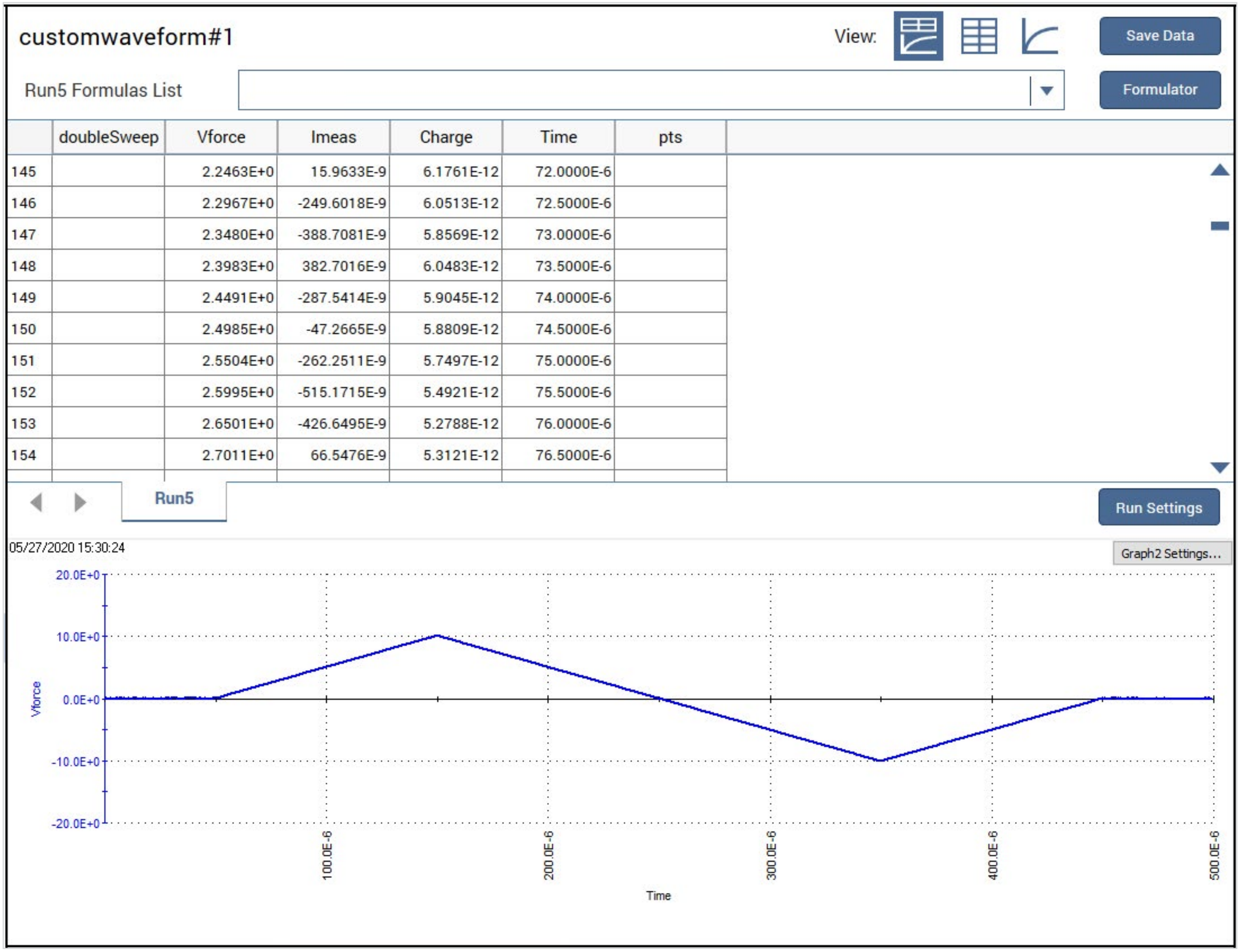
1. A test is developed and debugged on the 4200A-SCS within the Clarius software using user modules. Users can either use an existing user module or create their own using KULT. User modules are modified or written in the C-language code.
2. The user modules are tested in the Clarius software by creating and executing a User Test Module or UTM.
3. Once the user modules are debugged, they are remotely called by the S530 when the 4200A-SCS is set to the KXCI mode. This is the mode when the 4200A-SCS is set to remote control configuration and is anticipating receiving commands.
4. Using the KTE environment on the S530, properly formatted test command strings are sent to the 4200A-SCS system to execute the tests. Examples of these tests follow.

