

HP 4142B Modular DC Source/Monitor

Control Software Programming Manual



**HEWLETT
PACKARD**

HP Part No. 04142-90310
Printed in Japan Jun 1991

Edition 1
E0691

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

WARNING



HIGH VOLTAGE SHOCK HAZARD (MAX. 1000 V dc)

The HP 41423A HVU (± 1000 V), HP 41420A SMU (± 200 V), and HP 41421B SMU (± 100 V) force dangerous voltages on the FORCE, GUARD, and SENSE terminals. To prevent an electrical shock, the following safety precautions must be observed.

- ◆ Ground the HP 4142B using a three-conductor ac power cable.
- ◆ Connect the Interlock (INTLK) terminal to a switch that turns off when the shielding box access door is opened.
- ◆ For HVU, connect the OUTPUT ON/OFF STATUS terminal to a warning indicator.
- ◆ For HVU, perform the operation tests of the INTLK and OUTPUT ON/OFF STATUS circuits at least once a day, before using the HP 4142B.
- ◆ Before touching the connections of the FORCE, GUARD, and SENSE terminals, turn the HP 4142B off, and discharge any capacitors (if connected).
 - If you do not turn the HP 4142B off, perform the following four steps:
 - 1) Set the HVU and SMU output switches to off.
 - 2) For HVU, confirm that the warning indicator is not lit.
 - 3) Open the shielding box access door (open the INTLK terminal).
 - 4) Discharge any capacitors, if connected.
- ◆ Warn workers around the HP 4142B about dangerous conditions.

JAPANESE: HP 4142Bの安全上の注意



高電圧感電注意 (最大 1000 V dc)

HP 41423A HVU (± 1000 V)、HP 41420A SMU (± 200 V)、およびHP 41421B SMU (± 100 V)は、危険電圧をFORCE端子、GUARD端子、およびSENSE端子に出力します。感電事故防止のため、必ず下記の事項を実施してください。

- ◆ 3極電源ケーブルを使用して、HP 4142Bを接地する
- ◆ インターロック (INTLK) 端子を、シールド・ボックスの蓋が開いたときにオープンとなるよう接続する
- ◆ OUTPUT ON/OFF STATUS端子を警告インジケータに接続する (HVU使用時)
- ◆ INTLK回路およびOUTPUT ON/OFF STATUS回路の動作テストを、1日に1回以上、使用前に行う (HVU使用時)
- ◆ FORCE端子、GUARD端子、およびSENSE端子の接続に触れる前に、HP 4142Bの電源をオフにし、キャパシタが接続されているならば、キャパシタを放電する
 - 電源をオフにしない場合には、下記の4事項をすべて実施する
 - 1) HVUおよびSMUの出力スイッチをオフにする
 - 2) 警告インジケータが消灯していることを確認する (HVU使用時)
 - 3) シールド・ボックスの蓋をあける (INTLK端子をオープンにする)
 - 4) キャパシタが接続されているならば、キャパシタを放電する
- ◆ 周囲の他の作業者に対しても、高電圧危険に対する注意を徹底する

GERMAN: Sicherheitsmaßnahmen für den HP 4142B

WARNUNG



HOCHSPANNUNGS-BERÜHRUNGSGEFAHR (MAX. 1000 VDC)

Bei den Geräten HP 41423A HVU (± 1000 V), HP 41420A SMU (± 200 V) und HP 41421B SMU (± 100 V) gefährliche spannungen an den FORCE-, GUARD- und SENSE-Klemmen. Um einen Elektroschock zu vermeiden, sind folgende Sicherheitsmaßnahmen zu beachten.

- ◆ Gerät HP 4142B mit einem Dreileiter-AC-Starkstromkabel erden.
- ◆ Die Interlock-Klemme (INTLK) mit einem Schalter verbinden, der beim Öffnen der Abschirmkasten-Zugangstür ausgeschaltet wird.
- ◆ Bei Gerät HVU die Klemme OUTPUT ON/OFF STATUS mit einer Warnanzeige verbinden.
- ◆ Bei Gerät HVU die Funktionsprüfungen der Schaltkreise INTLK und OUTPUT ON/OFF STATUS mindestens einmal täglich durchführen, bevor HP 4142B verwendet wird.
- ◆ Vor Berühren der Verbindungen an den FORCE-, GUARD- und SENSE-Klemmen, Gerät HP 4142B ausschalten und (falls angeschlossen), die Kondensatoren entladen.
Falls HP 4142B nicht ausgeschaltet wird, sind folgende vier Schritte durchzuführen:
 - 1) Die Ausgangsschalter von HVU und SMU auf AUS stellen.
 - 2) Bei HVU kontrollieren, ob die Warnanzeige nicht leuchtet.
 - 3) Die Zugangstür des abgeschirmten Kasten öffnen (die Klemme INTLK öffnen).
 - 4) Vorhandene Kondensatoren entladen.
- ◆ Die Arbeitskräfte im Bereich des HP 4142B über die bestehende Gefahr unterrichten.

FRENCH: Consignes de sécurité relatives à l'équipement HP 4142B

DANGER D'ELECTROCUTION



HAUTE TENSION CONTINUE (JUSQU'À 1000 Vc.c.)

Les instruments HP 41423A HVU (± 1000 V), HP 41420A SMU (± 200 V) et HP 41421B SMU (± 100 V) présentent des tensions dangereuses aux bornes "FORCE", "GUARD" et "SENSE". Pour éviter tout risque d'électrocution, respecter les consignes suivantes.

- ◆ Mettre à la terre l'équipement HP 4142B en utilisant un câble secteur triphasé.
- ◆ Connecter la borne de verrouillage "INTLK" à un commutateur coupant l'alimentation lorsque la porte d'accès à la boîte blindée est ouverte.
- ◆ Pour le module HVU, connecter la borne "OUTPUT ON/OFF STATUS" à une lampe d'avertissement.
- ◆ Pour le module HVU, effectuer les essais de fonctionnement des circuits "INTLK" et "OUTPUT ON/OFF STATUS" au moins une fois par jour, avant d'utiliser l'équipement HP 4142B.
- ◆ Avant de toucher les connexions des bornes "FORCE" "GUARD" et "SENSE", mettre hors tension l'équipement HP 4142B et décharger tous les condensateurs éventuellement raccordés.
Au lieu de mettre l'équipement HP 4142B hors tension, l'on peut procéder de la manière suivante:
 - 1) Mettre les commutateurs de sortie des modules HVU et SMU en position d'arrêt.
 - 2) Pour le module HVU, s'assurer que la lampe d'avertissement est éteinte.
 - 3) Ouvrir la porte d'accès à la boîte blindée (pour mettre hors circuit la borne "INTLK").
 - 4) Décharger tous les condensateurs éventuellement raccordés.
- ◆ Avertir toute personne travaillant à proximité de l'équipement HP 4142B des dangers que présente cet équipement.

Printing History

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

June 1991 ... Edition 1

Contents

1. General Information	
How To Use This Manual	1-1
What Is Control Software?	1-1
Control Software Files	1-1
Required Bin Files	1-2
2. Programming Basics	
Introduction	2-1
Before Programming	2-1
Programming Procedures	2-2
Measurement Program Flow	2-2
Collector Characteristics Example Program	2-3
Declare Com Block and Option Base	2-4
Declare Array Variables for Measurement Data	2-4
Assign Constants	2-4
Initialize Computer and HP 4142B	2-4
Set Measurement Conditions	2-5
Set Output Switches to ON	2-5
Force Voltage/Current	2-5
Perform Measurement	2-5
Zero Output	2-6
Set Output Switches to OFF	2-6
Display Measurement Results	2-6
Executing Measurement Programs	2-7
3. Control Software Description	
Introduction	3-1
Subprogram Descriptions	3-2
HP4142_DRV Subprograms	3-2
GRAPHICS Subprograms	3-3
PARA_4142 User-Defined Functions	3-3
High Speed Spot Measurements	3-4
Staircase Sweep Measurements	3-5
1-ch Pulsed Spot Measurements	3-6
Pulsed Sweep Measurements	3-7
Staircase Sweep With Pulsed Bias Measurements	3-8
2-ch Pulsed Spot Measurements	3-9
Pulsed Sweep With Pulsed Bias Measurements	3-10
Analog Search Measurements	3-11
Quasi-Pulsed Measurements	3-12
Parameter Measurements Using Para_4142 Functions	3-13

4. Programming Reference

Introduction	4-1
Conventions	4-2
Reading the Syntax	4-2
CALL Statement	4-2
Parameters	4-2
Auto_cal	4-4
Cal_hp4142	4-5
Ch_sw_off	4-6
Ch_sw_on	4-7
Connect_relay1	4-8
Connect_relay2	4-9
Dpulse_i	4-10
Dpulse_measure	4-12
Dpulse_v	4-14
Dsweep_piv	4-16
Dump_screen	4-18
FNBvcbo	4-19
FNBvces	4-21
FNHfe	4-23
FNIds	4-26
FNR_measure	4-29
FNVth1	4-33
FNVth2	4-35
FNVth3	4-38
Force_i	4-41
Force_v	4-43
Init_computer	4-45
Init_hp4142	4-46
Lingraph	4-47
Loggraph	4-49
Measure_asearch	4-51
Measure_bdm	4-54
Measure_i	4-55
Measure_v	4-57
Para_hfe	4-59
Para_vth	4-62
Pulse_i	4-65
Pulse_measure	4-67
Pulse_v	4-69
Recover_output	4-71
Self_test	4-72
Set_amonitor	4-74
Set_asource	4-76
Set_bdm	4-79
Set_iv	4-81
Set_piv	4-85
Set_pol	4-89
Set_smu	4-91
Set_vm	4-92
status	4-93

Sweep_iv	4-94
Sweep_miv	4-96
Sweep_mode	4-98
Sweep_pbias	4-99
Sweep_piv	4-101
Wbuild_file	4-103
Zero_output	4-105

A. Program Listings

HP4142_DRV File	A-1
Auto_cal	A-1
Cal_hp4142	A-1
Ch_sw_on	A-2
Ch_sw_off	A-3
Zero_output	A-4
Recover_output	A-5
Set_smu	A-6
Init_hp4142	A-6
Self_test	A-7
Force_i	A-8
Force_v	A-9
Measure_i	A-10
Measure_v	A-11
Set_vm	A-12
Set_iv	A-13
Sweep_iv	A-15
Sweep_miv	A-17
Sweep_mode	A-20
Dpluse_v	A-20
Pulse_v	A-21
Dpulse_i	A-21
Pulse_i	A-22
Dpulse_measure	A-23
Pulse_measure	A-25
Set_piv	A-26
Dsweep_piv	A-27
Sweep_piv	A-29
Sweep_pbias	A-31
Set_asource	A-33
Set_amonitor	A-33
Measure_asearch	A-34
Para_vth	A-36
Para_hfe	A-40
Set_v_range	A-44
Set_i_range	A-45
Module_type	A-46
Detect_error	A-46
Tau_read	A-47
Cal_hold_time	A-49
V_range	A-49
Set_bdm	A-50

Measure_bdm	A-51
Set_pol	A-52
Connect_relay1	A-52
Connect_relay2	A-53
GRAPHICS File	A-55
Dump_screen	A-55
Init_computer	A-56
Wbuild_file	A-58
Lingraph	A-60
Loggraph	A-62
Draw_lin	A-65
Draw_log1	A-67
Draw_log2	A-70
Draw_log3	A-73
Disp_error	A-76
Zoom_plot_area	A-77
Write_title	A-78
Zoom_graph_area	A-79
Write_axes_name	A-80
Calc_lin_axis	A-81
Label_lin_xaxis	A-82
Label_lin_yaxis	A-85
Calc_log_axis	A-88
Label_log_yaxis	A-90
Label_log_xaxis	A-92
PARA_4142 File	A-94
FNHfe	A-94
FNbvces	A-96
FNIds	A-97
FNVth1	A-99
FNVth2	A-100
FNVth3	A-102
FNR_measure	A-103

Figures

2-1. Example Program for Measuring Collector Characteristics 2-3

General Information

How To Use This Manual

This manual contains the information required to program the HP 4142B Modular DC Source/Monitor using the furnished Control Software.

Chapter 1 provides general control software information; Chapter 2 describes control software programming basics and provides an example program; Chapter 3 describes the types of measurements that can be performed using the control software; Chapter 4 provides an alphabetical listing of control software subprograms and user-defined functions, including syntax and other usage rules; and Appendix A contains the source code for the subprograms and user-defined functions.

Detailed HP 4142B hardware information and operating procedures are discussed in the Operation Manual.

What Is Control Software?

The HP 4142B's control software is a library of subprograms and user defined functions written in BASIC, which allows you to easily modify and enhance the software to suit your measurement requirements. Control software is designed for use on HP 9000 Series 200/300 computers running BASIC 3.0 or later versions, and is provided on a 3.5 inch micro-flexible disc (single-sided format). The three main functions of the 4142B's control software are:

- To control the HP 4142B's plug-in units (HP4142_DRV).
- To display measurement data in a graphics format (GRAPHICS).
- To measure and calculate semiconductor DC parameters (PARA_4142).

Control Software Files

Control software consists of three files, HP4142_DRV, GRAPHICS, and PARA_4142, as briefly described in the following paragraphs.

- | | |
|------------|--|
| HP4142_DRV | Contains subprograms for controlling HP 4142B operation. You can use these subprograms just as you would BASIC keywords, thus simplifying measurement programming. |
| GRAPHICS | Contains utility subprograms for displaying measurement results in a graphics format. These subprograms simplify the scaling, labeling, and plotting of graph axes, and can even provide an optional grid pattern. |
| PARA_4142 | Contains user-defined functions for obtaining complex DC semiconductor parameter measurement results with a minimum of time and effort. |

For more complete control software information, refer to Chapters 3 and 4.

Required Bin Files

The HP 4142B's control software requires the following BIN files.

Driver HPIB

Language GRAPH, GRAPHX, IO, MAT, PDEV, KBD, ERR
Extensions

If you're using an external disc drive or other storage medium, you may need to load other BIN files in addition to these. Refer to the BASIC User's Guide for more details.

Programming Basics

Introduction

This chapter describes basic control software programming procedures for controlling the HP 4142B. To simplify the explanation procedure, a practical example for measuring the collector characteristics (I_C - V_C curve) of a bipolar transistor is included.

Before Programming

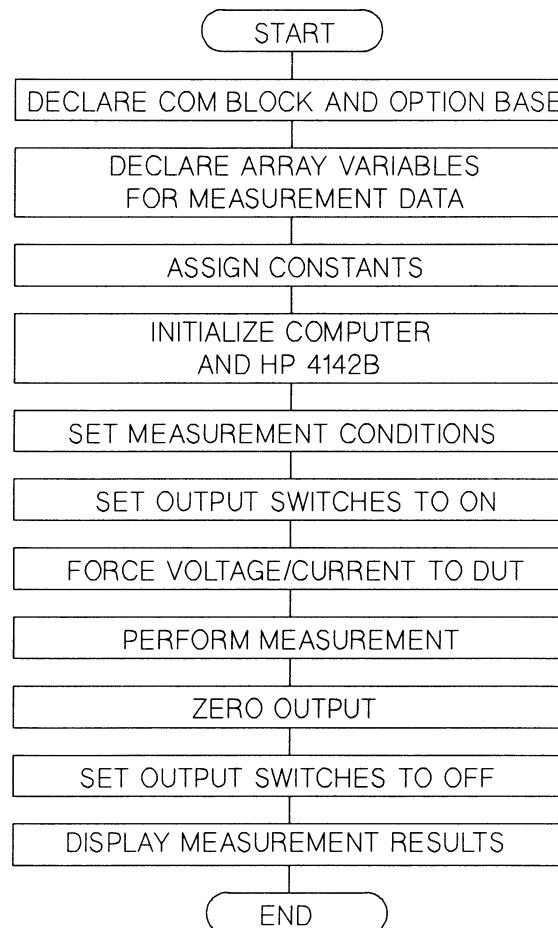
Before you begin to program your HP 4142B, the control software requires the following BIN files:

Driver	HPIB
Language	GRAPH, GRAPHX, IO, MAT, PDEV, KBD, ERR
Extensions	

Programming Procedures

Measurement Program Flow

Following figure shows the basic flow of a typical measurement program.



Measurement Program Flow Chart

Collector Characteristics Example Program

The following figure shows an example program for measuring the collector characteristics of a bipolar transistor. This program follows the program flow shown in the above figure, and is fully executable when linked with the appropriate control software subprograms. Descriptions for each program section are contained in succeeding paragraphs. Refer to Chapter 4 for complete subprogram information.

```
10  ! Bipolar Tr Ic-Vc Measurement using Sweep Function
20  !
30  !  LOADSUB ALL FROM "HP4142_DRV"
40  !  LOADSUB ALL FROM "GRAPHICS"
50  !
60  OPTION BASE 1
70  ASSIGN @Hp4142 TO 717
80  COM @Hp4142
90  !
100 DIM Vc(101), IC(101)
110 INTEGER Collector, Base, Emitter, Point
120 Collector=2
130 Base=3
140 Emitter=4
150 Vstart=0
160 Vstop=1
170 Point=101
180 Ib=1.0E-5
190 Ve=0
200 !
210 Init_computer
220 Init_hp4142
230 Set_smu(1)
240 !
250 Ch_sw_on
260 Force_i(Base, Ib, 0, 5)
270 Force_v(Emitter, Ve, 0, 1.00E-1)
280 Set_iv(Collector, 1, 0, Vstart, Vstop, Point, 0, 0, .01)
290 Sweep_iv(Collector, 2, 0, Ic(*), Vc(*))
300 Zero_output
310 Ch_sw_off
320 !
330 Lingraph(Vstart, Vstop, 0, Ic(Point)*1000, "Collector
    Voltage (V)", "Collector Current (mA)", "Ic-Vc Curve",1)
340 FOR I=1 TO Point
350     DRAW Vc(I),IC(I)*1000
360 NEXT I
370 !
380 END
```

Figure 2-1. Example Program for Measuring Collector Characteristics

Declare Com Block and Option Base

In the HP 4142B's control software, subprograms communicate with the main program and with other subprograms via a COM block. It is necessary, therefore, to declare a COM block at the beginning of your measurement program. If you want the default lower array boundary to be 1, instead of 0, include the OPTION BASE 1 statement at the beginning of your measurement program. Program lines 60 - 80 declare a COM block and OPTION BASE 1.

```
60  OPTION BASE 1           where 717 is the HP 4142B's HP-IB address
70  ASSIGN @Hp4142 TO 717
80  COM @Hp4142
```

Declare Array Variables for Measurement Data

The DIM statement dimensions and reserves array variable memory for measurement data. In this example, the lower array boundary is set to 1 because the default was set to 1 by the OPTION BASE 1 statement.

```
100 DIM Vc (101), Ic (101)
```

Assign Constants

Program lines 110 to 190 assign constants to the program variables that will be used as parameters for the subprograms.

```
110 INTEGER Collector,Base,  declares variables as type integer
    Emitter,Point
120 Collector=2              assigns channel#2 SMU to the collector
130 Base=3                  assigns channel#3 SMU to the base
140 Emitter=4               assigns channel#4 SMU to the emitter
150 Vstart=0                sets the sweep start voltage value to 0 V
160 Vstop=1                 sets the sweep stop voltage value to 1 V
170 Point=101              specifies the number of measurement points
180 Ib=1.0E-5              sets the base current value to 1.0E-5A
190 Ve=0                   sets the emitter voltage value to 0 V
```

Initialize Computer and HP 4142B

The Init_computer and Init_hp4142 control software subprograms initialize the display of the computer and the HP 4142B prior to measurement execution.

```
210 Init_computer
220 Init_hp4142
```

When using these subprograms in your measurement programs, be sure they precede other control software subprograms.

Set Measurement Conditions

The Set_smu subprogram sets the number of samples that will be taken and averaged for each measurement.

```
230 Set_smu (1)      sets the number of samples for averaging to 1
```

Set Output Switches to ON

The Ch_sw_on subprogram sets the output switch of specified SMUs, HVUs, HCU, and VSs to ON, thus enabling output.

```
250 Ch_sw_on
```

Line 250 sets the output switch of all SMUs, HVUs, HCU, and VSs to ON.

Force Voltage/Current

The Force_i and Force_v subprograms force the specified current and voltage. Voltage can be forced from SMUs, HVU or VSs; current can be forced from SMUs or HVUs.

```
260 Force_i (Base, Ib, 0, 5)
270 Force_v (Emitter, Ve, 0, 1.00E-1)
```

In line 260, the specified current, *Ib* (line 180), is forced to the *Base* (from the *ch#3* SMU—line 130) with AUTO ranging, and *V compliance* set to 5 V.

In line 270, the specified voltage, *Ve* (line 190), is forced to the *Emitter* (from the *ch#4* SMU—line 140) with AUTO ranging, and *I compliance* set to 100 mA.

Perform Measurement

The Set_iv subprogram sets the parameters for a staircase sweep measurement. The Sweep_iv subprogram triggers a staircase sweep measurement in accordance with the parameters specified in the Set_iv program and returns the measurement results.

```
280 Set_iv (Collector, 1, 0, Vstart, Vstop, Point, 0, 0, .01)
290 Sweep_iv (Collector, 2, 0, Ic ( * ), Vc ( * ))
```

In line 280, the source unit (*ch#2* SMU—line 120) is connected to the *Collector*, and set for a linear voltage sweep with AUTO ranging, 0 s *hold time*, 0 s *delay time*, and 10 mA *I compliance*.

In line 290, the voltage sweep is triggered, and the collector current is measured at each step with AUTO ranging. Measurement data and source data are returned to *Ic (*)* and *Vc (*)*, respectively.

Zero Output

Line 300 memorizes the present settings for all modules, then sets all modules to 0 V output.

```
300 Zero_output
```

Set Output Switches to OFF

Line 310 sets the output switch of all SMUs, HVUs and VSs to OFF.

```
310 Ch_sw_off
```

Display Measurement Results

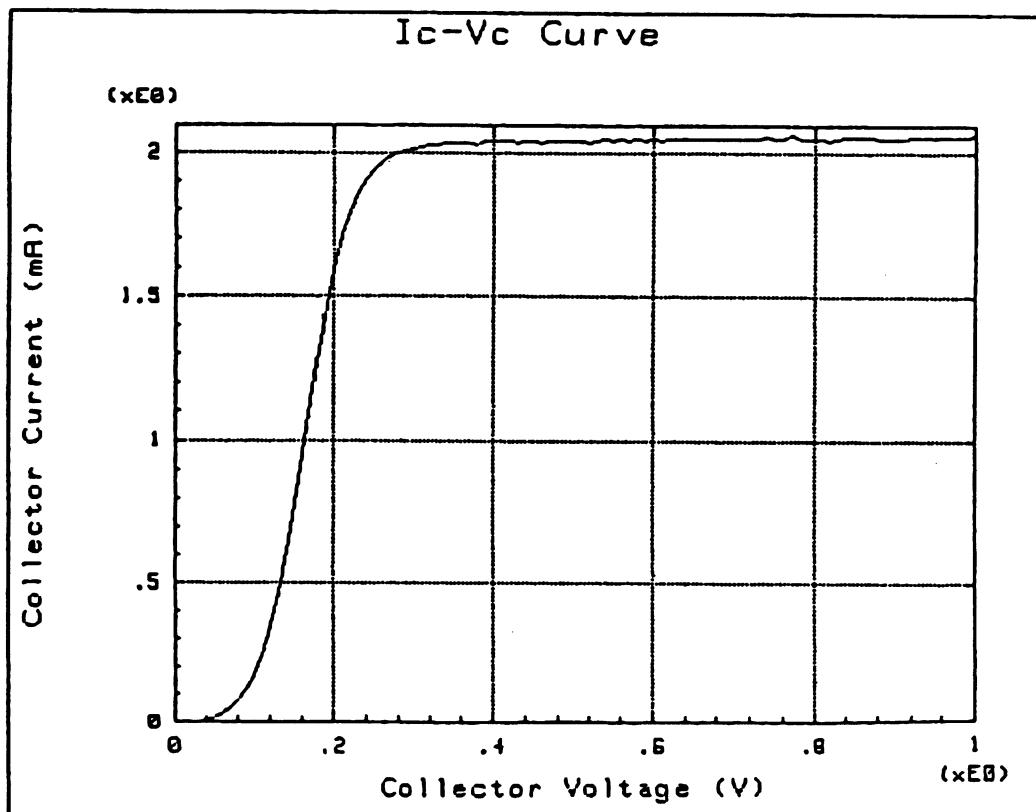
Line 330 draws, numbers, and labels the axes for a linear X-Y graph.

```
330 Lingraph (Vstart, Vstop, 0, Ic(Point) * 1000, "Collector Voltage  
(V)", "Collector Current (mA)", "Ic-Vc Curve", 1)
```

Lines 340 to 360 draws the measurement data in the graph created by line 330.

```
340 FOR I=1 TO Point  
350   DRAW Vc ( I ), Ic ( I ) * 1000  
360 NEXT I
```

The following figure shows example output from this measurement program.



Example Output from Measurement Program

Executing Measurement Programs

To use the HP 4142B's control software, you must link the required control software subprograms to your measurement program using HP BASIC's LOADSUB statement. To execute the example program, you'll need to link it to the HP4142_DRV and GRAPHICS files by removing the comment symbols (!) from lines 30 and 40.

Control Software Description

Introduction

This chapter provides a brief description of each control software subprogram and user-defined function, and gives examples of the types of measurements that can be made using the control software.

The HP4142_DRV subprograms let you perform High Speed Spot, Staircase Sweep, 1-ch Pulsed Spot, Pulsed Sweep, Staircase Sweep with Pulsed Bias, 2-ch Pulsed Spot, Pulsed Sweep with Pulsed Bias, Analog Search, Quasi-Pulsed measurements.

The user-defined functions in PARA_4142 facilitate measurements of some of the more common dc semiconductor parameters, and the utility subprograms in GRAPHICS facilitate displaying measurement results in an X-Y graphics format.

Subprogram Descriptions

The following tables list the control software subprograms and user-defined functions contained in the HP4142_DRV, GRAPHICS, and PARA_4142 files, respectively, and give a brief description of each.

HP4142_DRV Subprograms

Subprogram	Description
Auto_cal	Enables/disables the Auto-Calibration function.
Cal_hp4142	Performs self-calibration.
Ch_sw_on	Sets SMU/HVU/HCU/VS output switches to ON.
Ch_sw_off	Sets SMU/HVU/HCU/VS output switches to OFF.
Zero_output	Sets SMU/HVU/HCU/VS output to 0 V.
Recover_output	Resets SMU/HVU/HCU/VS output from Zero Output to previous settings.
Set_pol	Sets the polarity of the HVU output.
Set_smu	Sets number of samples for averaging.
Set_vm	Sets VM operation mode.
Init_hp4142	Resets the HP 4142B.
Self_test	Initiates Self-Test and displays results.
Force_i	Forces specified current from a specified SMU or HVU.
Force_v	Forces specified voltage from a specified SMU, HVU or VS.
Measure_i	Performs high speed spot I measurement using a specified SMU, HVU or VS.
Measure_v	Performs high speed spot V measurement using a specified SMU, HVU or VM.
Set_iv	Sets staircase sweep measurement parameters.
Sweep_iv	Performs staircase sweep measurements.
Sweep_miv	Performs multichannel staircase sweep measurements.
Sweep_mode	Sets miscellaneous sweep operation parameters.
Pulse_i	Sets a specified SMU, HVU or HCU to pulsed current source mode.
Pulse_v	Sets a specified SMU, HVU, HCU or VS to pulsed voltage source mode.
Dpulse_i	Sets a specified SMU or HCU to pulsed current source mode.
Dpulse_v	Sets a specified SMU, HCU or VS to pulsed voltage source mode.
Pulse_measure	Performs 1-ch pulsed spot measurement.
Dpulse_measure	Performs 2-ch pulsed spot measurement.
Set_piv	Sets pulsed sweep measurement parameters.

3-2 Control Software Description

Sweep_piv	Performs pulsed sweep measurements.
Sweep_pbias	Performs staircase sweep with pulsed bias measurements.
Dsweep_piv	Performs pulsed sweep with pulsed bias measurements.
Para_hfe	Calculates parameters required for h_{FE} measurement using AFU.
Para_vth	Calculates parameters required for V_{TH} measurement using AFU.
Set_asource	Sets search SMU parameters for an analog search measurement.
Set_amonitor	Sets sense SMU parameters for an analog search measurement.
Measure_asearch	Performs analog search measurement.
Set_bdm	Sets quasi-pulsed measurement parameters.
Measure_bdm	Performs quasi-pulsed measurement.
Connect_relay1	Sets the output of the control pins for module selector.
Connect_relay2	Sets the output of the control pins for external relay control.

GRAPHICS Subprograms

Subprogram	Description
Wbuild_file	Generates BSDM format measurement data file.
Lingraph	Generates linear-linear X-Y axes and grid pattern.
Loggraph	Generates linear-log X-Y axes and grid pattern.
Dump_screen	Performs a screen dump to a printer.
Init_computer	Initializes the computer and display.

PARA_4142 User-Defined Functions

Function	Description
FNHfe	Uses AFU to measure forward current transfer ratio (h_{FE}) of bipolar transistors.
FNBvbo	Measures collector-base breakdown voltage.
FNBvces	Measures collector-emitter breakdown voltage.
FNIds	Measures drain current for a specified drain and gate voltage.
FNVth1	Measures gate-source threshold voltage of an enhancement-type FET when a specified drain current is forced.
FNVth2	Measures gate-source voltage at two different drain currents and calculates V_{TH} using linear extrapolation.
FNVth3	Uses AFU to measure V_{TH} .
FNR_measure	Performs a two-terminal or four terminal (Kelvin connection) resistance measurement.

High Speed Spot Measurements

For high speed spot measurements, the control software provides the Force_i, Force_v, Measure_i, and Measure_v subprograms. Force_i specifies an SMU as a constant current source, and sets and outputs the specified current. Force_v specifies an SMU or VS as a constant voltage source, and sets and outputs the specified voltage. Measure_i specifies an SMU or VS as a measurement unit, and performs a high speed spot current measurement. Measure_v specifies an SMU or VM as a measurement unit, and performs a high speed spot current measurement. The following is an example high speed spot measurement program:

```
10  ASSIGN @Hp4142 TO 717
20  COM @Hp4142
30  INTEGER B_ch, C_ch
40  !                               Emitter      : GNDU
50  B_ch=3                         ! Base        : Ch#3
60  C_ch=2                         ! Collector   : Ch#2
70  Ib=2.E-5
80  Vc=2
90  !
100 Init_hp4142
110 Ch_sw_on
120 Force_i (B_ch, Ib, 0, 2)
130 Force_v (C_ch, Vc, 0, 1.E-2)
140 Measure_i (C_ch, Imeas)
150 Zero_output
160 Ch_sw_off
170 END
```

120: Forces current from ch#3 unit.

130: Forces voltage from ch#2 unit.

140: Measures current at ch#2 unit, and stores result in Imeas.

Staircase Sweep Measurements

For staircase sweep measurements, the control software provides the Set_iv, Sweep_iv, Sweep_miv, and Sweep_mode subprograms. Set_iv specifies an SMU or VS as a staircase sweep source and establishes sweep parameters. Sweep_iv specifies an SMU or VM as a measurement unit, triggers the units, and performs single channel measurement using the specified measurement unit. Sweep_miv specifies multiple SMUs or VMs as measurement units, triggers the units, and performs multichannel measurements using the specified measurement units. Sweep_mode lets you set the automatic sweep abort function and output after sweep parameters. The following is an example staircase sweep measurement program:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER B_ch, C_ch, Vc_no_step
50  DIM Imeas(101)
60  !                               Emitter      : GNDU
70  B_ch=3                          ! Base        : Ch#3
80  C_ch=2                          ! Collector   : Ch#2
90  Vc_start=0
100 Vc_stop=1
110 Vc_no_step=1-1
120 Ic_comp=.01
130 Ib=1.E-5
140 !
150 Init_hp4142
160 Ch_sw_on
170 Set_iv (C_ch, 1, 0, Vc_start, Vc_stop, Vc_no_step, 0, 0,
    Ic_comp)
180 Sweep_mode (2,2)
190 Force_i (B_ch, Ib, 0, 2)
200 Sweep_iv (C_ch, 2, 0, Imeas(*))
210 Zero_output
220 Ch_sw_off
230 END
```

170: Sets ch#2 unit as a staircase voltage sweep source.

180: Sets sweep conditions.

200: Triggers a staircase voltage sweep at ch#2 unit, measures current at ch#2 unit for each sweep step, and stores the results in *Imeas(*)*.

1-ch Pulsed Spot Measurements

For 1-ch pulsed spot measurements, the control software provides the `Pulse_v`, `Pulse_i`, and `Pulse_measure` subprograms. `Pulse_i` specifies an SMU or HCU as a pulsed current source. `Pulse_v` specifies a SMU, HCU, or VS as a pulsed voltage source and establishes pulse parameters. `Pulse_measure` specifies an SMU, HCU, or VM as a measurement unit, triggers the units, and performs a single channel measurement using the specified measurement unit. The following is an example pulsed spot measurement program:

```
10  ASSIGN @Hp4142 TO 717
20  COM @Hp4142
30  INTEGER B_ch, C_ch
40  !                               Emitter      : GNDU
50  B_ch=3                         ! Base        : Ch#3
60  C_ch=2                         ! Collector   : Ch#2
70  Ib=5.E-3
80  Ic=5.E-2
90  !
100 Init_hp4142
110 Ch_sw_on
120 Pulse_i (B_ch, 0, 0, Ib, 1.E-3, 0, 1.E-1, 2)
130 Force_i (C_ch, Ic, 0, 2)
140 Pulse_measure (C_ch, 1, Vmeas,)
150 Zero_output
160 Ch_sw_off
170 END
```

120: Sets ch#3 unit as a pulsed current source.

140: Triggers a pulse current at ch#2 unit, measures voltage at ch#2 unit, and stores the result in *Vmeas*.

Pulsed Sweep Measurements

For pulsed sweep measurements, the control software provides Set_piv and Sweep_piv subprograms. Set_piv specifies an SMU, HCU, or VS as a pulsed sweep source and establishes sweep parameters. Sweep_piv specifies an SMU, HCU, or VM as a measurement unit, triggers the units, and performs a single channel measurement using the specified measurement unit.

The following is an example pulsed sweep measurement program:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER A_ch, No_step
50  DIM Imeas(101),Vsource(101)
60  !                               Cathode      : GNDU
70  A_ch=2                          ! Anode       : Ch#2
80  Pv_start=0
90  Pv_stop=.9
100 No_step=91
110 If_comp=.1
120 !
130 Init_hp4142
140 Ch_sw_on
150 Set_piv (A_ch, 1, 0, 0, Pv_start, Pv_stop, No_step, 1.E-3, 1.E-
160 Sweep_piv (A_ch, 2, 0, Imeas(*), Vsource(*))
170 Zero_output
180 Ch_sw_off
190 END
```

150: Sets ch#2 unit as a pulsed voltage sweep source.

160: Triggers a pulsed voltage sweep at ch#2 unit, measures current at ch#2 unit for each sweep step, and stores the results in *Imeas(*)*, and *Vsource(*)*.

Staircase Sweep With Pulsed Bias Measurements

For staircase sweep with pulsed bias measurements, the control software provides Pulse_v, Pulse_i, Set_iv, and Sweep_pbias subprograms. Set_iv specifies an SMU or VS as a staircase sweep source, and establishes sweep parameters. Pulse_i or Pulse_v specifies an SMU, HCU, or VS as a pulsed bias source, and establishes pulse parameters. Sweep_pbias specifies an SMU, HCU, or VM as a measurement unit, triggers the units, and performs a single channel measurement using the specified measurement unit. The following is an example staircase sweep with pulsed bias measurement program:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER B_ch, C_ch, Vc_no_step
50  DIM Imeas(101),Vsource(101)
60  !                               Emitter      : GNDU
70  B_ch=3                          ! Base        : Ch#3
80  C_ch=2                          ! Collector   : Ch#2
90  Vc_start=0
100 Vc_stop=20
110 Vc_no_step=101
120 Ic_comp=.1
130 Ib=3.E-4
140 !
150 Init_hp4142
160 Ch_sw_on
170 Pulse_i (B_ch, 0, 0, Ib, 1.E-3, 0, 5.E-2, 2)
180 Set_iv (C_ch, 1, 0, Vc_start, Vc_stop, Vc_no_step, 0, 0,
    Ic_comp)
190 Sweep_iv (C_ch, 2, 0, Imeas(*),Vsource(*))
200 Zero_output
210 Ch_sw_off
220 END
```

170: Sets ch#3 unit as a pulsed current bias source.

180: Sets ch#2 unit as a staircase voltage sweep source.

190: Triggers a staircase voltage sweep at ch#2 unit, and for each sweep step, the pulsed current bias is forced to ch#1 unit and the current is measured at ch#2 unit. Results are stored in *Imeas(*)* and *Vsource(*)*.

2-ch Pulsed Spot Measurements

For 2-ch pulsed spot measurements, the control software provides `Pulse_v`, `Pulse_i`, `Dpulse_v`, `Dpulse_i`, `Set_iv`, and `Dpulse_measure` subprograms. `Pulse_i` or `Pulse_v` specifies an SMU or HCU as one pulsed source, and establishes pulse parameters. `Dpulse_i` or `Dpulse_v` specifies an SMU or HCU as the other pulsed source, and establishes pulse parameters. At least one HCU is required. `Dpulse_measure` specifies an SMU, HCU, or VM as a measurement unit, triggers the units, and performs a single channel measurement using the specified unit. The following is an example 2-ch pulsed spot measurement program:

```
10  ASSIGN @Hp4142 TO 717
20  COM @Hp4142
30  INTEGER B_ch, C_ch
40  !                               Emitter      : GNDU
50  B_ch=2                          ! Base       : HPSMU (Ch#2)
60  C_ch=5                          ! Collector  : HCU (Ch#5)
70  Ib=1
80  Ic=10
90  !
100 Init_hp4142
110 Ch_sw_on
120 Pulse_i (B_ch, 0, 0, Ib, 1.E-4, 0, 1.E-3, 2)
130 Dpulse_i (C_ch, 0, 0, Ic, 5)
140 Dpulse_measure (C_ch, 1, Vce)
150 Zero_output
160 Ch_sw_off
170 END
```

120: Sets ch#2 unit as a pulsed current source.

130: Sets ch#5 unit as a pulsed current source

140: Triggers a pulse current at ch#5 unit, measures voltage at ch#5 unit, and stores the result in *Vce*.

Pulsed Sweep With Pulsed Bias Measurements

For pulsed sweep with pulsed bias measurements, the control software provides Dpulse_i, Dpulse_v, Set_piv, and Dsweep_piv subprograms. Set_piv specifies an SMU or HCU as the pulsed sweep source, and establishes sweep parameters. Dpulse_i or Dpulse_v specifies an SMU or HCU as the pulse bias source, and establishes pulse parameters. At least one HCU is required. Dsweep_piv specifies an SMU, HCU, or VM as a measurement unit, triggers the units, and performs a single channel measurement using the specified measurement unit. The following is an example pulsed sweep with pulsed bias measurement program:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER B_ch, C_ch, Vc_no_step
50  DIM Ic(100)
60  !                               Emitter      : GNDU
70  B_ch=2                          ! Base        : HPSMU (Ch#2)
80  C_ch=5                          ! Collector  : HCU (Ch#5)
90  Vc_start=.1
100 Vc_stop=10
110 Vc_no_step=100
120 Ic_comp=10
130 Ib=5.E-2
140 !
150 Init_hp4142
160 Ch_sw_on
170 Set_piv (C_ch, 1, 0, 0, Vc_start, Vc_stop, Vc_no_step, 2.E-4,
    2.E-2, 0, Ic_comp)
180 Dpulse_i (B_ch, 0, 0, Ib, 2)
190 Dsweep_piv (C_ch, 2, 0, Ic(*))
200 Zero_output
210 Ch_sw_off
220 END
```

170: Sets ch#5 HCU as a pulsed voltage sweep source.

180: Sets ch#2 HPSMU as a pulsed current bias source.

190: Triggers a pulsed voltage sweep at ch#5 HCU, and for each sweep step, the pulsed current bias is forced to ch#2 HPSMU and the current is measured at ch#5 HCU. Results are stored in *Ic(*)*.

Analog Search Measurements

For analog search measurements, the control software provides the Para_hfe, Para_vth, Set_asource, Set_amonitor, and Measure_asearch subprograms. Para_hfe and Para_vth calculate and return optimized variables for performing h_{FE} and V_{TH} measurements using the AFU. Set_asource and Set_amonitor set analog search parameters, and Measure_asearch triggers the analog search, and returns measurement values. The following is an example analog search measurement program:

```
10  ASSIGN @Hp4142 TO 717
20  COM @Hp4142
30  INTEGER B_ch, C_ch, Status
40  !                               Emitter      : GNDU
50  B_ch=3                         ! Base        : Ch#3
60  C_ch=2                         ! Collector   : Ch#2
70  Vb_start=0
80  Vb_stop=1
90  Vb_rate=200
100 Ib_comp=1.15E-4
110 Vc=1
120 Ic_target=1.E-3
130 Ic_comp=1.15E-3
140 Integ_time=4.5E-4
150 Delay_time=1.E-4
160 !
170 Init_hp4142
180 Ch_sw_on
190 Set_asource (B_ch, Vb_start, Vb_stop, Vb_rate, 0, Delay_time,
    Ib_comp)
200 Set_amonitor (C_ch, 1, Vc, Ic_target, Ic_comp)
210 Measure_asearch (1, 4, Integ_time, Search_data, Sense_data,
    Status)
220 Zero_output
230 Ch_sw_off
240 END
```

190: Sets ch#3 SMU as a search SMU.

200: Sets ch#2 SMU as a sense SMU.

210: Triggers the analog search, and stores results in *Search_data* and *Sense_data*.

Quasi-Pulsed Measurements

For quasi-pulsed measurements, the control software provides the Set_bdm and Measure_bdm subprograms.

Set_bdm specifies an HVU or SMU as a quasi-pulsed source and establishes parameters for quasi-pulsed measurements.

Measure_bdm specifies an HVU or SMU as a measurement unit, triggers the units, and performs quasi-pulsed measurements.

The following is an example quasi-pulsed measurement program:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER C_ch,Status
50  !
60  C_ch=2
70  Vrange=1000
80  Vstart=500
90  Vstop=1000
100 Hold_time=0
110 Delay_time=0
120 Icomp=1.0E-3
130 !
140 Init_hp4142
150 Ch_sw_on
160 Set_bdm(C_ch,Vrange,Vstart,Vstop,Hold_time,Delay_time,Icomp)
170 Measure_bdm (C_ch,0,0,Bvces,Status)
180 Zero_output
190 Ch_sw_off
200 PRINT "BVces="; Bvces
210 END
```

160: Sets ch#2 unit as a quasi-pulsed source.

170: Triggers a quasi-pulsed voltage at ch#2 unit, measures voltage at ch#2 unit, and stores the result in *Bvces*.

Parameter Measurements Using Para_4142 Functions

The PARA_4142 file contains user-defined functions that simplify the measurement of common dc semiconductor parameters, thereby minimizing programming requirements.

PARA_4142 contains the FNBvco, FNBvces, FNHfe, FNIds, FNVth1, FNVth2, and FNVth3 functions for measuring transistor dc parameters, and the FNR_measure function for performing resistance measurements. The following is an example h_{FE} measurement program using the FNHfe function:

```
10  OPTION BASE 1
20  ASSIGN @Hp4142 TO 717
30  COM @Hp4142
40  INTEGER Channel(3)
50  Channel(1)=1           ! Collector      : Ch#1 SMU
60  Channel(2)=2           ! Base          : Ch#2 SMU
70  Channel(3)=3           ! Emitter      : Ch#3 SMU
80  Vc=5
90  Ic=1.E-3
100 Vb_start=0
110 Vb_stop=5
120 Hfe=FNHfe(Channel(*), Vc, Ic, Vb_start, Vb_stop)
130 END
```

120: Performs measurements and calculates h_{FE} .

Programming Reference

Introduction

This chapter contains an alphabetical listing of the HP 4142B Control Software subprograms that can be called in user-written programs. Each entry lists the subprogram name and the file that contains the subprogram, shows the syntax of the calling context, explains pass parameters, and gives example statements, as shown below.

- | | | |
|---|---|---|
| 1 | — | Auto_cal |
| 2 | — | File: HP4142_DRV |
| 3 | — | This subprogram sets Auto-Calibration ON or OFF. |
| 4 | — | HP-IB Command:
CM |
| 5 | — | Syntax

Auto_cal (<i>auto calibration</i>) |
| 6 | — | Parameters

▪ <i>auto calibration</i> (I/O: I, type: integer)
0: Auto-Calibration OFF
1: Auto-Calibration ON |
| 7 | — | Example Statements

Auto_cal (1)
Auto_cal (0) |

1. Subprogram name
2. Name of the Control Software File that contains this subprogram
3. Subprogram description
4. Name of the HP-IB command that is used in this subprogram

5. Syntax of the calling context
6. Pass parameter explanation
7. Example statements

Conventions

The following conventions are used throughout this chapter.

Reading the Syntax

Required parameters, in which you must substitute a value or variable, are shown in *standard italics* in the syntax calling context.

Optional parameters, in which you may substitute a value or variable, are shown in *standard italics* and are delimited by brackets [] in the syntax calling context.

Most optional parameters have default values assigned. The default values are listed in each parameter explanation.

In the following example, the *range* and *status* are both optional parameters, and the *status* parameter cannot be specified unless the range parameter is specified.