Example Applications

Two examples illustrate how to integrate the 4200A-SCS user libraries into the KTE environment: the pulse double sweep measurement test and the pulse current and measure voltage test.

Pulse double sweep measurement

This example application is a ramp-up/ramp-down sweep test using the 4225-PMU. In this test, the pulsed voltage sweeps in the sub-microsecond time scale and measures the resulting current. This particular test comes with the 4200A-SCS system in the '*nvm*' user library and is called the '*doubleSweep*' user module. **Figure 4** shows the results of executing this test. The graph shows the raw measurements of the transient voltage; the table lists the current and charge that the test measures.

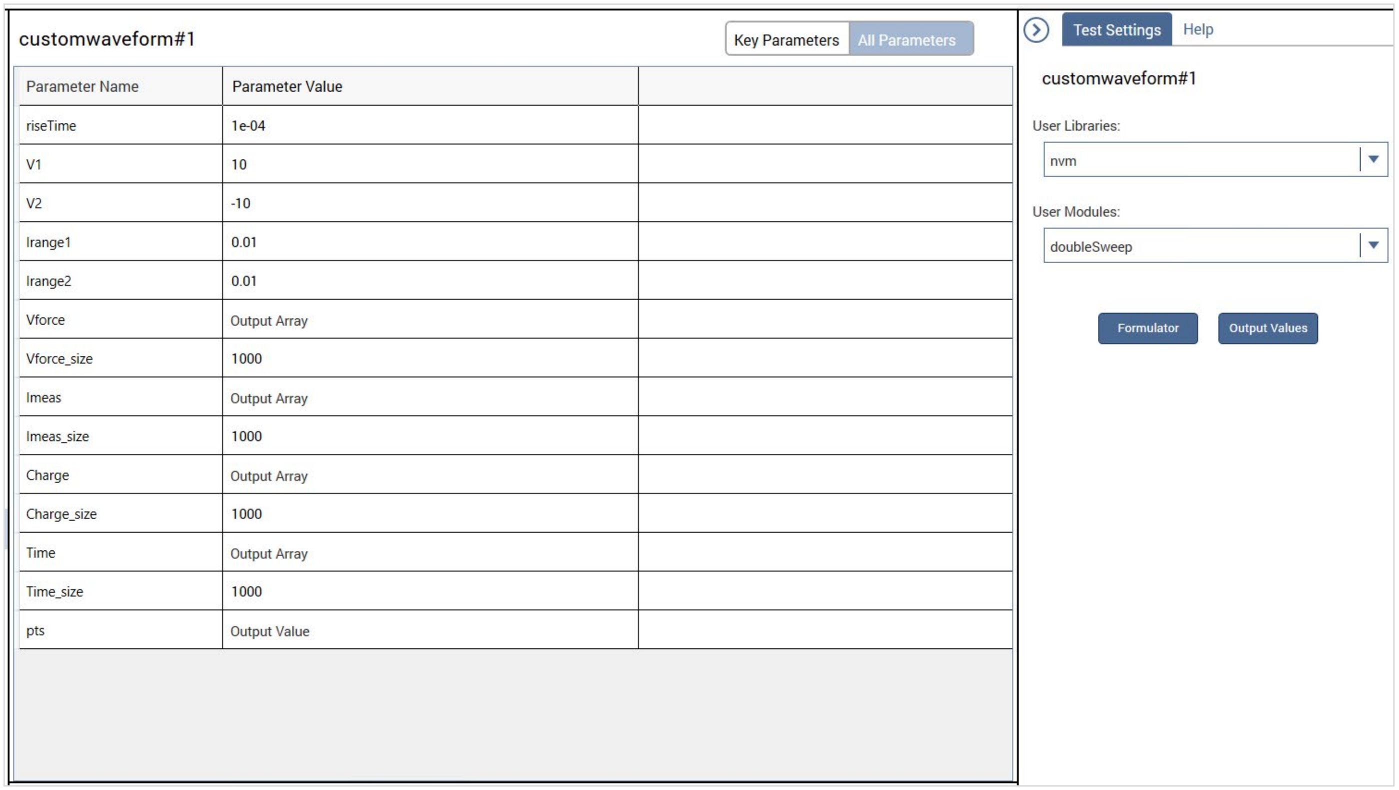


	doubleSweep	Vforce	Imeas	Charge	Time	pts
145		2.2463E+0	15.9633E-9	6.1761E-12	72.0000E-6	
146		2.2967E+0	-249.6018E-9	6.0513E-12	72.5000E-6	
147		2.3480E+0	-388.7081E-9	5.8569E-12	73.0000E-6	
148		2.3983E+0	382.7016E-9	6.0483E-12	73.5000E-6	
149		2.4491E+0	-287.5414E-9	5.9045E-12	74.0000E-6	
150		2.4985E+0	-47.2665E-9	5.8809E-12	74.5000E-6	
151		2.5504E+0	-262.2511E-9	5.7497E-12	75.0000E-6	
152		2.5995E+0	-515.1715E-9	5.4921E-12	75.5000E-6	
153		2.6501E+0	-426.6495E-9	5.2788E-12	76.0000E-6	
154		2.7011E+0	66.5476E-9	5.3121E-12	76.5000E-6	