```
Measure_v ( measurement ch#, voltage value name [, range] [, status] )
```

CALL Statement

Except for the following three cases, the CALL statement can be omitted when calling the subprogram:

- If the subprogram is called from the keyboard.
- If the subprogram is called after the THEN keyword in an IF statement.
- In an ON [event] CALL statement.

Parameters

The parameters are written in standard italics. After the parameter, the following information is enclosed in parentheses.

I/O	If an I is indicated here, you can pass a parameter by value or by reference. The I parameter is used to pass actual values required by the subprogram. If an O is indicated here, you can pass a parameter only by reference. The O parameter is used to store any values returned by the subprogram.
type	Each parameter has a type (integer, real, string, or numeric array). If the type of the parameter is integer, use the INTEGER command to declare a variable for the parameter. If the type of the parameter is string or numeric array, use the DIM or REAL command to define the array as the parameter.
unit	If a parameter has a unit (V for voltage, I for current, etc.), the unit is indicated here.

Note

Pass by reference—the calling context actually gives the subprogram access to the calling context’s value area (which is essentially access to the calling context’s variable).

Auto_cal

File: HP4142_DRV

This subprogram sets Auto-Calibration ON or OFF.

HP-IB Command:

CM

Syntax

Auto_cal (*auto calibration*)

Parameters

- *auto calibration* (I/O: I, type: integer)

0: Auto-Calibration OFF

1: Auto-Calibration ON

Example Statements

Auto_cal (1)

Auto_cal (0)

Cal_hp4142

File: HP4142_DRV

This subprogram performs Self-Calibration.

HP-IB Command:

CA

Syntax

```
Cal_hp4142 ( [ slot# ] )
```

Parameters

- *slot#* (I/O: I, type: integer)

The *slot#* parameter selects a unit to be calibrated. If you specify the *slot#* parameter, the unit that is installed in the specified slot is calibrated.

Unit	ch#
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VS/VMU	1 to 8
AFU	1 to 8

Default= all units from slot#1 to #8.

Example Statements

```
Cal_hp4142
```

```
Cal_hp4142 (4)
```

Ch_sw_off

File: HP4142_DRV

This subprogram disables a specified unit by setting the output switches to OFF.

HP-IB Command:

CL

Syntax

`Ch_sw_off ( [ ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] )`

Parameters

- *ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

Default = all units

Example Statements

`Ch_sw_off (1)`

`Ch_sw_off (Ch1,Ch2,Ch3)`

`Ch_sw_off`

Ch_sw_on

File: HP4142_DRV

This subprogram enables a specified unit by setting the output switches to ON.

HP-IB Command:

CN

Syntax

```
Ch_sw_on ( [ ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] [ , ch# ] )
```

Parameters

- *ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HCU	2 to 8
VS	1 to 8, 11 to 18, or 21 to 28

Default = all units

Example statements

```
Ch_sw_on (1)
```

```
Ch_sw_on (Ch1,Ch2,Ch3)
```

```
Ch_sw_on
```

Connect_relay1

File: HP4142_DRV

This subprogram controls the output of the CONTROL connector pins for the Module Selector control.

Note



Before controlling the output of the CONTROL connector pins, all source unit outputs are set to zero (the same conditions after a Zero_output execution). After controlling the output of the CONTROL connector pins, all outputs of the source units are returned to the output states previous to the execution of this subprogram.

HP-IB Command:

ERC

Syntax

`Connect_relay1 ( control value )`

Parameters

■ *control value* (I/O: I, type: integer)

- 0: Disconnects all units
- 1: Connects SMU
- 2: Connects HVU
- 3: Connects HCU

Example Statement

`Connect_relay1(1)`

Connect_relay2

File: HP4142_DRV

This subprogram controls the output of the CONTROL connector pins for the 16 bit external relay control.

HP-IB Command:

ERC

Syntax

```
Connect_relay2 (dry switching [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ]
[ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ] [ , pin# ]
[ , pin# ] [ , pin# ] [ , pin# ] )
```

Parameters

■ *dry switching* (I/O: I, type: integer)

0: Dry switching on

The HP 4142B automatically sets all outputs of source units to zero (same as the conditions after the Zero_output execution), and changes the outputs of the CONTROL connector pins. Then the outputs are returned to the output states previous to subprogram execution.

1: Dry switching off

Without changing the source unit outputs, the HP 4142B changes the outputs of the CONTROL connector pins.

■ *pin#* (I/O: I, type: integer)

The pin number to be set to LOW. Unspecified pins are set to HIGH.

1 to 16

Example Statement

```
Connect_relay2(0,1,3,5,7)
```

Dpulse_i

File: HP4142_DRV

This subprogram specifies an SMU or HCU as the pulsed current source and specifies the parameters for 2-ch pulsed spot or pulsed sweep with pulsed bias measurements.

HP-IB Command:

PDI

Syntax

For SMUs:

Dpulse_i (*output ch#*, *I output range*, *base current*, *pulse current* [, *V compliance*])

For HCU:

Dpulse_i (*output ch#*, *I output range*, *base current*, *pulse current*, *V compliance*)

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8

- *I output range* (I/O: I, type: real)

0: Auto ranging
1E-8: 10 nA Limited Auto ranging
1E-7: 100 nA Limited Auto ranging
1E-6: 1 μ A Limited Auto ranging
1E-5: 10 μ A Limited Auto ranging
1E-4: 100 μ A Limited Auto ranging
1E-3: 1 mA Limited Auto ranging
1E-2: 10 mA Limited Auto ranging
1E-1: 100 mA Limited Auto ranging
1: 1 A Limited Auto ranging
10: 10 A Limited Auto ranging

- *base current* (I/O: I, type: real, unit: A)
pulse current (I/O: I, type: real, unit: A)

Unit	<i>base current</i> ¹	<i>pulse current</i> ¹
HPSMU	0 to ± 1	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$	0 to $\pm 100\text{E}-3$
HCU	0 ²	0 to ± 10

¹ The *pulse current* and *base current* polarity must be the same.

² During base value output, the HCU output is 0 V and no current.

- *V compliance* (I/O: I, type: real, unit: V)

Unit	<i>V compliance</i>
HPSMU	0 to E200
MPSMU	0 to E100
HCU	0 to E10

Default:

- If the specified SMU is set to I source mode before the trigger:
Default = the setting before trigger
- If the specified SMU is set to V source mode before the trigger:
Default = none

Example Statements

```
Dpulse_i (1, 100E-6, 3E-5, 5E-5, 20)
```

```
Dpulse_i (3, 10E-6, 1E-7, 5E-6, 5)
```

Dpulse_measure

File: HP4142_DRV

This subprogram triggers a 2-ch pulsed spot measurement. A measurement is performed at the specified measurement *ch#*, and the measurement value is returned to a specified measurement variable.

HP-IB Command:

MM, FL, RV, RI, PDM, XE

Syntax

For Voltage Measurements:

```
Dpulse_measure ( measurement ch#, 1, measurement variable [ , primary pulse ch# ]  
[ , V measurement range ] [ , status ] )
```

For Current Measurements:

```
Dpulse_measure ( measurement ch#, 2, measurement variable [ , primary pulse ch# ]  
[ , I measurement range ] [ , status ] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VM	1 to 8, 11 to 18, or 21 to 28

- *measurement variable* (I/O: O, type: real, unit: V or A)

The measured value is returned to this parameter.

- *primary pulse ch#* (I/O: I, type: integer)

0: auto selection
2 to 8: for HCU *ch#*

When you use two HCUs as two pulse sources for the 2-ch pulsed spot measurements, specify this parameter to select the HCU in which the *pulse width* of the Pulse_i or Pulse_v subprogram is set. The pulse width of the other HCU is fixed to about 1 ms, and can not be specified. If you specify 0 as this parameter, the HCU that is set by the Dpulse_i or Dpulse_v subprogram is selected automatically.

When one pulse source is the SMU and the other pulse source is the HCU, the *pulse width* is always set to the HCU. In such a case, specify 0 for this parameter.

Default = 0

■ *V measurement range* (I/O: I, type: real)

The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of an SMU or HCU is set to the Compliance range automatically, regardless of the specified value.

0: 40 V range fixed
 2: 2 V range fixed
 20: 20 V range fixed
 40: 40 V range fixed

Default = 0

■ *I measurement range* (I/O: I, type: real)

0: Compliance range
 -1E-8: 10 nA range fixed
 -1E-7: 100 nA range fixed
 -1E-6: 1 μ A range fixed
 -1E-5: 10 μ A range fixed
 -1E-4: 100 μ A range fixed
 -1E-3: 1 mA range fixed
 -1E-2: 10 mA range fixed
 -1E-1: 100 mA range fixed
 -1: 1 A range fixed
 -10: 10 A range fixed

Default = 0

■ *status* (I/O: O, type: integer)

The measurement status information is returned to this parameter. See the status description in this chapter.

Example Statements

```
Dpulse_measure (1, 1, Voltage)
```

```
Dpulse_measure (Channel, 2, Current, 0, -1E-1, Status)
```

Dpulse_v

File: HP4142_DRV

This subprogram specifies an SMU or HCU as the pulsed V source and specifies its parameters for 2-ch pulsed spot or pulsed sweep with pulsed bias measurements.

HP-IB Command:

PDV

Syntax

`Dpulse_v ( output ch#, V output range, base voltage, pulse voltage [ , I compliance ] )`

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8

- *V output range* (I/O: I, type: real)

0: Auto ranging
2: 2 V Limited Auto ranging
20: 20 V Limited Auto ranging
40: 40 V Limited Auto ranging
100: 100 V Limited Auto ranging
200: 200 V Limited Auto ranging

- *base voltage* (I/O: I, type: real, unit: V)
pulse voltage (I/O: I, type: real, unit: V)

Unit	<i>base voltage</i>	<i>pulse voltage</i>
HPSMU	0 to ± 200	0 to ± 200
MPSMU	0 to ± 100	0 to ± 100
HCU	0	0 to ± 10

- *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HPSMU	$\pm 2\text{E}-9$ to ± 1
MPSMU	$\pm 2\text{E}-9$ to $\pm 100\text{E}-3$
HCU	$\pm 1\text{E}-6$ to ± 10

Default:

- If the specified SMU or HCU is set to V source mode before the trigger:

Default = the setting before trigger

- If the specified SMU or HCU is set to I source mode before the trigger:

Default = none

Example Statements

```
Dpulse_v (1, 20, 0, 5, 1E-4)
```

```
Dpulse_v (5, 40, -10, 30, 1E-3)
```

Dsweep_piv

File: HP4142_DRV

This subprogram triggers pulsed sweep with pulsed bias measurements, performs the measurements at the specified *measurement ch#* for each sweep step, then returns the measurement values and sweep source values to the *measurement value array(*)* and *source value array(*)*, respectively.

HP-IB Command:

MM, WNU?, FL, RV, RI, PDM, XE

Syntax

For Voltage Measurements:

```
Dsweep_piv ( measurement ch#, 1, V measurement range, measurement value array(*)  
[ , primary pulse ch#] [ , source value array(*)] )
```

For Current Measurements:

```
Dsweep_piv ( measurement ch#, 2, I measurement range, measurement value array(*)  
[ , primary pulse ch#] [ , source value array(*)] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VM	1 to 8, 11 to 18, 21 to 28

- *V measurement range* (I/O: I, type: real)

The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of SMUs or HCU is set to the Compliance range automatically, regardless of the specified value.

0: 40 V range fixed
2: 2 V range fixed
20: 20 V range fixed
40: 40 V range fixed

Default = 0

- *I measurement range* (I/O: I, type: real)

0: Compliance range
 -1E-8: 10 nA range fixed
 -1E-7: 100 nA range fixed
 -1E-6: 1 μ A range fixed
 -1E-5: 10 μ A range fixed
 -1E-4: 100 μ A range fixed
 -1E-3: 1 mA range fixed
 -1E-2: 10 mA range fixed
 -1E-1: 100 mA range fixed
 -1: 1 A range fixed
 -10: 10 A range fixed

Default = 0

- *measurement value array(*)* (I/O: O, type: numeric array, unit: V or A)

The measurement value of each sweep step is returned to this array. The number of elements of the *measurement value array(*)* must be larger than number of sweep steps.

- *primary pulse ch#* (I/O: I, type: integer)

0: auto selection
 2 to 8: for HCU ch#

When you use two HCU as two pulse sources for the pulsed sweep with pulsed bias measurement, specify this parameter to select the HCU in which the *pulse width* of the Pulse_i or Pulse_v subprogram is set. The pulse width of the other HCU is fixed to about 1 ms, and can not be specified. If you specify 0 as this parameter, the HCU that is set by the Dpulse_i or Dpulse_v subprogram is selected automatically.

When one pulse source is the SMU and the other pulse source is the HCU, the *pulse width* is always set to the HCU. In such a case, specify 0 for this parameter.

Default = 0

- *source value array(*)* (I/O: O, type: numeric array, unit: V or A)

The sweep source values of each sweep step is returned to this array. The number of elements of the *source value array(*)* must be larger than number of sweep steps.

Example Statements

```
Dsweep_piv (3, 1, 0, 2, Voltage(*), 3, Sweep_source(*))
```

```
Dsweep_piv (Ch, 2, -1, Current(*), 0, Source(*))
```

Note



When using this subprogram, it is recommended that you include OPTION BASE 1 at the beginning of your program. Including OPTION BASE 1 in your program causes array element numbering to start from 1 instead of 0. This makes it easier to keep track of the array elements in the *measurement value array(*)* and [*source value array(*)*] because the *n*th returned value is array element *n* instead of *n - 1*.

Dump_screen

File: GRAPHICS

This subprogram performs a screen dump to the printer. The printer's HP-IB address must be set to 01, and the select code must be 7.

Syntax

```
Dump_screen ( [ expand output ] )
```

Parameters

- *expand output* (I/O: I, type: integer)

Expanded output can be performed with this parameter.

0: Does not expand output

1: Expands output

Default = 0

Example Statements

```
Dump_screen
```

```
Dump_screen (1)
```

FNBvcbo

File: PARA_4142

This function measures the collector-base breakdown voltage of a bipolar transistor, and returns the measurement result to a user-specified variable.

Syntax

FNBvcbo (*channel*(*), *test current* [, *V compliance*] [, *hold time*])

Parameters

- *channel*(*) (I/O: I, type: integer array)

channel(*) is an integer array containing 1 to 3 elements as follows:

- *channel*(1) is the ch# of the SMU or HVU that must be connected to the collector. This SMU forces test current.
- *channel*(2) is the ch# of the SMU or GNDU that must be connected to the base.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected instead of an SMU, you must specify only one element for *channel*(*), or you must specify *channel*(2) = 99.

- *channel*(3) is the ch# of the SMU that must be connected to the emitter.

If an SMU is connected to the emitter, this function turns the output switch of the SMU to OFF.

If you only specify one or two elements for *channel*(*), the emitter must be opened.

- *test current* (I/O: I, type: real, unit: A)

The output current value that is forced from the *channel*(1) unit.

Unit	<i>test current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$

- *V compliance* (I/O: I, type: real, unit: V)

The voltage compliance value for the *channel*(1) unit.

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

FNBvcbo

Default = 20

- *hold time* (I/O: I, type: real, unit: s)

The wait time for a measurement by the *channel(1)* unit.

Default = 0.4

Example Statements

```
Vcbo = FNBvcbo (Channel(*), 1.E-5)
```

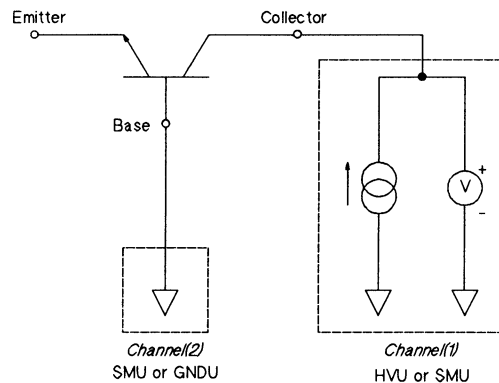
```
Vcbo = FNBvcbo (Channel(*), Ic, V_limit, Hold_time)
```

Semantics

When using this function, the units at *channel(*)* must be connected to the transistor as shown in the circuit below. The unit at *channel(1)* outputs *test current*, waits the *hold time*, and measures the transistor's collector-base voltage. The *hold time* is required to allow the current source to fully charge the output capacitance of the unit, and to allow the transistor to complete minority carrier generation.

The measurement result, which is the transistor's collector-base breakdown voltage, is returned to the user-specified variable.

Immediately after the measurement, this function sets all units to 0 V output, and sets all unit output switches to OFF.



Measurement Circuit

FNBvces

File: PARA_4142

This function measures the collector-emitter breakdown voltage of a bipolar transistor with the base and substrate both connected to the emitter. The function then returns the measurement result to a user-specified variable.

Syntax

FNBvces (*channel*(*), *test current* [, *V compliance*] [, *hold time*])

Parameters

- *channel*(*) (I/O: I, type: integer array)

channel(*) is an integer array containing 1 or 2 elements as follows:

- *channel*(1) is the ch# of the SMU or HVU that must be connected to the collector. This unit outputs *test current*.
- *channel*(2) is the ch# of the SMU or GNDU that must be connected to the base and emitter.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected, you must specify only one element for *channel*(*), or you must specify *channel*(2) = 99.

- *test current* (I/O: I, type: real, unit: A)

The output current value that is forced from the *channel*(1) unit.

Unit	<i>test current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$

- *V compliance* (I/O: I, type: real, unit: V)

The voltage compliance value for the *channel*(1) unit.

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

Default = 20

- *hold time* (I/O: I, type: real, unit: s)

FNBvces

The wait time for the measurement.

Default = 0.4

Example Statements

```
Vces = FNBvces (Channel(*), 1.E-3)
```

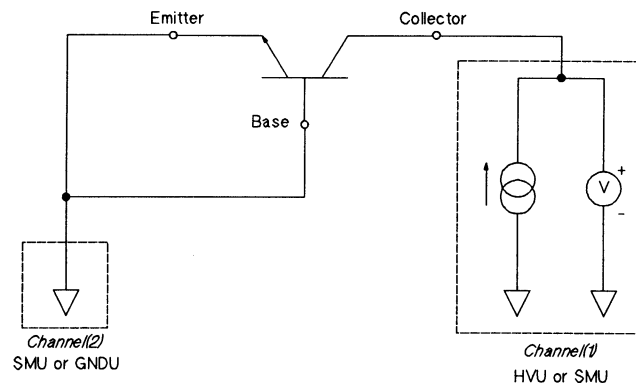
```
Vces = FNBvces (Channel(*), Ic, Vlimit, Hold_time)
```

Semantics

When using this function, the units at *channel(*)* must be connected to the transistor as shown in the circuit below. The unit at *channel(1)* outputs *test current*, waits the *hold time*, and measures the transistor's collector-emitter voltage. The *hold time* is required to allow the current source to fully charge the output capacitance of the unit, and to allow the transistor to complete minority carrier generation.

The measurement result, which is the transistor's collector-emitter breakdown voltage, is returned to the user-specified variable.

Immediately after the measurement, this function sets all units to 0 V output, and sets all unit output switches to OFF.



Measurement Circuit

FNHfe

File: PARA_4142

This function performs measurements and calculates the forward current transfer ratio (h_{FE}) of bipolar transistors by using an Analog Feedback Unit and two SMUs.

Syntax

FNHfe (*channel(*)*, *collector voltage*, *target collector current*, *start base voltage*, *stop base voltage* [, *ramp rate*] [, *feedback integration time*] [, *h_{FE} value*])

Parameters

- *channel(*)* (I/O: I, type: integer array)

channel()* is an integer array containing 2 or 3 elements as follows:

- *channel(1)* is the sense SMU ch#, and must be connected to the collector.
- *channel(2)* is the search SMU ch#, and must be connected to the base.
- *channel(3)* is the ch# of the unit that must be connected to the emitter.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected instead of an SMU, you must specify only two elements for *channel(*)*, or you must specify *channel(3)* = 99.

- *collector voltage* (I/O: I, type: real, unit: V)

The output voltage value that is forced from the *channel(1)* SMU (sense SMU).

Unit	<i>collector voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *target collector current* (I/O: I, type: real, unit: A)

The target current value that is sensed by the *channel(1)* SMU (sense SMU).

Unit	<i>target collector current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100E-3$

This function sets the sense SMU I compliance equal to $1.1 \times (\text{target collector current})$.

- *start base voltage* (I/O: I, type: real, unit: V)

The search start voltage value of the *channel(2)* SMU (search SMU).

FNHfe

Unit	<i>start base voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

■ *stop base voltage* (I/O: I, type: real, unit: V)

The search stop voltage value of the *channel(2)* SMU (search SMU).

Unit	<i>stop base voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

■ *ramp rate* (I/O: I, type: real, unit: V/s)

The ramp rate value of the *channel(2)* SMU (search SMU). The allowable values are 0.5 to 100000.

Default = 500

See the Set_asearch subprogram description.

■ *feedback integration time* (I/O: I, type: real, unit: s)

The feedback integration time for the analog feedback measurement. The allowable values are $0.5\text{E}-6$ to $450\text{E}-3$.

Default = 0.005

See the Measure_asearch subprogram description.

■ *h_{FE} value* (I/O: I, type: real)

The *h_{FE} value* is an expected value, and is used to calculate I compliance for the search SMU as shown in the following equation.

$$\text{I compliance} = 1.1 \times (\text{target collector current}) / h_{FE} \text{ value}$$

Default = 50

Example Statements

```
Hfe = FNHfe (Channel(*), 6, 1E-2, 0, 0.6)
```

```
Hfe = FNHfe (Channel(*), Vce, Ic, Vbmin, Vbmax)
```

```
Hfe = FNHfe (Channel(*), Vce, Ic, Vbmin, Vbmax, Ramp, Integ, hFE_value)
```

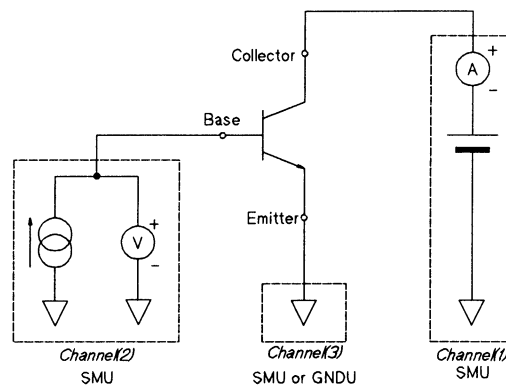
Semantics

When using this function, the SMUs and GNDU at *channel(*)* must be connected to the transistor as shown in the circuit below. The specified *collector voltage* polarity must be positive for NPN transistors, and negative for PNP transistors.

For this measurement, the function uses the AFU and two SMUs. The function uses the *channel(2)* SMU to vary the base voltage until the *channel(1)* SMU senses that the *target collector current* has been reached. Then, the function calculates h_{FE} by dividing the *target collector current* by the base current, and returns the result to the user-specified variable.

If an abnormal condition is detected (see the status description in this chapter), or if the *target collector current* cannot be reached, -9999999.9999 is returned to the user-specified variable.

Immediately after the measurement, this function sets all SMUs to 0 V output, and sets all SMU output switches to OFF.



Measurement Circuit

FNIds

File: PARA_4142

This function measures the drain current for a specified *drain voltage* and *gate voltage*, and returns the measurement result to a user-specified variable.

Syntax

```
FNIds ( channel(*), drain voltage, gate voltage [ , drain current compliance]
[ , substrate voltages] )
```

Parameters

■ *channel(*)* (I/O: I, type: integer array)

channel()* is an integer array containing 2 to 4 elements as follows:

- *channel(1)* is the ch# of the SMU or HVU that must be connected to the drain.
- *channel(2)* is the ch# of the SMU or HVU that must be connected to the gate.
- *channel(3)* is the ch# of the module that must be connected to the source.

If an SMU or HVU is connected, this function sets the module to 0 V output.

If a GNDU is connected instead of an SMU, you must specify only two elements for *channel(*)*, or you must specify *channel(3)* = 99.

- *Channel(4)* is the ch# of the SMU or HVU that must be connected to the substrate.

This function turns the SMU or HVU output switch to ON, and forces *substrate voltage* to the substrate ONLY IF *channel(*)* has four elements and *substrate voltage* is specified.

■ *drain voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(1)* unit.

Unit	<i>drain voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

■ *gate voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(2)* unit.

Unit	<i>gate voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

- *drain current compliance* (I/O: I, type: real, unit: A)

The current compliance for the *channel(1)* unit.

Unit	<i>drain current compliance</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$

Default = $1\text{E}-6$

- *substrate voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(4)* unit.

Unit	<i>substrate voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

Default = no connection

Example Statements

```
Ids = FNIds (Channel(*), 6, 0)
```

```
Ids = FNIds (Channel(*), Vds, Vgs, Id_limit, Vsub)
```

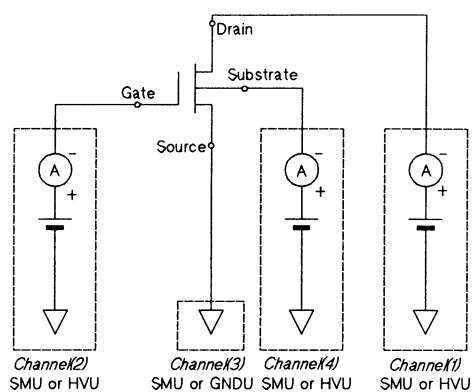
Semantics

When using this function, the SMUs, HVUs, and GNDU at *channel(*)* must be connected to the transistor as shown in the circuit below.

The measurement result obtained by the *channel(1)* SMU, which is the transistor's drain current, is returned to the user-specified variable. If an abnormal condition is detected (see the status description in this chapter), -9999999.99999 is returned to the user-specified variable.

Immediately after the measurement, this function sets all SMUs to 0 V output, and sets all SMU output switches to OFF.

FNlds



Measurement Circuit

FNR_measure

File: PARA_4142

This function uses either a two-terminal or a four-wire (Kelvin connection) method, and performs measurements to determine an unknown resistance. The measurement result is then returned to a user specified variable.

Syntax

FNR_measure (*H-force ch#*, *L-force ch#*, *H-sense ch#*, *L-sense ch#*, *measurement mode*, *test current* [, *V compliance*])

Parameters

- *H-force ch#* (I/O: I, type: integer)
L-force ch# (I/O: I, type: integer)

The *ch#* of the SMU or HVU that must be connected to one of the resistor's terminals.

- *H-sense ch#* (I/O: I, type: integer)
L-sense ch# (I/O: I, type: integer)

The *ch#* of the SMU, HVU or VM that must be connected to one of the resistor's terminals.

- *measurement mode* (I/O: I, type: integer)

Selects the resistance measurement mode.

- 1: 2-channel measurement
- 2: 4-channel measurement
- 3: differential measurement

- *test current* (I/O: I, type: real, unit: A)

This parameter specifies the following: *L-force* unit I output for measurement mode 1, *H-force* unit I output for measurement mode 2 or 3.

Unit	<i>test current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$

- *V compliance* (I/O: I, type: real, unit: V)

This parameter specifies the following: *L-force* unit V compliance for measurement mode 1, *H-force* and *H-sense* unit V compliance for measurement mode 2, and *H-force* unit V compliance for measurement mode 3.

FNR_measure

Unit	<i>V compliance</i>
HPSMU	0 to ±200
MPSMU	0 to ±100
HVU	0 to ±1000

Default = 20

Example Statements

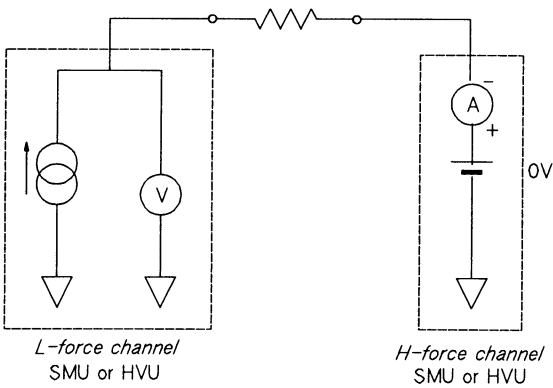
```
R = FNR_measure (1, 2, 3, 4, 2, 1.E-6)
```

```
Resistance = FNR_measure (H_force, L_force, H_sense,  
L_sense, Mode, Itest, Vlimit)
```

Semantics

■ Measurement mode = 1

Two SMUs or HVUs are used for the two-terminal resistance measurement. Refer to the circuit shown below.



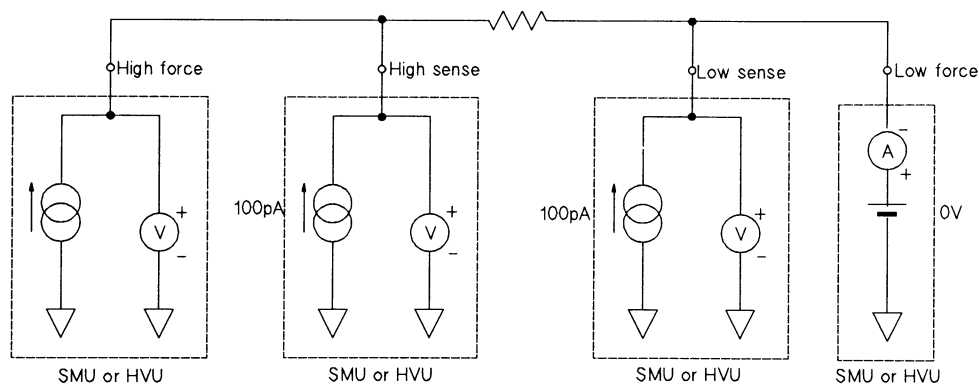
Measurement Circuit

These two units are specified by *H-force ch#* and *L-force ch#*. *H-sense ch#* and *L-sense ch#* are ignored.

This function sets the *H-force* unit to 0 V, and forces *test current* from the *L-force* unit. Then, the *H-force* unit measures current, and the *L-force* unit measures voltage. The function then calculates the unknown resistance using Ohm's law and these measurement values. This measurement result is returned to the user-specified variable.

■ Measurement mode = 2

Four SMUs or HVUs are used for the four-wire (Kelvin connection) resistance measurement. Refer to the circuit shown below.



Measurement Circuit

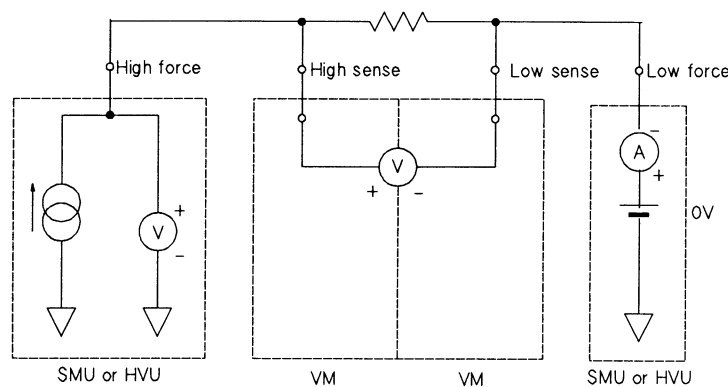
The units specified by *H-force ch#*, *L-force ch#*, *H-sense ch#*, and *L-sense ch#* must be connected as close to the unknown resistance as possible.

This function sets the units as shown in the circuit above, and forces *test current* from the *H-force* unit. Then, the function measures the *L-force* unit current, the *L-sense* unit voltage, and the *H-sense* unit voltage, and uses the equation below to calculate the unknown resistance. This measurement result is returned to the user-specified variable.

$$R_x = (L\text{-sense Voltage} - H\text{-sense Voltage}) / L\text{-force Current}$$

■ Measurement mode = 3

Two SMUs or HVUs and two VMs are used for this differential measurement. Refer to the circuit shown below.



Measurement Circuit

This function forces *test current* from the *H-force* unit. Then, the *L-force* unit measures current, and the *H-sense* VM and the *L-sense* VM measure the voltage drop across the unknown resistance. The function then uses Ohm's law and these measurement values to calculate the unknown resistance. This measurement result is returned to the user-specified variable.

FNR_measure

This function automatically includes a wait time (that is inversely proportional to the specified *test current*) to allow the SMU output capacitance and the DUT capacitance to fully charge before the measurement is made.

If a current measurement value is less than $9.E-7$ A ($9.E-10$ A for *measurement mode* = 1), or a voltage measurement value is less than $1.E-2$ V, or compliance is reached, then -9999999.99999 is returned to the user-specified variable. If *test current* is set to its upper limit (± 1 A), the *L-force* SMU may reach current compliance.

Immediately after the measurement, this function sets all modules to 0 V output, and sets all module output switches to OFF.

FNVth1

File: PARA_4142

This function measures the gate-source threshold voltage of an enhancement-type FET when a specified *drain current* is forced. The function then returns the measurement result to a user-specified variable. The threshold voltage of an enhancement-type FET is defined as the gate voltage required to cause a predetermined value of drain current to flow.

Syntax

FNVth1 (*channel*(*), *drain current* [, *gate voltage compliance*] [, *substrate voltage*])

Parameters

- *channel*(*) (I/O: I, type: integer array)

channel(*) is an integer array containing 1 to 3 elements as follows:

- *channel*(1) is the ch# of the SMU that must be connected to the drain and gate. The drain must be connected to the gate.
- *channel*(2) is the ch# of the unit that must be connected to the source.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected instead of an SMU, you must specify only one element for *channel*(*), or you must specify *channel*(2) = 99.

- *channel*(3) is the ch# of the SMU that must be connected to the substrate. This function turns the SMU output switch to ON, and forces *substrate voltage* to the substrate ONLY IF *channel*(*) has three elements and *substrate voltage* is specified.

- *drain current* (I/O: I, type: real, unit: A)

The current value that is forced by the *channel*(1) SMU.

Unit	<i>drain current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *gate voltage compliance* (I/O: I, type: real, unit: V)

The voltage compliance value for the *channel*(1) SMU.

Unit	<i>gate voltage compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

Default = 20

FNVth1

- *substrate voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(3)* SMU.

Unit	<i>substrate voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

Default = no connection

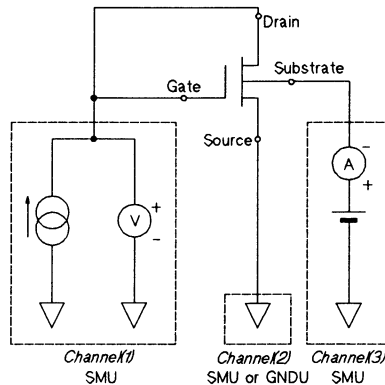
Example Statements

```
Vth = FNVth1 (Channel(*), 1.E-5)
```

```
Vth = FNVth1 (Channel(*), Id, Vgs_limit, Vsub)
```

Semantics

When using this function, the SMUs and GNDU at *channel(*)* must be connected to the transistor as shown in the circuit below.



Measurement Circuit

This function automatically includes a wait time (depending on the specified *drain current*) to allow the SMU capacitance to fully charge before the measurement is made.

The measurement result obtained by the *channel(1)* SMU is returned to the user-specified variable as the gate-source threshold voltage. If an abnormal condition is detected (see the status description in this chapter), -9999999.99999 is returned to the user-specified variable.

Immediately after the measurement, this function sets all SMUs to 0 V output, and sets all SMU output switches to OFF.

FNVth2

File: PARA_4142

This function determines the gate-source threshold voltage of an enhancement-type FET by measuring the gate-source voltage at two different *drain current* settings, then uses these values to calculate the threshold voltage as described in Semantics. The function then returns the calculated result to a user-specified variable.

Syntax

```
FNVth2 ( channel(*), drain current 1, drain current 2 [ , gate voltage compliance]
[ , substrate voltage] )
```

Parameters

- *channel*(*) (I/O: I, type: integer array)

channel(*) is an integer array containing from 1 to 3 elements

- *channel*(1) is the ch# of the SMU that must be connected to the drain and gate. The drain must be connected to the gate.
- *channel*(2) is the ch# of the module that must be connected to the source.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected instead of an SMU, you must specify only one element for *channel*(*), or you must specify *channel*(2) = 99.

- *channel*(3) is the ch# of the SMU that must be connected to the substrate. This function turns the SMU output switch to ON, and forces *substrate voltage* to the substrate ONLY IF *channel*(*) has three elements and *substrate voltage* is specified.

- *drain current 1* (I/O: I, type: real, unit: A)
- drain current 2* (I/O: I, type: real, unit: A)

The current value that is forced by the *channel*(1) SMU.

Unit	<i>drain current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *gate voltage compliance* (I/O: I, type: real, unit: V)

The voltage compliance value for the *channel*(1) SMU.

Unit	<i>gate voltage compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

FNVth2

Default = 20

- *substrate voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(3)* SMU.

Unit	<i>substrate voltage</i>
HPSMU	0 to ±200
MPSMU	0 to ±100

Default = no connection

Example Statements

```
Vth = FNVth2 (Channel(*), 1.E-6, 1.E-4)
```

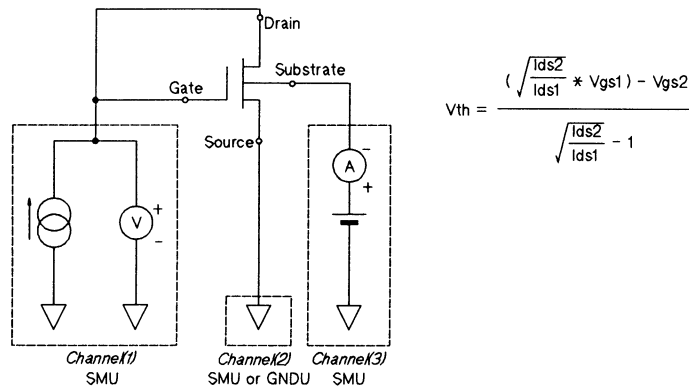
```
Vth = FNVth2 (Channel(*), Id1, Id2, Vgs_limit, Vsub)
```

Semantics

When using this function, the SMUs and GNDU at *channel(*)* must be connected to the transistor as shown in the circuit below.

This function automatically includes a wait time (depending on the specified *drain current*) to allow the SMU capacitance to fully charge before the measurement is made.

The transistor's gate voltage is measured twice: once when *drain current 1* is being forced, and again when *drain current 2* is being forced. The function then calculates the gate-source threshold voltage using the equation shown below:



Measurement Circuit

where I_{ds1} and I_{ds2} are the two specified drain currents, and V_{gs1} and V_{gs2} are the measured gate voltages. The calculated result is returned to the user-specified variable as the threshold voltage.

If an abnormal condition is detected (see the status description in this chapter),
–9999999.99999 is returned to the user-specified variable.

Immediately after the measurement, this function sets all SMUs to 0 V output, and sets all SMU output switches to OFF.

FNVth3

File: PARA_4142

This function measures the gate-source threshold voltage of depletion- or enhancement-type FETs by using the Analog Feedback Unit and two SMUs. The function then returns the measurement result to a user-specified variable.

Syntax

FNVth3 (channel(), target drain current, drain voltage, start gate voltage, stop gate voltage, gate current compliance [, ramp rate] [, feedback integration time] [, substrate voltage])*

Parameters

■ *channel(*)* (I/O: I, type: integer array)

channel()* is an integer array containing 2 to 4 elements as follows:

- *channel(1)* is the sense SMU ch#, and must be connected to the drain.
- *channel(2)* is the search SMU ch#, and must be connected to the gate.
- *channel(3)* is the ch# of the module that must be connected to the source.

If an SMU is connected, this function sets the SMU to 0 V output.

If a GNDU is connected, you must specify only two elements for *channel(*)*, or you must set *channel(3)* = 99.

- *channel(4)* is the ch# of the SMU that must be connected to the substrate. This function turns the SMU output switch to ON, and forces *substrate voltage* to the substrate ONLY IF *channel(*)* has four elements and *substrate voltage* is specified.

■ *target drain current* (I/O: I, type: real, unit: A)

The target current value for the *channel(1)* SMU (sense SMU).

Unit	<i>target drain current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

■ *drain voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(1)* SMU.

Unit	<i>drain voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *start gate voltage* (I/O: I, type: real, unit: V)

The search start voltage value for the *channel(2)* SMU.

Unit	<i>start gate voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *stop gate voltage* (I/O: I, type: real, unit: V)

The search stop voltage value for the *channel(2)* SMU.

Unit	<i>stop gate voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *gate current compliance* (I/O: I, type: real, unit: A)

The current compliance value for the *channel(2)* SMU.

Unit	<i>gate current compliance</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *ramp rate* (I/O: I, type: real, unit: V/s)

The ramp rate value of the *channel(2)* SMU (search SMU). The allowable values are 0.5 to 100000.

Default = 500

See the Set_asearch subprogram description.

- *feedback integration time* (I/O: I, type: real, unit: s)

The feedback integration time for the analog feedback measurement.

The allowable values are $0.5\text{E}-6$ to $450\text{E}-3$.

Default = 0.1

See the Measure_asearch subprogram description.

- *substrate voltage* (I/O: I, type: real, unit: V)

The voltage value that is forced by the *channel(4)* SMU.

FNVth3

Unit	substrate voltage
HPSMU	0 to ± 200
MPSMU	0 to ± 100

Example Statements

```
Vth = FNVth3 (Channel(*), Id, Vds, Vgmin, Vgmax, Igmax, Ramp, Integ, Vsub)
```

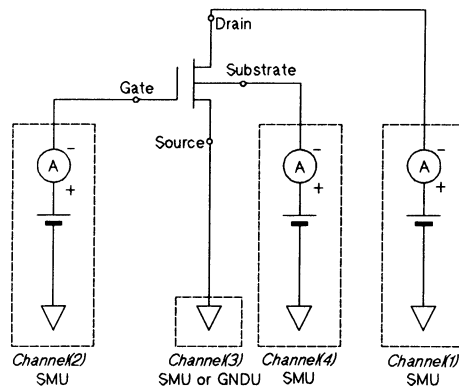
Semantics

When using this function, the SMUs and GNDU at *channel(*)* must be connected to the transistor as shown in the circuit below.

For this measurement, the function uses the AFU and two SMUs. The *channel(2)* SMU varies the gate voltage until the *channel(1)* SMU senses that the *target drain current* has been reached. Then, the function returns the gate voltage to the user-specified variable.

If an abnormal condition is detected (see the status description in this chapter), or if the threshold voltage is not between the *start gate voltage* and the *stop gate voltage*, the function returns -9999999.99999 to the user-specified variable.

Immediately after the measurement, this function sets all SMUs to 0 V output, and sets all SMU output switches to OFF.



Measurement Circuit

Force_i

File: HP4142_DRV

This subprogram forces *output current* from a specified SMU or HVU.

HP-IB Command:

DI

Syntax

```
Force_i ( output ch#, output current [ , I output range] [ , V compliance]
[ , compliance polarity mode] )
```

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8

- *output current* (I/O: I, type: real, unit: A)

Unit	<i>output current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$

- *I output range* (I/O: I, type: real)

0: Auto ranging
 1E-9: 1 nA Limited Auto ranging
 1E-8: 10 nA Limited Auto ranging
 1E-7: 100 nA Limited Auto ranging
 1E-6: 1 μA Limited Auto ranging
 1E-5: 10 μA Limited Auto ranging
 1E-4: 100 μA Limited Auto ranging
 1E-3: 1 mA Limited Auto ranging
 1E-2: 10 mA Limited Auto ranging
 1E-1: 100 mA Limited Auto ranging
 1: 1 A Limited Auto ranging

Default = 0

Force_i

- *V compliance* (I/O: I, type: real, unit: V)

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

Default:

- ☐ If the unit is set to I source mode:
Default = the previous setting
- ☐ If the unit is set to V source mode:
Default = none

- *compliance polarity mode* (I/O: I, type: integer)

- 0: Auto mode
- 1: Manual mode

Default = 0

Example Statements

```
Force_i (1, 1.0E-6, 0)
```

```
Force_i (Ch1, 1.0E-6, 0, 10.0, 0)
```

```
Force_i (Trbase, Current, Range, Compliance, Polarity)
```

Force_v

File: HP4142_DRV

This subprogram forces *output voltage* from a specified SMU, HVU, or VS.

HP-IB Command:

DV

Syntax

For SMUs or HVUs:

```
Force_v ( output ch#, output voltage [ , V output range ] [ , I compliance ]
         [ , compliance polarity mode ] )
```

For VSs:

```
Force_v ( output ch#, output voltage [ , V output range ] )
```

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

- *output voltage* (I/O: I, type: real, unit: V)

Unit	<i>output voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000
VS	0 to ± 40

- *V output range* (I/O: I, type: real)

0: Auto ranging
 2: 2 V Limited Auto ranging
 20: 20 V Limited Auto ranging
 40: 40 V Limited Auto ranging
 100: 100 V Limited Auto ranging
 200: 200 V Limited Auto ranging

Force_v

500: 500 V Limited Auto ranging
1000: 1000 V Limited Auto ranging

Default = 0

- *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HPSMU	$\pm(1\text{E}-12 \text{ to } 1)$
MPSMU	$\pm(1\text{E}-12 \text{ to } 100\text{E}-3)$
HVU	$\pm(1\text{E}-9 \text{ to } 10\text{E}-3)$

Default:

- If the SMU is set to V source mode:

Default = the previous setting

- If the SMU is set to I source mode:

Default = none

- *compliance polarity mode* (I/O: I, type: integer)

0: Auto mode

1: Manual mode

Default = 0

Example Statements