05/27/2020 15:30:24

Figure 4. The *doubleSweep* user module graphed in the 4200A-SCS's Clarius Analyze view.

The test parameters of the '*doubleSweep*' user module (**Figure 5**) are shown in the 4200A-SCS's Configure view. As shown, the rise and fall times are 100 μs and the voltage peaks are set at ±10 V. The sweep has a total of 1,000 measured points, with 500 ns measurement time steps.



customwaveform#1

Parameter Name	Parameter Value
riseTime	1e-04
V1	10
V2	-10
Irange1	0.01
Irange2	0.01
Vforce	Output Array
Vforce_size	1000
Imeas	Output Array
Imeas_size	1000
Charge	Output Array
Charge_size	1000
Time	Output Array
Time_size	1000
pts	Output Value

User Libraries: nvm

User Modules: doubleSweep

Figure 5. Configure view of the Clarius software on the 4200A-SCS listing the test parameters of the *doubleSweep* user module.

Once this test is properly configured and executed on the 4200A-SCS system, it can be executed from the S530 KTE environment, either as a KULT module call or LPT calls.

Appendix A lists example code using a KULT module call. In this case, the input parameters of the *doubleSweep* user module in the KTE software can be:

stat [0]= doubleSweep(4, 5,1e-4, 10, -10,1e-2, vF, 1000, iMeas, 1000, Q, 1000, Time, 1000, pts);

Here's an example using a sequence of LPT calls:

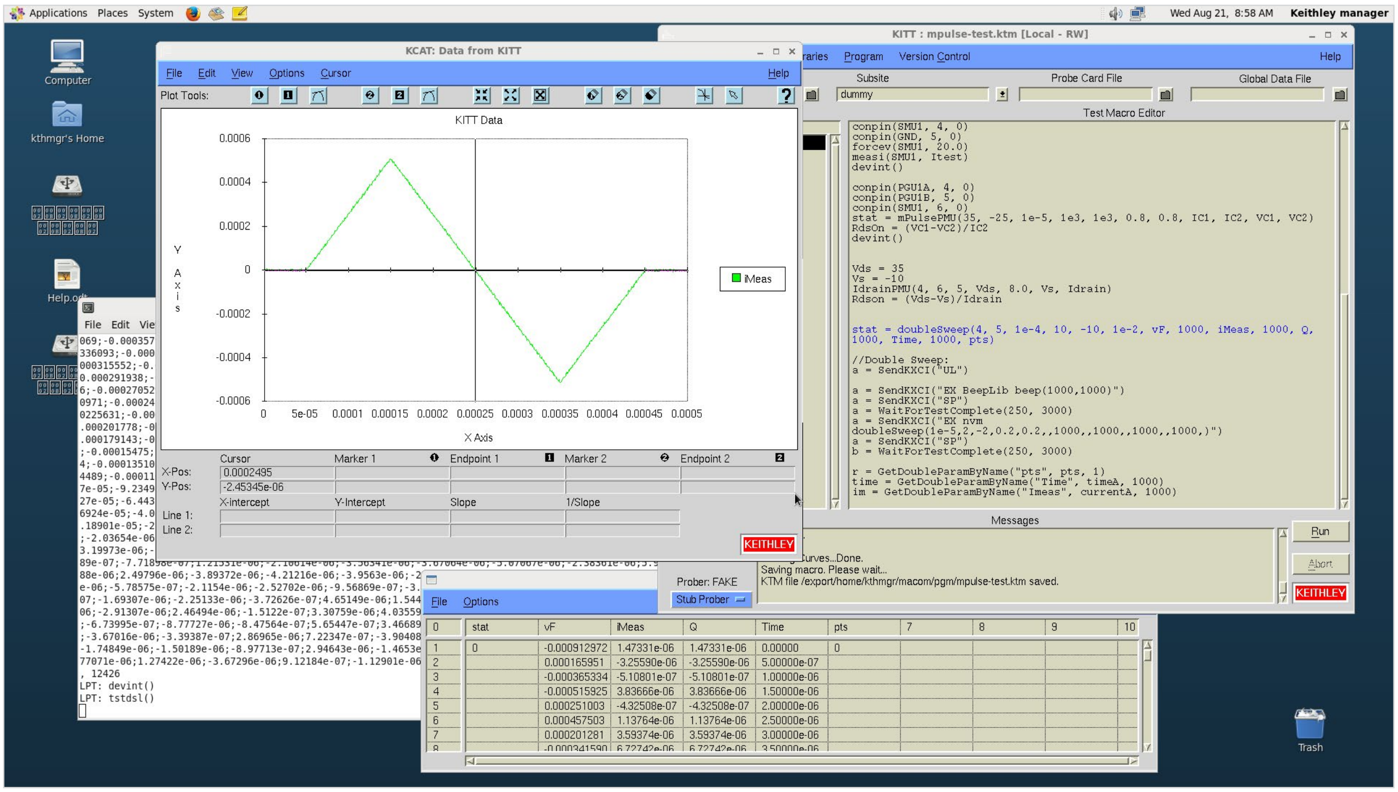
```
//Double Sweep:
a [0]= SendKXCI("UL");
a [0]= SendKXCI("EX BeepLib beep(1000,1000)");
a [0]= SendKXCI("SP");
a [0]= WaitForTestComplete(250, 3000);
a [0]= SendKXCI("EX nvm doubleSweep(1e-5,2,-2,0.2,0.2,,1000,,1000,,1000,,1000,)");
a [0]= SendKXCI("SP");
b [0]= WaitForTestComplete(250, 3000);
r [0]= GetDoubleParamByName("pts", pts, 1);
time [0]= GetDoubleParamByName("Time", timeA, 1000);
im [0]= GetDoubleParamByName("Imeas", currentA, 1000);
```

These functions are part of the *kxcicom* library, which must be installed on the S530 system. This library communicates with the 4200A-SCS system through the KXCI interface. The library's main function is the *SendKXCI* function, which sends the string to be executed by KXCI. The following explanations and notes describe commands from the previous block of code:

- **SendKXCI():**
  - **UL**, this argument is an instruction to KXCI to expect a request to execute a call from UL (User Library).
  - **EX BeepLib beep(1000,1000)** is an instruction to load the user library named 'BeepLib' with a function call 'beep'.
  - **EX nvm doubleSweep(1e-5,2,-2,0.2,0.2,,1000,,1000,,1000,,1000,)")** calls the user library 'nvm' with a function call 'doubleSweep', with a list of explicitly specified input arguments.