```
Force_v (1, 10.0, 0)
```

```
Force_v (Drain, Vds, 0, 1.0E-3)
```

```
Force_v (Ch1, Voltage, Range, Compliance, Polarity)
```

Init_computer

File: GRAPHICS

This subprogram sets keyboard auto-repeat parameters, clears the screen and turns off softkey labels, and enables alpha and graphics screens.

If the computer has a color display, this subprogram sets yellow alpha characters on a blue background.

Syntax

```
Init_computer ( [separate pages])
```

Parameters

- *separate page* (I/O: I, type: integer)

This optional parameter is available only for HP 9000 Series 300 computers that have a bit-mapped color display.

0: separate page OFF

1: separate page ON

Default = 0

Example Statements

```
Init_computer (1)
```

```
Init_computer
```

Init_hp4142

File: HP4142_DRV

This subprogram sets all plug-in units to the initial settings, and clears the output data buffer and all error conditions. This subprogram also sets the trigger mode to TM2. After this subprogram is executed, all bits of the status byte are masked except bits 7, 6, 5, and 3.

To ensure correct measurement execution, include Init_hp4142 at the beginning of the your program, before other Control Software subprograms.

HP-IB Command:

*RST, *SRE, TM

Syntax

```
Init_hp4142
```

Example Statements

```
Init_hp4142
```

Lingraph

File: GRAPHICS

This subprogram clears the CRT, draws and numbers a linear graph with *X-min* and *Y-min* as the minimum values and *X-max* and *Y-max* as the maximum values, and labels both axes and the graph.

Syntax

```
Lingraph ( X-min, X-max, Y-min, Y-max, X-axis name$, Y-axis name$, graph name$ [ , grid control] [ , quadrant] [ , right subtitle$] [ , left subtitle$])
```

Parameters

- *X-min* (I/O: I, type: real)
- X-max* (I/O: I, type: real)
- Y-min* (I/O: I, type: real)
- Y-max* (I/O: I, type: real)

These parameters define the lower left and upper right point of the graph. For the best results, you should set *X-min*, *X-max*, *Y-min*, and *Y-max* values that include all measurement values. The maximum and minimum values for an axis cannot be the same. *X-min* larger than *X-max*, and *Y-min* larger than *Y-max* are allowed.

- *X-axis name*\$ (I/O: I, type: string)
- Y-axis name*\$ (I/O: I, type: string)

Maximum 30 characters are allowed. For *quadrant* values 1 to 4, less than 20 characters are recommended. Otherwise, titles overlap other areas or other titles.

- *graph name*\$ (I/O: I, type: string)

The title of the graph. Maximum 35 characters are allowed.

- *grid control* (I/O: I, type: integer)

0: no grid lines
1: dotted grid lines
2: solid grid lines

Default = 0

- *quadrant* (I/O: I, type: integer)

0: full screen
1: upper right
2: upper left
3: lower left
4: lower right

Default = 0

- *right subtitle*\$ (I/O: I, type: string)
- left subtitle*\$ (I/O: I, type: string)

Maximum 20 characters are allowed.

Lingraph

Default = none

Example Statements

```
Lingraph (-5, 5, -1E-3, 1E-3, "Vce",  
         "Ic", "Vce-Ic Characteristics")
```

```
Lingraph (X_min, X_max, Y_min, Y_max, X_name$, Y_name$,  
         Title$, Grid, Quadrant, Right_subtitle$, Left_subtitle$)
```

Loggraph

File: GRAPHICS

This subprogram clears the CRT, draws and numbers a log graph with *X-min* and *Y-min* as the minimum values and *X-max* and *Y-max* as the maximum values, and labels both axes and the graph.

Syntax

```
Loggraph ( X-min, X-max, Y-min, Y-max, X-axis name$, Y-axis name$, graph name$
[ , graph mode] [ , grid control] [ , quadrant] [ , right subtitle] [ , left subtitle])
```

Parameters

- *X-min* (I/O: I, type: real)
- X-max* (I/O: I, type: real)
- Y-min* (I/O: I, type: real)
- Y-max* (I/O: I, type: real)

These parameters define the lower left and upper right point of the graph. For the best results, you should specify *X-min*, *X-max*, *Y-min*, and *Y-max* values that include all measurement values. The minimum and maximum values cannot be equal. For a log axis, the minimum and the maximum value for an axis must have the same polarity (0 is not allowed for a log axis). *X-min* larger than *X-max*, and *Y-min* larger than *Y-max* are allowed.

- *X-axis name*\$ (I/O: I, type: string)
- Y-axis name*\$ (I/O: I, type: string)

Maximum 30 characters are allowed. For *quadrant* values 1 to 4, less than 20 characters are recommended. Otherwise, titles overlap other areas or other titles.

- *graph name*\$ (I/O: I, type: string)

Maximum 35 characters are allowed.

- *graph mode* (I/O: I, type: integer)

Selects each axis scaling mode.

- 1: X-lin Y-log
- 2: X-log Y-lin
- 3: X-log Y-log

- *grid control* (I/O: I, type: integer)

- 0: no grid lines
- 1: dotted grid lines
- 2: solid grid lines

Default = 0

- *quadrant* (I/O: I, type: integer)

- 0: full screen
- 1: upper right
- 2: upper left

Loggraph

3: lower left

4: lower right

Default = 0

■ *right subtitle\$* (I/O: I, type: string)

left subtitle\$ (I/O: I, type: string)

Maximum 20 characters are allowed.

Default = none

Example Statements

```
Loggraph (0, 5, 1.E-12, 1.E-6, "Vgs", "Id",  
  "Vgs-Id Characteristics", 1)
```

```
Loggraph (Xmin, Xmax, Ymin, Ymax, Xname$, Yname$,  
  Graph_name$, Mode, Griding)
```

Measure_asearch

File: HP4142_DRV

This subprogram triggers an analog search measurement, and returns the measurement value(s) as determined by the parameters specified in this subprogram and the parameters specified in the Set_asource and Set_amonitor subprograms.

HP-IB Command:

FMT, MM, ASM, XE

Syntax

Measure_asearch (search operation mode, search measurement mode, feedback integration time, search data [, sense data] [, status])

Parameters

- *search operation mode* (I/O: I, type: integer)
 - 1: negative feedback search
 - 2: positive feedback search
 - 3: ramp wave search (greater then target)
 - 4: ramp wave search (less then target)
- *search measurement mode* (I/O: I, type: integer)
 - 1: search SMU V measurement
 - 2: search SMU I measurement
 - 3: search SMU V and sense SMU V or I measurement
 - 4: search SMU I and sense SMU V or I measurement

If the sense SMU is set to I source (voltage monitor) mode, a V measurement is made.

If the sense SMU is set to V source (current monitor) mode, an I measurement is made.

- *feedback integration time* (I/O: I, type: real, unit: s)

Measure_asearch

Search SMU Output Range ¹	<i>feedback integration time</i>	Resolution
2 V	50E-6 to 450E-6	50E-6
	0.5E-3 to 4.5E-3	0.5E-3
	5E-3 to 45E-3	5E-3
	50E-3 to 450E-3	50E-3
20 V	5E-6 to 45E-6	5E-6
	50E-6 to 450E-6	50E-6
	0.5E-3 to 4.5E-3	0.5E-3
	5E-3 to 45E-3	5E-3
40 V	2.5E-6 to 4.5E-6	0.5E-6
	5E-6 to 45E-6	5E-6
	50E-6 to 450E-6	50E-6
	0.5E-3 to 4.5E-3	0.5E-3
	5E-3 to 25E-3	5E-3
100 V	1E-6 to 4.5E-6	0.5E-6
	5E-6 to 45E-6	5E-6
	50E-6 to 450E-6	50E-6
	0.5E-3 to 4.5E-3	0.5E-3
	5E-3 to 10E-3	5E-3
200 V	0.5E-6 to 4.5E-6	0.5E-6
	5E-6 to 45E-6	5E-6
	50E-6 to 450E-6	50E-6
	0.5E-3 to 4.5E-3	0.5E-3
	5E-3	

¹ The *feedback integration time* allowed depends on the V output range that is automatically set for the search SMU. Refer to the Set_asource subprogram description.

The *feedback integration time* parameter has no meaning when the *search operation mode* parameter is set to 3 or 4.

- *search data* (I/O: O, type: real, unit: V or A)
- *sense data* (I/O: O, type: real, unit: V or A)

If you specify *search measurement mode* = 1 or 2, the search SMU measurement value is returned to *search data*. If you specify *search measurement mode* = 3 or 4, the search SMU measurement value is returned to *search data*, and the sense SMU measurement value is returned to the *sense data*.

- *status* (I/O: O, type: integer)

If you specify *status*, you must declare it as integer type. After the measurement is performed, an integer is returned to *status*.

If the *status* parameter is not set and an illegal status occurs, a message is displayed on the CRT.

For the returned *status*, the tens digit corresponds to search SMU measurement data and the ones digit corresponds to sense SMU measurement data. For example, if the returned status is 60, then 6 is the search SMU data status and 0 is the sense SMU data status.

For the meanings of the returned *status*, refer to the status description in this chapter.

Example Statements

```
Measure_asearch (1, 1, 1E-3, Data)
```

```
Measure_asearch (Search_mode, Meas_mode, Integ,  
Search_value, Sense_value, Status)
```

Measure_bdm

File: HP4142_DRV

This subprogram triggers a quasi-pulsed measurement and returns the measurement value as determined by the parameters specified in this subprogram and the parameters specified in the Set_bdm subprograms.

HP-IB Command:

MM, BDM, XE

Syntax

`Measure_bdm (measurement ch#, detection interval, V/I measurement, measured value
[, status])`

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HVU	2 to 8
HPSMU	2 to 8
MPSMU	1 to 8

- *detection interval* (I/O: I, type: integer)

0: Short
1: Long

- *V/I measurement* (I/O: I, type: integer)

0: Voltage measurement
1: Current measurement

- *measured value* (I/O: O, type: real, unit: V/A)

The measured value is returned to this parameter.

- *status* (I/O: O, type: integer)

If you specify status, you must declare it as integer type. After the measurement is performed, an integer is returned to *status*.

Refer to the status description in this chapter.

Example Statements

`Measure_bdm(2,0,1,Bvceo)`

Measure_i

File: HP4142_DRV

This subprogram triggers a high speed spot current measurement using the specified unit independent of the source mode (V/I source mode), and returns the measurement value to the specified return current variable.

HP-IB Command:

FMT, TI

Syntax

```
Measure_i ( measurement ch#, return current variable [ , I measurement range]
[ , status] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VS	1 to 8, 11 to 18, or 21 to 28

- *return current variable* (I/O: O, type: real, unit: A)

The measured current value is returned to this parameter.

- *I measurement range* (I/O: I, type: real)

0:	Auto ranging
1E-9:	1 nA Limited Auto ranging
10E-9:	10 nA Limited Auto ranging
100E-9:	100 nA Limited Auto ranging
1E-6:	1 μ A Limited Auto ranging
10E-6:	10 μ A Limited Auto ranging
100E-6:	100 μ A Limited Auto ranging
1E-3:	1 mA Limited Auto ranging
10E-3:	10 mA Limited Auto ranging
100E-3:	100 mA Limited Auto ranging
1:	1 A Limited Auto ranging
10:	10 A Limited Auto ranging
-1E-9:	1 nA range fixed
-10E-9:	10 nA range fixed

Measure_i

-100E-9:	100 nA range fixed
-1E-6:	1 μ A range fixed
-10E-6:	10 μ A range fixed
-100E-6:	100 μ A range fixed
-1E-3:	1 mA range fixed
-10E-3:	10 mA range fixed
-100E-3:	100 mA range fixed
-1:	1 A range fixed
-10:	10 A range fixed

Default = 0

- *status* (I/O: O, type: integer)

If you specify *status*, you must declare it as integer type. After the measurement is performed, an integer is returned to *status*.

Refer to the status description in this chapter.

Example Statements

```
Measure_i (1, Current)
```

```
Measure_i (Ch1, Current, 0)
```

```
Measure_i (Ch1, Ids, 10E-6, Data_status)
```

```
Measure_i (Drain, Ids, Range, Data_status)
```

Measure_v

File: HP4142_DRV

This subprogram triggers a high speed spot voltage measurement using the specified unit independent of the source mode (V/I source mode), and returns the measurement value to the specified *return voltage variable*.

HP-IB Command:

FMT, TV

Syntax

```
Measure_v ( measurement ch#, return voltage variable [ , V measurement range]
           [ , status] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HCU	2 to 8
HVU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

- *return voltage variable* (I/O: O, type: real, unit: V)

The measured voltage value is returned to this parameter.

V measurement range (I/O: I, type: real)

- The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of SMUs, HVUs, or HCU is set to the Compliance ranging automatically, regardless of specified value.

0: Auto ranging
 0.2: 0.2 V range fixed
 2: 2 V range fixed
 20: 20 V range fixed
 40: 40 V range fixed

Default = 0

- *status* (I/O: O, type: integer)

If you specify *status*, you must declare it as integer type. After the measurement is performed, an integer is returned to *status*.

Measure_v

Refer to the status description in this chapter.

Example Statements

```
Measure_v (1, Voltage)
```

```
Measure_v (1, Voltage, 0)
```

```
Measure_v (Drain, Vds, Range, Data_status)
```

Para_hfe

File: HP4142_DRV

This subprogram calculates and returns parameters that are required for h_{FE} measurements using the Analog Feedback Unit (AFU). The parameter calculations are based on the measurement condition parameters and DUT characteristic parameters that you enter for this subprogram. This subprogram returns the optimized *base current compliance*, *feedback integration time*, *ramp rate*, and *delay time* values needed to perform the h_{FE} measurement.

Syntax

Para_hfe (*collector voltage*, *target collector current*, *collector current compliance*, *start base voltage*, *stop base voltage*, *hfe min*, *hfe max*, *transition frequency*, *input capacitance*, *base-collector capacitance*, *maximum base current*, *base current compliance*, *feedback integration time*, *ramp rate*, *delay time* [, *error*] [, *hold time*] [*collector capacitance*])

Parameters

- *collector voltage* (I/O: I, type: real, unit: V)

Input the *output voltage* for Set_amonitor subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *target collector current* (I/O: I, type: real, unit: A)

Input the *target current* for Set_amonitor subprogram as this parameter.

The allowable values are $\pm 1E-11$ to ± 0.9 .

- *collector current compliance* (I/O: I, type: real, unit: A)

Input the *I compliance* for the Set_amonitor subprogram as this parameter. It is recommended that the following equation is satisfied:

$$\text{collector current compliance} = 1.05 \times \text{target collector current}$$

The allowable values are $\pm 1.05E-11$ to ± 1 .

- *start base voltage* (I/O: I, type: real, unit: V)

Input the *search start voltage* for the Set_asource subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *stop base voltage* (I/O: I, type: real, unit: V)

Input the *search stop voltage* for the Set_asource subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *hfe min* (I/O: I, type: real)
- *hfe max* (I/O: I, type: real)

Input the h_{FE} MAX and MIN value of DUT as these parameters. See DUT data sheet.

- *transition frequency* (I/O: I, type: real, unit: Hz)

Para_hfe

Input the transition frequency f_T (frequency when $h_{FE} = 1$) of DUT as this parameter. See DUT data sheet.

■ *input capacitance* (I/O: I, type: real, unit: F)

Input the search SMU's load capacitance as this parameter.

You can calculate the search SMU's load capacitance by adding the base input capacitance C_{ib} of DUT (see DUT data sheet) to your measurement circuit's residual capacitance.

The allowable values are 0 to 20E-9.

■ *base-collector capacitance* (I/O: I, type: real, unit: F)

Add the base-collector capacitance C_{bc} of DUT (see DUT data sheet) to your measurement circuit's residual capacitance and input this figure for the parameter.

The allowable values are 0 to 20E-9.

■ *maximum base current* (I/O: O, type: real, unit: A)

The *target collector current / hfe min* value is returned to this parameter.

■ *base current compliance* (I/O: O, type: real, unit: A)

The *I compliance* for the Set_asource subprogram is returned to this parameter.

■ *feedback integration time* (I/O: O, type: real, unit: s)

The *feedback integration time* for Measure_asearch subprogram is returned to this parameter.

■ *ramp rate* (I/O: O, type: real, unit: V/s)

The *ramp rate* for the Set_asource subprogram is returned to this parameter.

■ *delay time* (I/O: O, type: real, unit: s)

The *delay time* for the Set_asource subprogram is returned to this parameter.

■ *error* (I/O: O, type: integer)

If optimized parameters cannot be calculated from the parameters that you input, an error code is returned to the *error* and the operation stops. If you do not specify the *error* parameter, an error message is displayed on the CRT. The error code meanings are as follows:

- 0: No error
- 10: $|collector\ voltage| > 200$ is not allowed
- 11: Allowed *target collector current* values are $\pm 1E-11$ to ± 0.9
- 12: *target collector current* and *collector voltage* must be the same polarity
- 13: $|collector\ current\ compliance| > 1$ or $|collector\ current\ compliance| < 1.05 \times |target\ collector\ current|$ is not allowed
- 14: *collector current compliance* and *target collector current* must be same polarity
- 15: $|start\ base\ voltage| > 200$ is not allowed
- 16: $|stop\ base\ voltage| > 200$ is not allowed
- 17: $(stop\ base\ voltage - start\ base\ voltage)$ and *collector voltage* must be same polarity
- 19: $hfe\ min < 1$ or $hfe\ max < 1$ is not allowed
- 20: $hfe\ max \leq hfe\ min$ is not allowed
- 21: *transition frequency* ≤ 0 is not allowed
- 22: *input capacitance* must be 0 to 20 nF

- 23: *base-collector capacitance* must be 0 to 20 nF
- 24: *collector capacitance* must be 0 to 20 nF
- 30: $|start\ base\ voltage - stop\ base\ voltage|$ is too small
- 31: $|start\ base\ voltage - stop\ base\ voltage| > 200$ is not allowed

■ *hold time* (I/O: O, type: real, unit: s)

The *hold time* for the Set_asource subprogram is returned to this parameter.

■ *collector capacitance* (I/O: I, type: real, unit: F)

Input the sense SMU's load capacitance as this parameter.

You can calculate the sense SMU's load capacitance by adding the collector capacitance C_c of DUT (see DUT data sheet) to your measurement circuit's residual capacitance.

The allowable values are 0 to 20E-9.

Example Statements

```
Para_hfe (5, 1.E-2, 5.E-2, 0, .8, 25, 200,
          1.E+8, 5.E-12, 2.E-12, Ib, Ib_max, Integ,
          Rspeed, Dtime, Err)
```

Semantics

When using the AFU to perform an h_{FE} measurement, first execute the following subprogram to get the optimized parameters for the Set_asource and Measure_asearch subprograms:

```
Para_hfe (Vc, Ic, Ic_comp, Vb_start, Vb_stop, Hfe_min,
          Hfe_max, Ft, Cib, Cob, Ib_max, Ib_comp, Integ, Rspeed,
          Dtime, Err)
Set_asource (Ch1, Vb_start, Vb_stop, Rspeed, Htime, Dtime,
            Ib_comp)
Set_amonitor (Ch2, 1, Vc, Ic, Ic_comp)
Measure_asearch (1, 4, Integ, Ibmeas, Icmeas, Status)
Hfe = Icmeas/Ibmeas
```

Para_vth

File: HP4142_DRV

This subprogram calculates and returns parameters that are required for V_{TH} measurements using the AFU. The parameter calculations are based on the measurement condition parameters and DUT characteristic parameters that you enter for this subprogram. This subprogram returns the optimized *feedback integration time*, *ramp rate*, and *delay time* that are needed to perform the V_{TH} measurement.

The search SMU must be connected to the gate, the sense SMU must be connected to the drain, and the GNDU or a 0 V output SMU must be connected to the source.

Syntax

Para_vth (*drain voltage*, *target drain current*, *drain current compliance*, *start gate voltage*, *stop gate voltage*, *gate current compliance*, *input capacitance*, *gate-drain capacitance*, *feedback integration time*, *ramp rate*, *delay time* [, *error*] [, *hold time*] [, *gm*] [, *output capacitance*])

Parameters

- *drain voltage* (I/O: I, type: real, unit: V)

Input the *output voltage* for the Set_amonitor subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *target drain current* (I/O: I, type: real, unit: A)

Input the *target current* for the Set_amonitor subprogram as this parameter.

The allowable values are $\pm 1E-11$ to ± 0.9 .

- *drain current compliance* (I/O: I, type: real, unit: A)

Input the *I compliance* for the Set_amonitor subprogram as this parameter. It is recommended that the following equation is satisfied:

$$\text{drain current compliance} = 1.05 \times \text{target drain current}$$

The allowable values are $\pm 1.05E-11$ to ± 1 .

- *start gate voltage* (I/O: I, type: real, unit: V)

Input the *search start voltage* for the Set_asource subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *stop gate voltage* (I/O: I, type: real, unit: V)

Input the *search stop voltage* for the Set_asource subprogram as this parameter.

The allowable values are 0 to ± 200 .

- *gate current compliance* (I/O: I, type: real, unit: A)

Input the *I compliance* for the Set_asource subprogram as this parameter.

The allowable values are $\pm 1E-12$ to ± 1 .

■ *input capacitance* (I/O: I, type: real, unit: F)

Input the search SMU's load capacitance as this parameter.

You can calculate the search SMU's load capacitance by adding the base input capacitance *Ciss* of DUT (see DUT data sheet) to your measurement circuit's residual capacitance.

The allowable values are 0 to $2\text{E}-8$.

■ *gate-drain capacitance* (I/O: I, type: real, unit: F)

Add the gate-drain capacitance *Crss* of DUT (see DUT data sheet) to your measurement circuit's residual capacitance and input this figure for the parameter.

The allowable values are 0 to $2\text{E}-8$.

■ *feedback integration time* (I/O: O, type: real, unit: s)

The *feedback integration time* for the Measure_asearch subprogram is returned to this parameter.

■ *ramp rate* (I/O: O, type: real, unit: V/s)

The *ramp rate* for the Set_asource subprogram is returned to this parameter.

■ *delay time* (I/O: O, type: real, unit: s)

The *delay time* for the Set_asource subprogram is returned to this parameter.

■ *error* (I/O: O, type: integer)

If optimized parameters cannot be calculated from the parameters that you input, an error code is returned to the *error* parameter and the operation stops. If you do not specify the *error* parameter, an error message is displayed on the CRT. The error code meanings are as follows:

- 0: No error
- 10: *drain voltage* > 200 V is not allowed
- 11: Allowed *target drain current* values are $\pm 1\text{E}-11$ to ± 0.9 A
- 12: *drain voltage* and *target drain current* must be the same polarity
- 13: *drain current compliance* > 1 A or *drain current compliance* < $1.05 \times \text{target drain current}$ is not allowed
- 14: *drain current compliance* and *target drain current* must be same polarity
- 15: $|\text{start gate voltage}| > 200$ is not allowed
- 16: $|\text{stop gate voltage}| > 200$ is not allowed
- 17: $(\text{stop gate voltage} - \text{start gate voltage})$ and *drain voltage* must be same polarity
- 19: Allowed *gate current compliance* values are $\pm 1\text{E}-12$ to ± 1 A
- 20: *gate input capacitance* must be 0 to 20 nF
- 21: *gate-drain capacitance* must be 0 to 20 nF
- 22: *output capacitance* must be 0 to 20 nF
- 30: $|\text{start gate voltage} - \text{stop gate voltage}|$ is too small
- 31: $|\text{start gate voltage} - \text{stop gate voltage}| > 200$ is not allowed

■ *hold time* (I/O: O, type: real, unit: s)

The *hold time* for the Set_asource subprogram is returned to this parameter.

■ *gm* (I/O: I, type: real, unit: siemens)

Input the transconductance of the DUT (see DUT data sheet) as this parameter.

Para_vth

- *output capacitance* (I/O: I, type: real, unit: F)

Input the sense SMU's load capacitance as this parameter.

You can calculate the sense SMU's load capacitance by adding the output capacitance Coss of DUT (see DUT data sheet) to your measurement circuit's residual capacitance.

The allowable values are 0 to 2E-8.

Example Statements

```
Para_vth (10, 1.E-5, 3.E-2, 0, 3, 1.E-5,  
          5.E-12, 2.E-12, Integ, Rspeed, Dtime, Err)
```

Semantics

When using the AFU to perform an V_{TH} measurement, first execute the following subprogram to get the optimized parameters for the Set_asource and Measure_asearch subprograms:

```
Para_vth (Vd, Id, Id_comp, Vg_start, Vg_stop, Ig_comp, Ciss,  
          Crss, Integ, Rspeed, Dtime, Err, Gm, Cd, Htime)  
Set_asource (Ch1, Vg_start, Vg_stop, Rspeed, Htime, Dtime,  
            Ig_comp)  
Set_amonitor (Ch2, 1, Vd, Id, Id_comp)  
Measure_asearch (1, 3, Integ, Vth, Id, Status)
```

Pulse_i

File: HP4142_DRV

This subprogram sets an SMU, HVU, or HCU to pulsed I source and specifies its parameters for a pulsed measurement.

HP-IB Command:

PI, PT

Syntax

For SMUs or HVUs:

*Pulse_i (output ch#, I output range, base current, pulse current, pulse width, hold time
[, pulse period] [, V compliance])*

For HCU:

*Pulse_i (output ch#, I output range, base current, pulse current, pulse width, hold
time, pulse period, V compliance)*

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8

- *I output range* (I/O: I, type: real)

0: Auto ranging
 1E-8: 10 nA Limited Auto ranging
 1E-7: 100 nA Limited Auto ranging
 1E-6: 1 μ A Limited Auto ranging
 1E-5: 10 μ A Limited Auto ranging
 1E-4: 100 μ A Limited Auto ranging
 1E-3: 1 mA Limited Auto ranging
 1E-2: 10 mA Limited Auto ranging
 1E-1: 100 mA Limited Auto ranging
 1: 1 A Limited Auto ranging
 10: 10 A Limited Auto ranging

Default = 0

- *base current* (I/O: I, type: real, unit: A)
- *pulse current* (I/O: I, type: real, unit: A)

Pulse_i

Unit	<i>base current</i> ¹	<i>pulse current</i> ¹
HPSMU	0 to ± 1	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$	0 to $\pm 10\text{E}-3$
HCU	0 ²	0 to ± 10

¹ The *pulse current* and *base current* polarity must be the same.

² During base value output, the HCU output is 0 V and no current.

■ *pulse width* (I/O: I , type: real, unit: s)

0.0001 to 0.05

resolution: 0.0001

■ *hold time* (I/O: I , type: real, unit: s)

0 to 655.35

resolution: 0.01

■ *pulse period* (I/O: I , type: real, unit: s)

0.01 to 0.5

resolution: 0.0001

Default = 0

■ *V compliance* (I/O: I, type: real, unit: V)

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000
HCU	0 to ± 10

Default:

- If the specified SMU is set to I source mode before the trigger:

Default = the setting before trigger

- If the specified SMU is set to V source mode before the trigger:

Default = none

Example Statements

```
Pulse_i (1, 0, 1E-3, 50-E-3, 1E-3, 0.1)
```

```
Pulse_i (Ch1, Range, Base, Value, Width, Hold_time, Period, Compliance)
```

Pulse_measure

File: HP4142_DRV

This subprogram triggers a 1-ch pulsed spot measurement. A pulse output is determined by the Pulse_v or Pulse_i subprogram, a measurement is performed at the specified measurement ch#, and the measurement value is returned to the specified *measurement variable*.

HP-IB Command:

MM, FL, RV, RI, XE

Syntax

For Voltage Measurements:

```
Pulse_measure ( measurement ch#, 1, measurement variable [ , V measurement range ]
[ , status ] )
```

For Current Measurements:

```
Pulse_measure ( measurement ch#, 2, measurement variable [ , I measurement range ]
[ , status ] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

- *measurement variable* (I/O: O, type: real, unit: V or A)

The measured value is returned to this parameter.

- *V measurement range* (I/O: I, type: real)

The V measurement range sets the voltage measurement range of VMs. The voltage measurement range of SMUs, HVUs, or HCU is set to the Compliance range automatically, regardless of the specified value.

0: 40 V range fixed
 2: 2 V range fixed
 20: 20 V range fixed
 40: 40 V range fixed

Default = 0

Pulse_measure

■ *I measurement range* (I/O: I, type: real)

0:	Compliance range
-1E-8:	10 nA range fixed
-1E-7:	100 nA range fixed
-1E-6:	1 μ A range fixed
-1E-5:	10 μ A range fixed
-1E-4:	100 μ A range fixed
-1E-3:	1 mA range fixed
-1E-2:	10 mA range fixed
-1E-1:	100 mA range fixed
-1:	1 A range fixed
-10:	10 A range fixed

Default = 0

■ *status* (I/O: O, type: integer)

If you specify *status*, you must declare it as integer type. After the measurement is performed, an integer is returned to *status*.

Refer to the status description in this chapter.

Example Statements

```
Pulse_measure (1, 1, Voltage)
```

```
Pulse_measure (2, 2, Current)
```

```
Pulse_measure (Ch1, Meas_mode, Meas_value, Range, Status)
```

Pulse_v

File: HP4142_DRV

This subprogram sets an SMU, HVU, HCU, or VS to pulsed V source and specifies its parameters for a pulsed measurement.

HP-IB Command:

PV, PT

Syntax

Pulse_v (output ch#, V output range, base voltage, pulse voltage, pulse width, hold time [, pulse period] [, I compliance])

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

- *V output range* (I/O: I, type: real)

0: Auto ranging
 2: 2 V Limited Auto ranging
 20: 20 V Limited Auto ranging
 40: 40 V Limited Auto ranging
 100: 100 V Limited Auto ranging
 200: 200 V Limited Auto ranging
 500: 500 V Limited Auto ranging
 1000: 1000 V Limited Auto ranging

Default = 0

- *base voltage* (I/O: I, type: real, unit: V)
- *pulse voltage* (I/O: I, type: real, unit: V)

Pulse_v

Unit	base voltage	pulse voltage
HPSMU	0 to ± 200	0 to ± 200
MPSMU	0 to ± 100	0 to ± 100
HVU ¹	0 to ± 1000	0 to ± 1000
HCU	0	0 to ± 10

¹ The *base voltage*, *start pulse voltage* and *stop pulse voltage* must be the same polarity.

- *pulse width* (I/O: I, type: real, unit: s)
0.0001 to 0.05 resolution: 0.0001
- *hold time* (I/O: I, type: real, unit: s)
0 to 655.35 resolution: 0.01
- *pulse period* (I/O: I, type: real, unit: s)
0.01 to 0.5 resolution: 0.0001
Default = 0
- *I compliance* (I/O: I, type: real, unit: A)

Unit	I compliance
HPSMU	$\pm 2\text{E}-9$ to ± 1
MPSMU	$\pm 2\text{E}-9$ to $\pm 100\text{E}-3$
HVU	$\pm 1\text{E}-9$ to $\pm 10\text{E}-3$
HCU	$\pm 1\text{E}-6$ to ± 10

Default:

- ☐ If the specified SMU or HCU is set to V source mode before the trigger: Default = the setting before trigger
- ☐ If the specified SMU or HCU is set to I source mode before the trigger: Default = none

Example Statements

```
Pulse_v (1, 0, 0.1, 10, 1E-3, 0)
```

```
Pulse_v (Ch1, Range, Base, Value, Width, Hold_time, Compliance)
```

Recover_output

File: HP4142_DRV

This subprogram restores the source unit settings that were cleared during the Zero_output subprogram execution.

Note



If the output polarity of the HVU was changed after executing Zero_output, this instruction is not acceptable.

HP-IB Command:

RZ

Syntax

Recover_output ([*ch#*] [, *ch#*] [, *ch#*] [, *ch#*] [, *ch#*] [, *ch#*] [, *ch#*] [, *ch#*])

Parameters

- *ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

Default: all units

Example Statements

```
Recover_output(1)

Recover_output(Ch1)

Recover_output(Ch1,Ch2,Ch3,Ch4)

Recover_output
```

Self_test

File: HP4142_DRV

This subprogram performs the HP 4142B Self-Test, then stores the test result code in *test results*, and a related message in *message\$*. You can then display this information.

HP-IB Command:

*TST?

Syntax

Self_test (*test type*, *test results* [, *message\$*])

Parameters

■ *test type* (I/O: I, type: integer)

If *test type* = 0, the mainframe and all units are tested.

If *test type* = 1 to 8, the unit at the corresponding slot# is tested.

If *test type* = 9, the mainframe is tested.

■ *test results* (I/O: O, type: integer)

The *test results* can be interpreted using the following table.

<i>test results</i>	Description
0	passed
1	Slot 1 unit failed
2	Slot 2 unit failed
4	Slot 3 unit failed
8	Slot 4 unit failed
16	Slot 5 unit failed
32	Slot 6 unit failed
64	Slot 7 unit failed
128	Slot 8 unit failed
256	Mainframe failed

If more than one unit failed, *test results* are the sum of the numbers corresponding to the failed units.

■ *message\$* (I/O: O, type: string)

If this parameter is used, DIM Message\$[60] is required.

Example Statements

```
Self_test (1, Result)
```

```
Self_test (Type, Result, Message$)
```

Set_amonitor

File: HP4142_DRV

This subprogram specifies an SMU at *sense ch#* to be the sense SMU for an analog search measurement, and sets the sense SMU parameters.

HP-IB Command:

AIV, AVI

Syntax

For Setting the Sense SMU to V Source (Current Monitor) Mode:

Set_amonitor (*sense ch#*, 1, *output voltage*, *target current* [, *I compliance*])

For Setting the Sense SMU to I Source (Voltage Monitor) Mode:

Set_amonitor (*sense ch#*, 2, *output current*, *target voltage* [, *V compliance*])

Parameters

- *sense ch#* (I/O: I, type: integer)

Unit	<i>sense ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8

- *output voltage* (I/O: I, type: real, unit: V)

Unit	<i>output voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *output current* (I/O: I, type: real, unit: A)

Unit	<i>output current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *target current* (I/O: I, type: real, unit: A)

Unit	<i>target current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *target voltage* (I/O: I, type: real, unit: V)

Unit	<i>target voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *I compliance* (I/O: I, type: real, unit: A)

I compliance must be greater than *target current*.

Unit	<i>I compliance</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$

- *V compliance* (I/O: I, type: real, unit: V)

V compliance must be greater than *target voltage*.

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

Example Statements

```
Set_amonitor (1, 1, 2.5, 100E-3)
```

```
Set_amonitor (8, 2, 50E-3, 5)
```

```
Set_amonitor (Ch1, Mode, Output_value, Goal_value, Compliance)
```

Set_asource

File: HP4142_DRV

This subprogram specifies an SMU at search ch# to be the search SMU for an analog search measurement, and specifies the search SMU parameters.

HP-IB Command:

ASV, AT

Syntax

Set_asource (*search ch#*, *search start voltage*, *search stop voltage*, *ramp rate*, *hold time*, *delay time* [, *I compliance*])

Parameters

- *search ch#* (I/O: I, type: integer)

Unit	<i>search ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8

- *search start voltage* (I/O: I, type: real, unit: V)
search stop voltage (I/O: I, type: real, unit: V)

Unit	<i>search start voltage</i> <i>search stop voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *ramp rate* (I/O: I, type: real, unit: V/s)

Output Range	ramp rate	Resolution
2 V	0.5 to 5	0.05 V/s
	5.5 to 50	0.5 V/s
	55 to 500	5 V/s
	550 to 5000	50 V/s
20 V	5.5 to 50	0.5 V/s
	55 to 500	5 V/s
	550 to 5000	50 V/s
	55500 to 50000	500 V/s
40 V	10 to 50	1 V/s
	55 to 100	5 V/s
	110 to 500	10 V/s
	550 to 1000	50 V/s
	1100 to 5000	100 V/s
	5500 to 10000	500 V/s
	11000 to 50000	1000 V/s
	55000 to 100000	5000 V/s
100 V	25 to 50	2.5 V/s
	55 to 250	5 V/s
	275 to 500	25 V/s
	550 to 2500	50 V/s
	2750 to 5000	250 V/s
	5500 to 25000	500 V/s
	27500 to 50000	2500 V/s
	55000 to 100000	5000 V/s
200 V	55 to 500	5 V/s
	550 to 5000	50 V/s
	5500 to 50000	500 V/s
	55000 to 100000	5000 V/s

- *hold time* (I/O: I, type: real, unit: s)

0 to 65.535 resolution: 0.001

- *delay time* (I/O: I, type: real, unit: s)

0 to 65.535 resolution: 0.001

Set_asource

- *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HPSMU	$\pm(1\text{E}-12 \text{ to } 1)$
MPSMU	$\pm(1\text{E}-12 \text{ to } 100\text{E}-3)$

Default:

- If the specified SMU is set to V source mode before the trigger:

Default = the setting before the trigger

- If the specified SMU is set to I source mode before the trigger:

Default = none

Example Statements

```
Set_asource (1, 0, 5, 500, 1, 1)
```

```
Set_asource (Ch1, Start, Stop, Rate, Hold_time, Delay_time, Compliance)
```

Set_bdm

File: HP4142_DRV

This subprogram specifies an HVU or SMU as the quasi-pulsed source and specifies its parameters.

HP-IB Command:

BDV, BDT

Syntax

*Set_bdm (output ch#, V output range, start voltage, stop voltage, hold time, delay time
[, I compliance])*

Parameter

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HVU	2 to 8
HPSMU	2 to 8
MPSMU	1 to 8

- *V output range* (I/O: I, type: real)

0: Auto ranging
 2: 2 V Limited Auto ranging
 20: 20 V Limited Auto ranging
 40: 40 V Limited Auto ranging
 100: 100 V Limited Auto ranging
 200: 200 V Limited Auto ranging
 500: 500 V Limited Auto ranging
 1000: 1000 V Limited Auto ranging

- *start voltage* (I/O: I, type: real, unit: V)

- *stop voltage* (I/O: I, type: real, unit: V)

Unit	<i>start voltage</i> <i>stop voltage</i>
HVU	0 to ± 1000
HPSMU	0 to ± 200
MPSMU	0 to ± 100

- *hold time* (I/O: I, type: real, unit: s)

Set_bdm

0 to 655.35

resolution: 0.01

- *delay time* (I/O: I, type: real, unit: s)

0 to 6.5535

resolution: 0.0001

- *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HVU	$\pm(1\text{E}-9 \text{ to } 10\text{E}-3)$
HPSMU	$\pm(1\text{E}-12 \text{ to } 1)$
MPSMU	$\pm(1\text{E}-12 \text{ to } 100\text{E}-3)$

Default:

- ☐ If the specified source unit is set to V source mode before the trigger:

Default = the setting before trigger

- ☐ If the specified source unit is set to I source mode before the trigger:

Default = none

Example Statement

```
Set_bdm(2,500,200,210,1,1)
```

Set_iv

File: HP4142_DRV

This subprogram sets the parameters for a staircase sweep source.

HP-IB Command:

WI, WV, WT

Syntax

For Voltage Sweeps:

Set_iv (output ch#, voltage sweep mode, V output range, start voltage, stop voltage, number of steps [, hold time] [, delay time] [, I compliance] [, power compliance])

For Current Sweeps:

Set_iv (output ch#, current sweep mode, I output range, start current, stop current, number of steps [, hold time] [, delay time] [, V compliance] [, power compliance])

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

- *voltage sweep mode* (I/O: I, type: integer)

- 1: linear sweep (single stair)
- 3: linear sweep (double stair)
- 1: log sweep (single stair)
- 3: log sweep (double stair)

- *current sweep mode* (I/O: I, type: integer)

- 2: linear sweep (single stair)
- 4: linear sweep (double stair)
- 2: log sweep (single stair)
- 4: log sweep (double stair)

- *V output range* (I/O: I, type: real)

- 0: Auto ranging
- 2: 2 V Limited Auto ranging
- 20: 20 V Limited Auto ranging
- 40: 40 V Limited Auto ranging

Set_iv

100: 100 V Limited Auto ranging
200: 200 V Limited Auto ranging
500: 500 V Limited Auto ranging
1000: 1000 V Limited Auto ranging

■ *I output range* (I/O: I, type: real)

0: Auto ranging
1E-9: 1 nA Limited Auto ranging
1E-8: 10 nA Limited Auto ranging
1E-7: 100 nA Limited Auto ranging
1E-6: 1 μ A Limited Auto ranging
1E-5: 10 μ A Limited Auto ranging
1E-4: 100 μ A Limited Auto ranging
1E-3: 1 mA Limited Auto ranging
1E-2: 10 mA Limited Auto ranging
1E-1: 100 mA Limited Auto ranging
1: 1 A Limited Auto ranging

■ *start voltage* (I/O: I, type: real, unit: V)

stop voltage (I/O: I, type: real, unit: V)

Unit	<i>start voltage</i> <i>stop voltage</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU ¹	0 to ± 1000
VS	0 to ± 40

¹ The *start voltage* and *stop voltage* must be the same polarity.

■ *start current* (I/O: I, type: real, unit: A)

stop current (I/O: I, type: real, unit: A)

Unit	<i>start current</i> <i>stop current</i>
HPSMU	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$
HVU ¹	0 to $\pm 10\text{E}-3$

¹ The *start current* and *stop current* must be the same polarity.

■ *number of steps* (I/O: I, type: integer)

2 to 1001

■ *hold time* (I/O: I, type: real, unit: s)

4-82 Programming Reference

0 to 655.35

resolution = 0.01

Default = 0

■ *delay time* (I/O: I, type: real, unit: s)

0 to 65.535

resolution = 0.001

Default = 0

■ *V compliance* (I/O: I, type: real, unit: V)

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000

Default:

- ☐ If the specified SMU is set to I source mode before the trigger:

Default = the setting before the trigger

- ☐ If the specified SMU is set to V source mode before the trigger:

Default = none

■ *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HPSMU	$\pm(1\text{E}-12 \text{ to } 1)$
MPSMU	$\pm(1\text{E}-12 \text{ to } 100\text{E}-3)$
HVU	$\pm(1\text{E}-9 \text{ to } 10\text{E}-3)$

Default:

- ☐ If the specified SMU is set to V source mode before the trigger:

Default = the setting before the trigger

- ☐ If the specified SMU is set to I source mode before the trigger:

Default = none

■ *power compliance* (I/O: I, type: real, unit: W)

0.001 to 14

resolution = 0.001

Default = Does not set the power compliance

Set_iv

Example Statements

```
Set_iv (1, 1, 0, 0, 10, 50, 1, 0)
```

```
Set_iv (Ch1, Sweep_mode, Range, Start_val, Stop_val, Steps,  
        Hold, Delay, I_comp, P_comp)
```

```
Set_iv (1, -2, 0, 1.0E-9, 1.0E-5, 51)
```

```
Set_iv (Ch8, Sweep_mode, Range, Start_val, Stop_val, Steps,  
        Hold, Delay, V_comp, P_comp)
```

Set_piv

File: HP4142_DRV

This subprogram sets the parameters of a pulsed sweep source for a pulsed sweep measurement.

HP-IB Command:

PWI, PWV, PT

Syntax

For Pulsed Voltage Sweeps:

```
Set_piv ( output ch#, V sweep mode, V output range, base voltage, start pulse voltage,
stop pulse voltage, number of steps, pulse width, pulse period, hold time [ , I compliance]
)
```

For Pulsed Current Sweeps:

```
Set_piv ( output ch#, I sweep mode, I output range, base current, start pulse
current, stop pulse current, number of steps, pulse width, pulse period, hold time
[ , V compliance] )
```

Parameters

- *output ch#* (I/O: I, type: integer)

Unit	<i>output ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VS	1 to 8, 11 to 18 or 21 to 28

- *V sweep mode* (I/O: I, type: integer)

- 1: linear sweep (single stair)
- 3: linear sweep (double stair)

- *I sweep mode* (I/O: I, type: integer)

- 2: linear sweep (single stair)
- 4: linear sweep (double stair)

- *V output range* (I/O: I, type: real)

- 0: Auto ranging
- 2: 2 V Limited Auto ranging
- 20: 20 V Limited Auto ranging
- 40: 40 V Limited Auto ranging

Set_piv

100: 100 V Limited Auto ranging
200: 200 V Limited Auto ranging
500: 500 V Limited Auto ranging
1000 1000 V Limited Auto ranging

■ *I output range* (I/O: I, type: real)

0: Auto ranging
1E-8: 10 nA Limited Auto ranging
1E-7: 100 nA Limited Auto ranging
1E-6: 1 μ A Limited Auto ranging
1E-5: 10 μ A Limited Auto ranging
1E-4: 100 μ A Limited Auto ranging
1E-3: 1 mA Limited Auto ranging
1E-2: 10 mA Limited Auto ranging
1E-1: 100 mA Limited Auto ranging
1: 1 A Limited Auto ranging
10: 10 A Limited Auto ranging

■ *base voltage* (I/O: I, type: real, unit: V)

start pulse voltage (I/O: I, type: real, unit: V)

stop pulse voltage (I/O: I, type: real, unit: V)

Unit	<i>base voltage</i>	<i>start pulse voltage</i> <i>stop pulse voltage</i>
HPSMU	0 to ± 200	0 to ± 200
MPSMU	0 to ± 100	0 to ± 100
HVU ¹	0 to ± 1000	0 to ± 1000
HCU	0	0 to ± 10
VS	0 to ± 40	0 to ± 40

¹ The *base voltage* and *pulse voltage* must be the same polarity.