    Note: The names of the output arguments are dropped and must stay blank and are separated by commas.
  - SP, serial spoll, is a command that confirms acknowledgment of the ongoing communication.
- **WaitForTestComplete(200,3000) —** This function has two arguments: the first one is spoll time, which in this case is 200 (ms), to confirm that command is completed; the second argument is timeout in ms.
- **im [0]= GetDoubleParamByName("Imeas", currentA, 1000) —** The GetDoubleParamByName returns a pointer to the measured data. The first argument should be the string variable, which matches the output argument name of the called function, "Imeas". The second argument, "currentA", is a pointer to the allocated variable of size 1000.

Figure 6 shows the test coded and presented on the S530, which sends a request to the 4200A-SCS to run the doubleSweep test from the NVM library. The data is retrieved and plotted on the screen.



	Cursor	Marker 1	Endpoint 1	Marker 2	Endpoint 2
X-Pos:	0.0002495				
Y-Pos:	-2.45345e-06				

	X-intercept	Y-Intercept	Slope	1/Slope
Line 1:				
Line 2:				

Subsite	dummy
Probe Card File	
Global Data File	

```
conpin(SMU1, 4, 0)
conpin(GND, 5, 0)
forcev(SMU1, 20.0)
measi(SMU1, Itest)
devint()

conpin(PGU1A, 4, 0)
conpin(PGU1B, 5, 0)
conpin(SMU1, 6, 0)
stat = mPulsePMU(35, -25, 1e-5, 1e3, 1e3, 0.8, 0.8, IC1, IC2, VC1, VC2)
RdsOn = (VC1-VC2)/IC2
devint()


Vds = 35
Vs = -10
IdrainPMU(4, 6, 5, Vds, 8.0, Vs, Idrain)
Rdson = (Vds-Vs)/Idrain


stat = doubleSweep(4, 5, 1e-4, 10, -10, 1e-2, vF, 1000, iMeas, 1000, Q,
1000, Time, 1000, pts)

//Double Sweep:
a = SendKXCI("UL")

a = SendKXCI("EX BeepLib beep(1000,1000)")
a = SendKXCI("SP")
a = WaitForTestComplete(250, 3000)
a = SendKXCI("EX nvm
doubleSweep(1e-5,2,-2,0.2,0.2,,1000,,1000,,1000,,1000,)")
a = SendKXCI("SP")
b = WaitForTestComplete(250, 3000)

r = GetDoubleParamByName("pts", pts, 1)
time = GetDoubleParamByName("Time", timeA, 1000)
im = GetDoubleParamByName("Imeas", currentA, 1000)
```

```
Curves...Done.
Saving macro. Please wait...
KTM file /export/home/kthmgr/macom/pgm/mpulse-test.ktm saved.
```

```
069;-0.000357
336093;-0.000
000315552;-0.
0.000291938;-
6;-0.00027052
0971;-0.00024
0225631;-0.00
.000201778;-0
.000179143;-0
;-0.00015475;
4;-0.00013510
4489;-0.00011
7e-05;-9.2349
27e-05;-6.443
6924e-05;-4.0
.18901e-05;-2
;-2.03654e-06
3.19973e-06;-
89e-07;-7.718
88e-06;2.49796e-06;-3.89372e-06;-4.21216e-06;-3.9563e-06;-2
e-06;-5.78575e-07;-2.1154e-06;-2.52702e-06;-9.56869e-07;-3.
07;-1.69307e-06;-2.25133e-06;-3.72626e-07;4.65149e-06;1.544
06;-2.91307e-06;2.46494e-06;-1.5122e-07;3.30759e-06;4.03559
;-6.73995e-07;-8.77727e-06;-8.47564e-07;5.65447e-07;3.46689
;-3.67016e-06;-3.39387e-07;2.86965e-06;7.22347e-07;-3.90408
-1.74849e-06;-1.50189e-06;-8.97713e-07;2.94643e-06;-1.4653e
77071e-06;1.27422e-06;-3.67296e-06;9.12184e-07;-1.12901e-06
, 12426
LPT: devint()
LPT: tstdsl()
```

0	stat	vF	iMeas	Q	Time	pts	7	8	9	10
1	0	-0.000912972	1.47331e-06	1.47331e-06	0.00000	0				
2		0.000165951	-3.25590e-06	-3.25590e-06	5.00000e-07					
3		-0.000365334	-5.10801e-07	-5.10801e-07	1.00000e-06					
4		-0.000515925	3.83666e-06	3.83666e-06	1.50000e-06					
5		0.000251003	-4.32508e-07	-4.32508e-07	2.00000e-06					
6		0.000457503	1.13764e-06	1.13764e-06	2.50000e-06					
7		0.000201281	3.59374e-06	3.59374e-06	3.00000e-06					
8		-0.000341590	6.72742e-06	6.72742e-06	3.50000e-06					

Figure 6. Results of calling the doubleSweep user module in the KTE environment.

Pulse current/voltage measurement using PMU: mPulse

The previous example illustrated in **Figure 6** was a test involving returning an array of data to show the transient device response. The following example shows how to run a single measurement test. *mPulse* is a simple test, with only a few parameters returned, which cannot be graphically represented. Here is the code:

```
conpin(PGU1A, 4, 0); //Instead of PGU (Pulse Generator Unit) unit it is possible to use GPT
(General Purpose Terminal)
conpin(PGU1B, 5, 0);
conpin(SMU1, 6, 0);
stat [0]= mPulsePMU(35, -25,1e-5, 1e3, 1e3, 0.8, 0.8, IC1, IC2, VC1, VC2);
RdsOn [0]= (*VC1 - *VC2 )/ *IC2;
devint();
```

In this case, the *mPulsePMU* function (shown in Appendix B) is used as an LPT function, which assumes that connections and test setup must be done outside of the test. The commands of interest from *mPulsePMU* are:

```
stat = SendKXCI("UL");
KTXEDebugMsg("%s: starts\n", mod);
sprintf(cmd, "EX mpulse mPulse(%g,%g,%g,%g,%g,%g,%g,,,,)",
    V1, V2, pulseWidth, Load1, Load2, iRange1, iRange2);
KTXEDebugMsg("%s: CMD:%s\n", mod, cmd);

stat = SendKXCI(cmd);
stat = SendKXCI("SP");
stat = WaitForTestComplete(delay, timeout);

r = GetDoubleParamByName("iC1", iC1, 1);
r = GetDoubleParamByName("iC2", iC2, 1);
r = GetDoubleParamByName("vC1", vC1, 1);
r = GetDoubleParamByName("vC2", vC2, 1);

stat = GetReturnValue();
```

First, the 'UL' command is sent to the 4200A-SCS to set it up to expect a request for the user test module. In the next two lines, the command string (*cmd*) is properly formatted and sent to the 4200A-SCS. This command calls for the user module '*mPulse*' from the 4200A-SCS library '*mpulse*' to be executed with corresponding values for *pulseWidth*, *Load1* (PMU channel1 load value), Load2 (PMU channel2 load value) and current ranges for both channels, *iRange1* and *iRange2*. After serial polling and confirmation of the completion of the test (*WaitForTestComplete*) is done, the data can be processed.

Return data is processed in the same way as in the previous example, by using the function *GetDoubleParamByName*("iC1", iC1, 1), where the string 'iC1' should match the output name of the 4200-SCS UTM module (see Appendix B). Here, iC1 is the pointer to the *double value in the C-routine on KTE. An additional function used here, *GetReturnValue*(), returns values of non-void functions, which usually contain the execution or status/error flag.

Please note that output names are not defined and left blank in the request for test execution in '*cmd*' string: *sprintf(cmd*, "*EX mpulse mPulse(%g,%g,%g,%g,%g,%g,%g,,,,)*". Last four arguments are not specified, and are separated by commas.