■ *base current* (I/O: I, type: real, unit: A)

start pulse current (I/O: I, type: real, unit: A)

stop pulse current (I/O: I, type: real, unit: A)

Unit	<i>base current</i> ¹	<i>start pulse current</i> ¹ <i>stop pulse current</i> ¹
HPSMU	0 to ± 1	0 to ± 1
MPSMU	0 to $\pm 100\text{E}-3$	0 to $\pm 100\text{E}-3$
HVU	0 to $\pm 10\text{E}-3$	0 to $\pm 10\text{E}-3$
HCU	0 ²	0 to ± 10

¹ The *start pulse current*, *stop pulse current*, and *base current* must be the same polarity.

² During base value output, the HCU output is 0 V and no current.

- *number of steps* (I/O: I, type: integer)
2 to 1001
- *pulse width* (I/O: I, type: real, unit: s)
0.0001 to 0.05
resolution: 0.0001
- *pulse period* (I/O: I, type: real, unit: s)
0.01 to 0.5
resolution: 0.0001
- *hold time* (I/O: I, type: real, unit: s)
0 to 655.35
resolution: 0.01
- *V compliance* (I/O: I, type: real, unit: V)

Unit	<i>V compliance</i>
HPSMU	0 to ± 200
MPSMU	0 to ± 100
HVU	0 to ± 1000
HCU	0 to ± 10

Default:

- ☐ If the specified SMU is set to I source mode before the trigger:
Default = the setting before the trigger
- ☐ If the specified SMU is set to V source mode before the trigger:
Default = none

Specify this parameter when you use an HCU as a pulsed sweep source.

- *I compliance* (I/O: I, type: real, unit: A)

Unit	<i>I compliance</i>
HPSMU	$\pm(1\text{E}-12 \text{ to } 1)$
MPSMU	$\pm(1\text{E}-12 \text{ to } 100\text{E}-3)$
HVU	$\pm(1\text{E}-9 \text{ to } 10\text{E}-3)$
HCU	$\pm(1\text{E}-6 \text{ to } 10)$

Default:

- ☐ If the specified SMU or HCU is set to V source mode before the trigger:

Set_piv

Default = the setting before the trigger

- If the specified SMU or HCU is set to I source mode before the trigger:

Default = none

Example Statements

```
Set_piv (11, 1, 0, 0.1, 0.1, 10, 15, 1E-3,  
10E-3, 0)
```

```
Set_piv (Ch, Mode, Range, Base, Start, Stop, Steps, Width,  
Period, Hold_time, I_comp)
```

Set_pol

This subprogram changes the polarity of the HVU output and sets the output voltage to 0 V. If the output switch of the unit is set to off, this subprogram also sets the switch to on.

Warning **Setting the output switch to on enables the unit to force dangerous voltages.**



Set the output switch to off whenever possible when the unit is not in use.

HP-IB Command:

POL

Execution Conditions

- The INTLK terminal is shorted.
- The other unit is not in HIGH VOLTAGE state (forcing more than 42 V, or *V compliance* set to more than 42 V).

Syntax

`Set_pol ( ch#, polarity )`

Parameters

- *ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HVU	2 to 8

- *polarity* (I/O: I, type: integer)

0: + polarity

1: – polarity

Example Statement

`Set_pol(2,0)`

Semantics

If the output switch of the unit is set to on, this subprogram changes the output polarity as follows:

1. Sets the unit to 0 V (same conditions as the DZ command execution).
2. Waits for either of the following conditions:
 - The unit continues to measure the output voltage until the output voltage becomes less than or equal to 30 V.

Set_pol

- Until 500 ms elapse.
3. Sets the output switch to off.
 4. Changes the output polarity.
 5. Sets the output switch to on.

Set_smu

File: HP4142_DRV

This subprogram sets the number of samples that are taken (A/D conversion) and averaged for the measurement.

HP-IB Command:

AV

Syntax

```
Set_smu ( averaging number )
```

Parameters

- *averaging number* (I/O: I, type: integer)

1 to 1023

–1 to –1023

If *averaging number* is a positive integer, AUTO mode averaging is performed.

If *averaging number* is a negative integer, POWER LINE CYCLE mode averaging is performed.

Example Statements

```
Set_smu (2)
```

```
Set_smu (–5)
```

```
Set_smu (Average_times)
```

Set_vm

File: HP4142_DRV

This subprogram sets the voltage measurement operation mode for the specified VM (HP 41424A).

HP-IB Command:

VM

Syntax

`Set_vm ( measurement ch#, VM operation mode )`

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
VM	1 to 8, 11 to 18 or 21 to 28

- *VM operation mode* (I/O: I, type: integer)

- 1: Grounded measurement
- 2: Differential measurement

Example Statements

`Set_vm (8, 1)`

`Set_vm (Channel, Vm_mode)`

status

Five subprograms (Dpulse_measure, Measure_asearch, Measure_i, Measure_v, Measure_bdm, and Pulse_measure) have the optional parameter status. If you specify status, you must declare it as integer type. After the measurement is performed, an integer is returned to status and the meanings are as follows:

- 0: Normal measurement data.
- 1: Another channel reached V compliance, I compliance, power compliance, or the current limit of VS.

Or, if another channel is the HCU, the measurement was performed before the pulsed output settled.

Either made the *pulse width* larger to wait for the settling time, or made the *I/V compliance* larger to speed up the settling time of the pulse output.
- 2: This measurement channel reached V compliance, I compliance, power compliance, or current limit of VS.

Or, if this channel is the HCU, the measurement was performed before the pulsed output settled.

Either made the *pulse width* larger to wait for the settling time, or made the *I/V compliance* larger to speed up the settling time of the pulse output.
- 3: This or another SMU or HVU oscillated, or the *pulse* value could not be set because the specified *pulse width* was too small.
- 4: Measurement exceeded measurement range, or dummy data were stored because the sweep measurement was automatically aborted by automatic sweep abort function or power compliance.
- 5: The target value for analog feedback measurement was not reached during a search (between *search start voltage* and *search stop voltage*).
- 6: Measurement was made before the analog feedback search was complete.
- 7: Measurement was made before the HVU was settled.
- 8: The detection time for the quasi-pulsed measurement is over the limit (3 s for Short mode, 12 s for Long mode). Set the *detection interval* to Long.
- 9: The settling detection for quasi-pulsed measurement can not perform because the output slew rate is too slow.

For the Measure_asearch subprogram, two digits are returned. The tens digit corresponds to the search SMU measurement data, and the ones digit corresponds to the sense SMU measurement data.

For example, if the returned *status* is 60, then 6 is the search SMU data status and 0 is the sense SMU data status.

Sweep_iv

File: HP4142_DRV

This subprogram triggers a staircase sweep measurement. Measurements are made at the measurement *ch#* for each sweep step, and the measurement values and sweep source values are returned to the measurement value array(*) and the source value array(*), respectively.

HP-IB Command:

MM, WNU?, FMT, RV, RI, XE

Syntax

For Voltage Measurements:

```
Sweep_iv ( measurement ch#, 1, V measurement range, measurement value array(*)  
[ , source value array(*)] )
```

For Current Measurements:

```
Sweep_iv ( measurement ch#, 2, I measurement range, measurement value array(*)  
[ , source value array(*)] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>measurement ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

- *V measurement range* (I/O: I, type: real)

The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of SMUs or HVUs is set to the Compliance range automatically, regardless of the specified value.

0: Auto ranging
0.2: 0.2 V range fixed
2: 2 V range fixed
20: 20 V range fixed
40: 40 V range fixed

- *I measurement range* (I/O: I, type: real)

0: Auto ranging
-1E-9: 1 nA range fixed
-1E-8: 10 nA range fixed

-1E-7: 100 nA range fixed
 -1E-6: 1 μ A range fixed
 -1E-5: 10 μ A range fixed
 -1E-4: 100 μ A range fixed
 -1E-3: 1 mA range fixed
 -1E-2: 10 mA range fixed
 -1E-1: 100 mA range fixed
 -1: 1 A range fixed

- *measurement value array (*)* (I/O: O, type: numeric array, unit: V or I)

The measurement value of each sweep step is returned to this array. The number of elements of the *measurement value array(*)* must be larger than the number of sweep steps.

- *source value array (*)* (I/O: O, type: numeric array, unit: V or I)

The sweep source values of each sweep step are returned to this array. The number of elements of the *source value array(*)* must be larger than the number of sweep steps.

Example Statements

```
Sweep_iv (1, 1, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_iv (6, 2, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_iv (Ch, Meas_mode, Range, Meas_data(*), Source_value(*))
```

Note



When using this subprogram, it is recommended that you include OPTION BASE 1 at the beginning of your program. Including OPTION BASE 1 in your program causes array element numbering to start from 1 instead of 0. This makes it easier to keep track of the array elements in the *measurement value array(*)* and [*source value array(*)*] because the *n*th returned value is array element *n* instead of *n - 1*.

Sweep_miv

File: HP4142_DRV

This subprogram triggers a multichannel staircase sweep measurement. Measurements are made at the units specified in the *measurement ch# array(*)*, and the measurement values and sweep source values are returned to the *measurement value array(*)* and *source value array(*)*, respectively.

HP-IB Command:

MM, WNU?, FMT, RV, RI, XE

Syntax

Sweep_miv (*measurement ch# array(*)*, *measurement mode array(*)*, *measurement range array(*)*, *measurement value array(*)* [, *source value array(*)*])

Parameters

■ *measurement ch# array(*)* (I/O: I, type: integer array)

Unit	ch#
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

Declare this parameter as a one dimensional array as follows:

INTEGER Channel (x)

x = number of measurement units (1 to 8)

EX.

INTEGER Channel (3)
Channel (1) = 1
Channel (2) = 4
Channel (3) = 6

Each element of this array defines a unit that measures any value in the measurements.

■ *measurement mode array(*)* (I/O: I, type: integer array)

<i>measurement mode</i>	description
1	voltage measurement
2	current measurement

Declare this parameter as a one dimensional array as shown in the following example:

```
INTEGER Mode (x)
```

x = number of measurement units (1 to 8)

Elements of this array correspond to elements in the *measurement ch# array(*)*.

- *measurement range array(*)* (I/O: I, type: numeric array)

Declare this parameter as a one dimensional array as shown in the following example:

```
DIM Range (x)
```

x = number of measurement units (1 to 8)

Elements of this array correspond to elements in the *measurement ch# array(*)*.

- *measurement value array(*)* (I/O: O, type: numeric array)

Declare this parameter as a two dimensional array as shown in the following example:

```
DIM Data (x,y)
```

x = number of measurement units (1 to 8) y = number of measurement data for each unit

y should be larger than the *number of steps* specified in the Set_iv subprogram. If y is larger, this subprogram sets y = *number of steps*. If y is smaller, all the measurement values cannot be stored in the array.

- *source value array(*)* (I/O: O, type: numeric array)

Declare this parameter as a one dimensional array as shown in the following example:

```
DIM S_data (y)
```

y = number of measurement data for each unit

y should be larger than the *number of steps* specified in the Set_iv subprogram. If y is larger, this subprogram sets y = *number of steps*. If y is smaller, all the measurement values cannot be stored in the array.

Example Statements

```
Sweep_miv (Channel(*), Mode(*), Range(*), Data(*), Sweep(*))
```

Note



When using this subprogram, it is recommended that you include OPTION BASE 1 at the beginning of your program. Including OPTION BASE 1 in your program causes array element numbering to start from 1 instead of 0. This makes it easier to keep track of the array elements in the *measurement value array(*)* and [*source value array(*)*] because the n th returned value is array element n instead of $n - 1$.

Sweep_mode

File: HP4142_DRV

This subprogram sets the *automatic sweep abort function* and *output after sweep* settings for sweep sources.

HP-IB Command:

WM

Syntax

`Sweep_mode ( automatic sweep abort function [ , output after sweep ] )`

Parameters

- *automatic sweep abort function* (I/O: I, type: integer)

1: OFF

2: ON

Initial = 1

- *output after sweep* (I/O: I, type: integer)

1: The start value is output after the sweep is completed.

2: The stop value is output after the sweep is completed.

Initial = 1

Default = 1

The *output after sweep* parameter is ignored for pulsed sweeps. Pulsed sweep sources always output the *base* value after the sweep is completed.

Example Statements

`Sweep_mode (2)`

`Sweep_mode (2, 2)`

`Sweep_mode (Stop_mode, End_set)`

Sweep_pbias

File: HP4142_DRV

This subprogram triggers a staircase sweep with pulsed bias measurement. The staircase sweep is output as determined by the Set_iv subprogram, and for each sweep step, a pulse is output as determined by the Pulse_i or Pulse_v subprogram. During each pulse output, a measurement is performed at the specified *measurement ch#*. Measurement values and sweep source values are returned to the *measurement value array(*)* and *source value array(*)*, respectively.

HP-IB Command:

MM, WNU?, FMT, FL, RV, RI, XE

Syntax

For Voltage Measurements:

```
Sweep_pbias ( measurement ch#, 1, V measurement range, measurement value array(*)
[ , source value array(*) ] )
```

For Current Measurements:

```
Sweep_pbias ( measurement ch#, 2, I measurement range, measurement value array(*)
[ , source value array(*) ] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	ch#
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

- *V measurement range* (I/O: I, type: real)

The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of SMUs, HVUs, or HCU is set to the Compliance range automatically, regardless of specified value.

0: 40 V range fixed
 2: 2 V range fixed
 20: 20 V range fixed
 40: 40 V range fixed

- *I measurement range* (I/O: I, type: real)

Sweep_pbias

0:	Compliance range
-1E-8:	10 nA range fixed
-1E-7:	100 nA range fixed
-1E-6:	1 μ A range fixed
-1E-5:	10 μ A range fixed
-1E-4:	100 μ A range fixed
-1E-3:	1 mA range fixed
-1E-2:	10 mA range fixed
-1E-1:	100 mA range fixed
-1:	1 A range fixed
-10:	10 A range fixed

- *measurement value array (*)* (I/O: O, type: numeric array, unit: V or I)

The measurement value of each sweep step is returned to this array. The number of elements of the *measurement value array(*)* must be larger than the number of sweep steps.

- *source value array (*)* (I/O: O, type: numeric array, unit: V or I)

The sweep source values of each sweep step are returned to this array. The number of elements of the *source value array(*)* must be larger than the number of sweep steps.

Example Statements

```
Sweep_pbias (1, 1, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_pbias (8, 2, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_pbias (Ch, Meas_mode, Range, Meas_data(*), Sweep_value(*))
```

Note



When using this subprogram, it is recommended that you include OPTION BASE 1 at the beginning of your program. Including OPTION BASE 1 in your program causes array element numbering to start from 1 instead of 0. This makes it easier to keep track of the array elements in the *measurement value array(*)* and *source value array(*)* because the *n*th returned value is array element *n* instead of *n - 1*.

Sweep_piv

File: HP4142_DRV

This subprogram triggers a pulsed sweep measurement. A pulsed sweep is output as determined by the Set_piv subprogram, measurements are made at the *measurement ch#* for each sweep step, and the measurement values and sweep source values are returned to *measurement value array(*)* and *source value array(*)*, respectively.

HP-IB Command:

MM, WNU?, FMT, FL, RV, RI, XE

Syntax

For Voltage Measurements:

```
Sweep_piv ( measurement ch#, 1, V measurement range, measurement value array(*),
           [ source value array(*) ] )
```

For Current Measurements:

```
Sweep_piv ( measurement ch#, 2, I measurement range, measurement value array(*),
           [ source value array(*) ] )
```

Parameters

- *measurement ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VM	1 to 8, 11 to 18 or 21 to 28

- *V measurement range* (I/O: I, type: real)

The *V measurement range* sets the voltage measurement range of VMs. The voltage measurement range of SMUs or HCU is set to the Compliance range automatically.

0: 40 V range fixed
 2: 2 V range fixed
 20: 20 V range fixed
 40: 40 V range fixed

- *I measurement range* (I/O: I, type: real)

0: Compliance range
 -1E-8: 10 nA range fixed
 -1E-7: 100 nA range fixed

Sweep_piv

-1E-6: 1 μ A range fixed
-1E-5: 10 μ A range fixed
-1E-4: 100 μ A range fixed
-1E-3: 1 mA range fixed
-1E-2: 10 mA range fixed
-1E-1: 100 mA range fixed
-1: 1 A range fixed
-10: 10 A range fixed

- *measurement value array (*)* (I/O: O, type: numeric array, unit: V or I)

The measurement value of each sweep step is returned to this array. The number of elements of the *measurement value array(*)* must be larger than the number of sweep steps.

- *source value array (*)* (I/O: O, type: numeric array, unit: V or I)

The sweep source values of each sweep step are returned to this array. The number of elements of the *source value array(*)* must be larger than the number of sweep steps.

Example Statements

```
Sweep_piv (1, 1, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_piv (Ch, Meas_mode, Range, Meas_data(*), Sweep_value(*))
```

```
Sweep_piv (4, 2, 0, Meas_data(*), Sweep_value(*))
```

```
Sweep_piv (Ch, Meas_mode, Range, Meas_data(*), Sweep_value(*))
```

Note



When using this subprogram, it is recommended that you include OPTION BASE 1 at the beginning of your program. Including OPTION BASE 1 in your program causes array element numbering to start from 1 instead of 0. This makes it easier to keep track of the array elements in the *measurement value array(*)* and *source value array(*)* because the *nth* returned value is array element *n* instead of *n - 1*.

Wbuild_file

File: GRAPHICS

This subprogram generates a measurement data file that conforms to the Basic Statistics and Data Manipulation (BSDM) format.

This subprogram converts the *data array*(*) that you set up to a BSDM format data file and stores the data file, *file comments*\$, and *variable names*\$ in the specified mass storage device. This subprogram can be used separately or with other subprograms to create a BSDM format data file. BSDM format data files can be evaluated by the HP 98820A Statistical Library. This software provides Histograms, Control Charts, Scattergrams, data storing, and many other statistical analysis routines.

Syntax

```
Wbuild_file ( file name, data array (*), file comments$, variable names$(*) )
```

Parameters

- *file name*\$ (I/O: I, type: string)

This parameter defines a file name of a file that is create by this subprogram. The input format is the same as HP BASIC format.

- *data array* (*) (I/O: I, type: numeric array)

Data that are specified by this parameter are stored in the file.

The *data array* (*) must be a two dimensional array with a lower boundary (1, 1). If OPTION BASE 1 is included in your program, this condition is automatically set up. If OPTION BASE 1 is not included in your program, you must set up *data array*(*) so that this condition is met.

The first dimension of this array represents the number of variables, and the number of elements must be within 50. The second dimension of this array represents the number of observations for each variable.

The total number of variables must be within 1500.

- *file comments*\$ (I/O: I, type: string)

This parameter specifies the title or comments for the data.

Max. 80 characters are allowed.

- *variable names*\$ (I/O: I, type: string)

Variable names.

Max. 50 variable names with max. 10 characters for each variable name.

These variable names correspond to elements in *data array*(*).

Wbuild_file

Example Statements

```
Wbuild_file ("DATA_1", M_data(*), "MEASUREMENT DATA", Var_name$(*))
```

```
Wbuild_file (File$, all_data(*), Title$, Name$(*))
```

Zero_output

File: HP4142_DRV

This subprogram sets the specified SMUs, HVUs, HCU, or VSs to Zero Output.

HP-IB Command:

DZ

Syntax

```
Zero_output ( [ ch#] [ , ch#] [ , ch#] [ , ch#] [ , ch#] [ , ch#] [ , ch#] [ , ch#])
```

Parameters

- *ch#* (I/O: I, type: integer)

Unit	<i>ch#</i>
HPSMU	2 to 8
MPSMU	1 to 8
HVU	2 to 8
HCU	2 to 8
VS	1 to 8, 11 to 18, or 21 to 28

Default = all units

Example Statements

```
Zero_output (1)
```

```
Zero_output (Ch1)
```

```
Zero_output (Ch1,Ch2,Ch3,Ch4)
```

```
Zero_output
```


Program Listings

This appendix contains the source code for all the control software sub-programs and user-defined functions.