The *mPulse* function from the *mpulse* user library of the 4200A-SCS was coded, debugged, and executed on the 4200A-SCS unit. This particular user library and user module is not included in the library of the 4200A-SCS, but it was shown as an example. The code for this routine uses PMU control functions.

Conclusion

The S530 can generate microsecond pulsed I-V sweeps and tests, as well as make AC impedance measurements by calling user libraries from the 4200A-SCS Parameter Analyzer. Users can employ the many built-in user libraries that come with the 4200A-SCS or they can create their own code.

Appendix A

Code for *doubleSweep*.c for S530 KTE

```
/* USRLIB MODULE INFORMATION

       MODULE NAME: doubleSweep
       MODULE RETURN TYPE: int
       NUMBER OF PARMS: 15
       ARGUMENTS:
              pinH,  int,   Input, ,      ,
              pinL,  int,   Input, ,      ,
              riseTime,     double,       Input, ,      ,
              V1,    double,       Input, ,      ,
              V2,    double,       Input, ,      ,
              iRange,       double,       Input, ,      ,
              vForce,       D_ARRAY_T,    Output,        ,      ,
              vFpts, int,   Input, ,      ,
              iMeas, D_ARRAY_T,    Output,        ,      ,
              iMeasPts,     int,   Input, ,      ,
              Charge,       D_ARRAY_T,    Output,        ,      ,
              ChargePts,    int,   Input, ,      ,
              Time,  D_ARRAY_T,    Output,        ,      ,
              TimePts,      int,   Input, ,      ,
              pts,   int *, Output,       ,      ,
       INCLUDES:
#include <stdio.h>
#include <stdlib.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
       END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION

       END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE VERSION CONTROL */
static char const vcid[] ="$Id: doubleSweep.c Local $";
/* USRLIB MODULE PARAMETER LIST */
#include <stdio.h>
#include <stdlib.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
```

```
int doubleSweep( int pinH, int pinL, double riseTime, double V1, double V2, double iRange, double
*vForce, int vFpts, double *iMeas, int iMeasPts, double *Charge, int ChargePts, double *Time, int
TimePts, int *pts )
{
/* USRLIB MODULE CODE */
int stat = 1;
char mod [] = "doubleSweep";
int a = 0;
char cmd[128];
int npts;
int timeout = 3000;

if(iMeasPts != vFpts || ChargePts != vFpts || vFpts != TimePts || iMeasPts < 10)
{
    stat = -1;
    KTXEDebugMsg("%s: Returns %d\n", mod, stat);
    return;
}else{npts = vFpts;}

conpin(PGU1A, pinH);
conpin(PGU1B, pinL);

sprintf(cmd, "EX nvm doubleSweep(%g,%g,%g,%g,%g,,%d,,%d,,%d,,%d,)",
    riseTime, V1, V2, iRange, iRange, npts, npts, npts,npts);

a = SendKXCI("UL");
a = SendKXCI(cmd);
a = SendKXCI("SP");
a = WaitForTestComplete(250, timeout);

a = GetDoubleParamByName("pts", pts, 1);
a = GetDoubleParamByName("Time", Time, npts);
a = GetDoubleParamByName("Imeas", iMeas, npts);
a = GetDoubleParamByName("Vforce", vForce, npts);
a = GetDoubleParamByName("Imeas", Charge, npts);
stat = GetReturnValue();
devint();

return(stat);
/* USRLIB MODULE END  */
}             /* End doubleSweep.c */
```

Appendix B

Code for *mPulsePMU*.c for S530KTE:

```
/* USRLIB MODULE INFORMATION

	MODULE NAME: mPulsePMU
	MODULE RETURN TYPE: int
	NUMBER OF PARMS: 11
	ARGUMENTS:
		V1,	double,		Input, 1,	,
		V2,	double,		Input, 0,	,
		pulseWidth,	double,		Input, 1e-5,	,
		Load1, double,		Input, 1e3,	,
		Load2, double,		Input, 1e3,	,
		iRange1,	double,		Input, 0.2,	,
		iRange2,	double,		Input, 0.2,	,
		iC1,	double *,	Output,		,	,
		iC2,	double *,	Output,		,	,
		vC1,	double *,	Output,		,	,
		vC2,	double *,	Output,		,	,
	INCLUDES:
#include <stdio.h>
#include <stdlib.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>
	END USRLIB MODULE INFORMATION
*/
/* USRLIB MODULE HELP DESCRIPTION

	END USRLIB MODULE HELP DESCRIPTION */
/* USRLIB MODULE VERSION CONTROL */
static char const vcid[] ="$Id: mPulsePMU.c Local $";
/* USRLIB MODULE PARAMETER LIST */
#include <stdio.h>
#include <stdlib.h>
#include <lptdef.h>
#include <lptdef_lowercase.h>
#include <math.h>

int mPulsePMU( double V1, double V2, double pulseWidth, double Load1, double Load2, double
iRange1, double iRange2, double *iC1, double *iC2, double *vC1, double *vC2 )
{
/* USRLIB MODULE CODE */
char cmd[256];
char mod[] = "mPulsePMU";
int stat, r;
int delay = 250;
int timeout = 5000;

stat = SendKXCI("UL");
KTXEDebugMsg("%s: starts\n", mod);
sprintf(cmd, "EX mpulse mPulse(%g,%g,%g,%g,%g,%g,%g,,,,)",
    V1, V2, pulseWidth, Load1, Load2, iRange1, iRange2);
KTXEDebugMsg("%s: CMD:%s\n", mod, cmd);

stat = SendKXCI(cmd);
stat = SendKXCI("SP");
stat = WaitForTestComplete(delay, timeout);

r = GetDoubleParamByName("iC1", iC1, 1);
r = GetDoubleParamByName("iC2", iC2, 1);
r = GetDoubleParamByName("vC1", vC1, 1);
r = GetDoubleParamByName("vC2", vC2, 1);

stat = GetReturnValue();
return(stat);

/* USRLIB MODULE END  */
}             /* End mPulsePMU.c */
```

Contact Information:

Australia	1 800 709 465
Austria*	00800 2255 4835
Balkans, Israel, South Africa and other ISE Countries	+41 52 675 3777
Belgium*	00800 2255 4835
Brazil	+55 (11) 3759 7627
Canada	1 800 833 9200
Central East Europe / Baltics	+41 52 675 3777
Central Europe / Greece	+41 52 675 3777
Denmark	+45 80 88 1401
Finland	+41 52 675 3777
France*	00800 2255 4835
Germany*	00800 2255 4835
Hong Kong	400 820 5835
India	000 800 650 1835
Indonesia	007 803 601 5249
Italy	00800 2255 4835
Japan	81 (3) 6714 3086
Luxembourg	+41 52 675 3777
Malaysia	1 800 22 55835
Mexico, Central/South America and Caribbean	52 (55) 56 04 50 90
Middle East, Asia, and North Africa	+41 52 675 3777
The Netherlands*	00800 2255 4835
New Zealand	0800 800 238
Norway	800 16098
People's Republic of China	400 820 5835
Philippines	1 800 1601 0077
Poland	+41 52 675 3777
Portugal	80 08 12370
Republic of Korea	+82 2 565 1455
Russia / CIS	+7 (495) 6647564
Singapore	800 6011 473
South Africa	+41 52 675 3777
Spain*	00800 2255 4835
Sweden*	00800 2255 4835
Switzerland*	00800 2255 4835
Taiwan	886 (2) 2656 6688
Thailand	1 800 011 931
United Kingdom / Ireland*	00800 2255 4835
USA	1 800 833 9200
Vietnam	12060128

*** European toll-free number. If not accessible, call:** +41 52 675 3777

Rev. 02.2018

KEITHLEY
A Tektronix Company

Find more valuable resources at TEK.COM

092220 SBG 1KW-67746-0