HP4142_DRV File

Auto_cal

```

35  SUB Auto_cal(INTEGER Cal_mode)
40  COM @Hp4142
45  !*****
50  !*          Set auto calibration mode          *
55  !*****
60  OUTPUT @Hp4142;"CM";Cal_mode
65  OUTPUT @Hp4142;"*OPC?"
70  !
75  ENTER @Hp4142;A$
80  S=SPOLL(@Hp4142)
85  IF BIT(S,5) THEN CALL Detect_error(S,"Auto_cal")
90  SUBEND

```

Cal_hp4142

```

105 SUB Cal_hp4142(OPTIONAL INTEGER Channel)
110 COM @Hp4142
115 !*****
120 !*          Execute calibration          *
125 !*****
130 IF NPAR=0 THEN OUTPUT @Hp4142;"CA"
135 IF NPAR=1 THEN OUTPUT @Hp4142;"CA";Channel
140 OUTPUT @Hp4142;"*OPC?"
145 !
150 ENTER @Hp4142;A$
155 S=SPOLL(@Hp4142)
160 IF BIT(S,5) THEN CALL Detect_error(S,"Cal_hp4142")
165 SUBEND

```

Ch_sw_on

```
180 SUB Ch_sw_on(OPTIONAL INTEGER Channel1,Channel2,Channel3,Channel4,Channel5,Channel6,Channel7,Channel8)
185 COM @Hp4142
190 !*****
195 !*          Connect the output switch          *
200 !*****
205 SELECT NPAR
210 CASE 0
215     OUTPUT @Hp4142;"CW"
220 CASE 1
225     OUTPUT @Hp4142;"CW";Channel1
230 CASE 2
235     OUTPUT @Hp4142;"CW";Channel1;",";Channel2
240 CASE 3
245     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3
250 CASE 4
255     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3;",";Channel4
260 CASE 5
265     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5
270 CASE 6
275     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6
280 CASE 7
285     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;
    ",";Channel7
290 CASE 8
295     OUTPUT @Hp4142;"CW";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;
    ",";Channel7;",";Channel8
300 END SELECT
305 !
310 OUTPUT @Hp4142;"*OPC?"
315 ENTER @Hp4142;A$
320 S=SPOLL(@Hp4142)
325 IF BIT(S,5) THEN CALL Detect_error(S,"Ch_sw_on")
330 SUBEND
```

Ch_sw_off

```
345  SUB Ch_sw_off(OPTIONAL INTEGER Channel1,Channel2,Channel3,Channel4,Channel5,Channel6,Channel7,
Channel8)
350    COM @Hp4142
355    !*****
360    !*          Disconnect the channel switch          *
365    !*****
370    SELECT NPAR
375    CASE 0
380        OUTPUT @Hp4142;"CL"
385    CASE 1
390        OUTPUT @Hp4142;"CL";Channel1
395    CASE 2
400        OUTPUT @Hp4142;"CL";Channel1;",";Channel2
405    CASE 3
410        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3
415    CASE 4
420        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3;",";Channel4
425    CASE 5
430        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5
435    CASE 6
440        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6
445    CASE 7
450        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;
",";Channel7
455    CASE 8
460        OUTPUT @Hp4142;"CL";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;
",";Channel7;",";Channel8
465    END SELECT
470    !
475    OUTPUT @Hp4142;"*OPC?"
480    ENTER @Hp4142;A$
485    S=SPOLL(@Hp4142)
490    IF BIT(S,5) THEN CALL Detect_error(S,"Ch_sw_off")
495  SUBEND
```

Zero_output

```
510 SUB Zero_output(OPTIONAL INTEGER Channel1,Channel2,Channel3,Channel4,Channel5,Channel6,Channel7,  
Channel8)  
515   COM @Hp4142  
520   !*****  
525   !*      Set the SMU or VS to zero output state      *  
530   !*****  
535   SELECT NPAR  
540   CASE 0  
545       OUTPUT @Hp4142;"DZ"  
550   CASE 1  
555       OUTPUT @Hp4142;"DZ";Channel1  
560   CASE 2  
565       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2  
570   CASE 3  
575       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3  
580   CASE 4  
585       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3;"",Channel4  
590   CASE 5  
595       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3;"",Channel4;"",Channel5  
600   CASE 6  
605       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3;"",Channel4;"",Channel5;"",Channel6  
610   CASE 7  
615       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3;"",Channel4;"",Channel5;"",Channel6;  
        "",Channel7  
620   CASE 8  
625       OUTPUT @Hp4142;"DZ";Channel1;"",Channel2;"",Channel3;"",Channel4;"",Channel5;"",Channel6;  
        "",Channel7;"",Channel8  
630   END SELECT  
635   !  
640   OUTPUT @Hp4142;"*OPC?"  
645   ENTER @Hp4142;A$  
650   S=SPOLL(@Hp4142)  
655   IF BIT(S,5) THEN CALL Detect_error(S,"Zero_output")  
660 SUBEND
```

Recover_output

```
675 SUB Recover_output(OPTIONAL INTEGER Channel1,Channel2,Channel3,Channel4,Channel5,Channel6,Channel7,  
Channel8)  
680 COM @Hp4142  
685 !*****  
690 !* Recover the SMU or VS from zero output state *  
695 !*****  
700 SELECT NPAR  
705 CASE 0  
710 OUTPUT @Hp4142;"RZ"  
715 CASE 1  
720 OUTPUT @Hp4142;"RZ";Channel1  
725 CASE 2  
730 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2  
735 CASE 3  
740 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3  
745 CASE 4  
750 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3;",";Channel4  
755 CASE 5  
760 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5  
765 CASE 6  
770 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6  
775 CASE 7  
780 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;  
",";Channel7  
785 CASE 8  
790 OUTPUT @Hp4142;"RZ";Channel1;",";Channel2;",";Channel3;",";Channel4;",";Channel5;",";Channel6;  
",";Channel7;",";Channel8  
795 END SELECT  
800 !  
805 OUTPUT @Hp4142;"*OPC?"  
810 ENTER @Hp4142;A$  
815 S=SPOLL(@Hp4142)  
820 IF BIT(S,5) THEN CALL Detect_error(S,"Recover_output")  
825 SUBEND
```

Set_smu

```
840 SUB Set_smu(INTEGER Average_time)
845 COM @Hp4142
850 !*****
855 !*          Set averaging time          *
860 !*****
865 OUTPUT @Hp4142;"AV";Average_time
870 OUTPUT @Hp4142;"*OPC?"
875 !
880 ENTER @Hp4142;A$
885 S=SPOLL(@Hp4142)
890 IF BIT(S,5) THEN CALL Detect_error(S,"Set_smu")
895 SUBEND
```

Init_hp4142

```
910 SUB Init_hp4142
915 COM @Hp4142
920 !*****
925 !*          4142B initial setting          *
930 !*****
935 OUTPUT @Hp4142;"*RST"      ! Reset 4142B
940 OUTPUT @Hp4142;"*SRE232"   ! Set status byte: bit7,6,5,3
945 OUTPUT @Hp4142;"TM2"      ! Set IE,TV,TI trigger mode
950 OUTPUT @Hp4142;"*OPC?"
955 !
960 ENTER @Hp4142;A$
965 S=SPOLL(@Hp4142)
970 IF BIT(S,5) THEN CALL Detect_error(S,"Init_hp4142")
975 SUBEND
```

Self_test

```
990 SUB Self_test(INTEGER Test_type,Result,OPTIONAL Error$)
995 COM @Hp4142
1000 !*****
1005 !*          Execute the self test          *
1010 !*****
1015 OUTPUT @Hp4142;"*TST?";Test_type
1020 !
1025 SELECT Test_type
1030 CASE 0 TO 9
1035     ENTER @Hp4142;Result
1040 CASE ELSE
1045     S=SPOLL(@Hp4142)
1050     CALL Detect_error(S,"Self_test")
1055 END SELECT
1060 !*****
1065 !*          Check fail code          *
1070 !*****
1075 IF NPAR=3 THEN
1080     ALLOCATE E$[60]
1085     IF Result THEN
1090         BEEP
1095         SELECT Test_type
1100         CASE 1 TO 8
1105             FOR I=0 TO 7
1110                 IF BIT(Result,I) THEN Error$="Channel "&VAL$(I+1)&" failed on self test"
1115             NEXT I
1120         CASE 9
1125             IF BIT(Result,8) THEN Error$="A/D or CPU failed on self test"
1130         CASE 0
1135             FOR I=0 TO 7
1140                 IF BIT(Result,I) THEN E$=E$&VAL$(I+1)
1145                 IF BIT(Result,I+1) THEN E$=E$&" ,"
1150             NEXT I
1155             IF BIT(Result,8) THEN E$=E$&" A/D or CPU"
1160             SELECT BIT(Result,0)
1165             CASE 0
1170                 Error$="Channel "&TRIM$(E$[2,60])&" failed on self test."
1175             CASE 1
1180                 Error$="Channel "&TRIM$(E$)&" failed on self test."
1185             END SELECT
1190         END SELECT
1195     IF BIT(Result,9) OR BIT(Result,10) THEN
1200         Error$="Interlock open. "&Error$
1205     END IF
1210 ELSE
1215     Error$="No Error"
1220 END IF
1225 END IF
1230 SUBEND
```

Force_i

```
1245 SUB Force_i(INTEGER Channel,REAL Current,OPTIONAL Range,V_compliance,INTEGER Polarity)
1250   COM @Hp4142
1255   !*****
1260   !*           Set output current           *
1265   !*****
1270   Module_type((Channel),Type$)
1275   IF Type$="HP41424" THEN
1280     DISP "Force_i: HP41424 is not specified for current source"
1285     BEEP
1290     STOP
1295   ELSE
1300     IF NPAR>2 THEN
1305       Rng=Range
1310       IF Rng<>0 THEN CALL Set_i_range(Rng)
1315     END IF
1320     !
1325     SELECT NPAR
1330     CASE 2
1335       OUTPUT @Hp4142;"DI";Channel;"",0,"";Current
1340     CASE 3
1345       OUTPUT @Hp4142;"DI";Channel;"",Rng,"";Current
1350     CASE 4
1355       OUTPUT @Hp4142;"DI";Channel;"",Rng,"";Current,"";V_compliance
1360     CASE 5
1365       OUTPUT @Hp4142;"DI";Channel;"",Rng,"";Current,"";V_compliance,"";Polarity
1370     END SELECT
1375   END IF
1380   !
1385   OUTPUT @Hp4142;"*OPC?"
1390   ENTER @Hp4142;A$
1395   S=SPOLL(@Hp4142)
1400   IF BIT(S,5) THEN CALL Detect_error(S,"Force_i")
1405 SUBEND
```

Force_v

```
1420 SUB Force_v(INTEGER Channel,REAL Voltage,OPTIONAL Range,I_compliance,INTEGER Polarity)
1425   COM @Hp4142
1430   !*****
1435   !*           Set output voltage           *
1440   !*****
1445   IF NPAR>2 THEN
1450     Rng=Range
1455     IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
1460   END IF
1465   !
1470   SELECT NPAR
1475   CASE 2
1480     OUTPUT @Hp4142;"DV";Channel;"",0,"";Voltage
1485   CASE 3
1490     OUTPUT @Hp4142;"DV";Channel;"",Rng,"";Voltage
1495   CASE 4
1500     OUTPUT @Hp4142;"DV";Channel;"",Rng,"";Voltage;"",I_compliance
1505   CASE 5
1510     OUTPUT @Hp4142;"DV";Channel;"",Rng,"";Voltage;"",I_compliance;"",Polarity
1515   END SELECT
1520   !
1525   OUTPUT @Hp4142;"*OPC?"
1530   ENTER @Hp4142;A$
1535   S=SPOLL(@Hp4142)
1540   IF BIT(S,5) THEN CALL Detect_error(S,"Force_v")
1545 SUBEND
```

Measure_i

```
1560 SUB Measure_i(INTEGER Channel,REAL Current,OPTIONAL REAL Range,INTEGER Status)
1565   COM @Hp4142
1570   !*****
1575   !* Spot measurement will execute -current value- *
1580   !*****
1585   IF NPAR>2 THEN
1590     Rng=Range
1595     IF Rng<>0 THEN CALL Set_i_range(Rng)
1600   END IF
1605   !
1610   SELECT NPAR
1615   CASE 2
1620     OUTPUT @Hp4142;"FMT2;TI";Channel;",";0"
1625   CASE 3
1630     OUTPUT @Hp4142;"FMT2;TI";Channel;",";Rng
1635   CASE 4
1640     OUTPUT @Hp4142;"FMT1;TI";Channel;",";Rng
1645   END SELECT
1650   !
1655   OUTPUT @Hp4142;"*OPC?"
1660   ENTER @Hp4142;A$
1665   S=SPOLL(@Hp4142)
1670   IF BIT(S,5) THEN CALL Detect_error(S,"Measure_i")
1675   !*****
1680   !*           Get measured data           *
1685   !*****
1690   IF NPAR<4 THEN
1695     ENTER @Hp4142;Current
1700   ELSE
1705     ENTER @Hp4142;Data$
1710     Current=VAL(Data$[4;12])
1715     Status$=Data$[1;1]
1720     IF Status$="N" THEN Status=0
1725     IF Status$="T" THEN Status=1
1730     IF Status$="C" THEN Status=2
1735     IF Status$="I" THEN Status=3
1740     IF Status$="V" THEN Status=4
1745     IF Status$="F" THEN Status=7
1750   END IF
1755   OUTPUT @Hp4142;"FMT1"
1760 SUBEND
```

Measure_v

```
1775 SUB Measure_v(INTEGER Channel,REAL Voltage,OPTIONAL Range,INTEGER Status)
1780   COM @Hp4142
1785   !*****
1790   !* Spot measurement will execute -voltage value- *
1795   !*****
1800   IF NPAR>2 THEN
1805     Rng=Range
1810     IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
1815   END IF
1820   !
1825   SELECT NPAR
1830   CASE 2
1835     OUTPUT @Hp4142;"FMT2;TV";Channel;"",0"
1840   CASE 3
1845     OUTPUT @Hp4142;"FMT2;TV";Channel;"",Rng
1850   CASE 4
1855     OUTPUT @Hp4142;"FMT1;TV";Channel;"",Rng
1860   END SELECT
1865   !
1870   OUTPUT @Hp4142;"*OPC?"
1875   ENTER @Hp4142;A$
1880   S=SPOLL(@Hp4142)
1885   IF BIT(S,5) THEN CALL Detect_error(S,"Measure_v")
1890   !*****
1895   !*           Get measured data           *
1900   !*****
1905   IF NPAR<4 THEN
1910     ENTER @Hp4142;Voltage
1915   ELSE
1920     ENTER @Hp4142;Data$
1925     Voltage=VAL(Data$[4;12])
1930     Status$=Data$[1;1]
1935     IF Status$="N" THEN Status=0
1940     IF Status$="T" THEN Status=1
1945     IF Status$="C" THEN Status=2
1950     IF Status$="I" THEN Status=3
1955     IF Status$="V" THEN Status=4
1960     IF Status$="F" THEN Status=7
1965   END IF
1970   OUTPUT @Hp4142;"FMT1"
1975 SUBEND
```

Set_vm

```
1990 SUB Set_vm(INTEGER Channel,INTEGER Vm_mode)
1995   COM @Hp4142
2000   !*****
2005   !*               Set VM operation mode               *
2010   !*****
2015   OUTPUT @Hp4142;"VM";Channel;" ";Vm_mode
2020   OUTPUT @Hp4142;"*OPC?"
2025   ENTER @Hp4142;A$
2030   S=SPOLL(@Hp4142)
2035   IF BIT(S,5) THEN CALL Detect_error(S,"Set_vm")
2040 SUBEND
```

Set_iv

```

2055 SUB Set_iv(INTEGER Channel,Swp_mode,REAL Range,Start,Stop,INTEGER No_step,OPTIONAL REAL H_time,
D_time,Iv_comp,Pw_comp)
2060 COM @Hp4142
2065 !*****
2070 !*          Set voltage sweep parameter          *
2075 !*****
2080 Rng=Range
2085 IF ABS(Swp_mode)=1 OR ABS(Swp_mode)=3 THEN
2090     IF Swp_mode=1 THEN Smode=1          ! Linear sweep ( single stair )
2095     IF Swp_mode=-1 THEN Smode=2        ! Log sweep   ( single stair )
2100     IF Swp_mode=3 THEN Smode=3          ! Linear sweep ( double stairs )
2105     IF Swp_mode=-3 THEN Smode=4         ! Log sweep   ( double stairs )
2110     IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
2115     !
2120     SELECT NPAR
2125     CASE 6
2130         OUTPUT @Hp4142;"WV";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step
2135     CASE 7
2140         OUTPUT @Hp4142;"WV";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"WT";H_time;
2145         "0"
2145     CASE 8
2150         OUTPUT @Hp4142;"WV";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"WT";H_time;
2155         "";D_time
2155     CASE 9
2160         OUTPUT @Hp4142;"WV";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"";Iv_comp;
2165         "WT";H_time;"";D_time
2165     CASE 10
2170         OUTPUT @Hp4142;"WV";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"";Iv_comp;
2175         "";Pw_comp;"WT";H_time;"";D_time
2175     END SELECT
2180 END IF
2185 !*****
2190 !*          Set current sweep parameter          *
2195 !*****
2200 IF ABS(Swp_mode)=2 OR ABS(Swp_mode)=4 THEN
2205     IF Swp_mode=2 THEN Smode=1          ! Linear sweep ( single stair )
2210     IF Swp_mode=-2 THEN Smode=2         ! Log sweep   ( single stair )
2215     IF Swp_mode=4 THEN Smode=3          ! Linear sweep ( double stairs )
2220     IF Swp_mode=-4 THEN Smode=4         ! Log sweep   ( double stairs )
2225     Module_type((Channel),Type$)
2230     !
2235     IF Type$="HP41424" THEN
2240         DISP "Set_iv: HP41424 is not specified for current source"
2245         BEEP
2250         STOP
2255     ELSE
2260         IF Rng<>0 THEN CALL Set_i_range(Rng)
2265         SELECT NPAR
2270         CASE 6
2275             OUTPUT @Hp4142;"WI";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step
2280         CASE 7
2285             OUTPUT @Hp4142;"WI";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"WT";H_time;
2290             "0"
2290         CASE 8
2295             OUTPUT @Hp4142;"WI";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"WT";H_time;
2300             "";D_time
2300         CASE 9
2305             OUTPUT @Hp4142;"WI";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"";Iv_comp;
2310             "WT";H_time;"";D_time
2310         CASE 10
2315             OUTPUT @Hp4142;"WI";Channel,"";Smode,"";Rng,"";Start,"";Stop,"";No_step;"";Iv_comp;
2320             "";Pw_comp;"WT";H_time;"";D_time

```

```
2320      END SELECT
2325      END IF
2330      END IF
2335      !
2340      OUTPUT @Hp4142;"*OPC?"
2345      ENTER @Hp4142;A$
2350      S=SPOLL(@Hp4142)
2355      IF BIT(S,5) THEN CALL Detect_error(S,"Set_iv")
2360      SUBEND
```

Sweep_iv

```

2375 SUB Sweep_iv(INTEGER Channel,Meas_mode,REAL Range,Meas_data(*),OPTIONAL REAL Sweep_value(*))
2380   OPTION BASE 1
2385   COM @Hp4142
2390   !*****
2395   !* Sweep measurement will execute -current & voltage- *
2400   !*****
2405   Data_num=SIZE(Meas_data,1)           ! check Meas_data(*) size
2410   OUTPUT @Hp4142;"MM2,";Channel
2415   OUTPUT @Hp4142;"WNU?"              ! check number of sweep step
2420   ENTER @Hp4142;No_step
2425   IF No_step<=Data_num THEN Data_num=No_step
2430   !
2435   IF NPAR=4 THEN OUTPUT @Hp4142;"FMT1,0"   ! get only measured data
2440   IF NPAR=5 THEN OUTPUT @Hp4142;"FMT1,1"   ! get measured and sweep data
2445   IF NPAR=4 THEN ALLOCATE Data$(No_step*16)
2450   IF NPAR=5 THEN ALLOCATE Data$[(No_step*16)*2]
2455   !*****
2460   !*           Set the measurement parameters           *
2465   !*****
2470   Rng=Range
2475   Module_type((Channel),Type$)
2480   SELECT Meas_mode
2485   CASE 1
2490     IF Type$="HP41424" THEN
2495       IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
2500       OUTPUT @Hp4142;"RV";Channel;",";Rng
2505     END IF
2510   CASE 2
2515     IF Type$="HP41424" THEN
2520       DISP "Sweep_iv: HP41424 is not specified for current measure"
2525       BEEP
2530       STOP
2535     ELSE
2540       IF Rng<>0 THEN CALL Set_i_range(Rng)
2545       OUTPUT @Hp4142;"RI";Channel;",";Rng
2550     END IF
2555   CASE ELSE
2560     DISP "Sweep_iv: Illegal measurement mode value"
2565     BEEP
2570     STOP
2575   END SELECT
2580   !*****
2585   !*           Start sweep and get the measured value           *
2590   !*****
2595   OUTPUT @Hp4142;"XE"
2600   OUTPUT @Hp4142;"*OPC?"
2605   ENTER @Hp4142;A$
2610   S=SPOLL(@Hp4142)
2615   IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_iv")
2620   ENTER @Hp4142;Data$
2625   !*****
2630   !*           Return only measured value           *
2635   !*****
2640   IF NPAR=4 THEN
2645     Pointer=1
2650     FOR I=1 TO Data_num*16 STEP 16
2655       Meas$=Data$(I,15)
2660       IF Meas$(1,1)<>"N" THEN GOSUB Error_disp
2665       Meas_data(Pointer)=VAL(Meas$(4,12))
2670       Pointer=Pointer+1
2675     NEXT I
2680   SUBEXIT

```

```

2685 END IF
2690 !*****
2695 !*      Return both measured and source value      *
2700 !*****
2705 IF NPAR=5 THEN
2710   Pointer=1
2715   FOR I=1 TO (Data_num*16)*2 STEP 32
2720     Meas$=Data$[I;15]
2725     Sweep$=Data$[I+16;15]
2730     IF Meas$[1;1]<>"N" THEN GOSUB Error_disp
2735     Meas_data(Pointer)=VAL(Meas$[4;12])
2740     Sweep_value(Pointer)=VAL(Sweep$[4;12])
2745     Pointer=Pointer+1
2750   NEXT I
2755 END IF
2760 SUBEXIT
2765 !
2770 Error_disp: !
2775   DISP "Sweep_iv: ";
2780   SELECT Meas$[1;1]
2785   CASE "T"
2790     DISP CHR$(129);" Another channel has reached compliance ";CHR$(128)
2795   CASE "C"
2800     DISP CHR$(129);" This channel has reached compliance ";CHR$(128)
2805   CASE "X"
2810     DISP CHR$(129);" This channel is oscillating ";CHR$(128)
2815   CASE "V"
2820     DISP CHR$(129);" Measurement data overflow ";CHR$(128)
2825   CASE "F"
2830     DISP CHR$(129);" HVU is not settled ";CHR$(128)
2835   END SELECT
2840   RETURN
2845   !
2850 SUBEND

```

Sweep_miv

```

2865 SUB Sweep_miv(INTEGER Channel(*),Meas_mode(*),REAL Range(*),Meas_data(*),OPTIONAL Sweep_value(*))
2870   OPTION BASE 1
2875   COM @Hp4142
2880   !*****
2885   !*           Set the measurement channel           *
2890   !*****
2895   Meas_channel=SIZE(Channel,1)           ! check meas channel number
2900   !
2905   SELECT Meas_channel
2910   CASE 1
2915       OUTPUT @Hp4142;"MM2,";Channel(1)
2920   CASE 2
2925       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2)
2930   CASE 3
2935       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2);",";Channel_(3)
2940   CASE 4
2945       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2);",";Channel(3);",";Channel(4)
2950   CASE 5
2955       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2);",";Channel(3);",";Channel(4);",";Channel(5)
2960   CASE 6
2965       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2);",";Channel(3);",";Channel(4);",";Channel(5);
2970       "",";Channel(6)
2975   CASE 7
2980       OUTPUT @Hp4142;"MM2,";Channel(1);",";Channel(2);",";Channel(3);",";Channel(4);",";Channel(5);
2985       "",";Channel(6);",";Channel(7);",";Channel(8)
2990   END SELECT
2995   !
3000   OUTPUT @Hp4142;"*OPC?"
3005   ENTER @Hp4142;A$
3010   S=SPOLL(@Hp4142)
3015   IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_miv")
3020   !*****
3025   !*           Initial setting           *
3030   !*****
3035   Data_num=SIZE(Meas_data,2)           ! check Meas_data(*) size
3040   OUTPUT @Hp4142;"WNU?"           ! check number of sweep step
3045   ENTER @Hp4142;No_step
3050   IF No_step<=Data_num THEN Data_num=No_step
3055   ALLOCATE Status$(Meas_channel,Data_num)[1]
3060   !
3065   FOR I=1 TO Meas_channel
3070       FOR J=1 TO Data_num
3075           Status$(I,J)="N"
3080       NEXT J
3085   NEXT I
3090   IF NPAR=4 THEN ALLOCATE Data_buffer$[(Meas_channel*16)*No_step]
3095   IF NPAR=5 THEN ALLOCATE Data_buffer$[((Meas_channel+1)*16)*No_step]
3100   IF NPAR=4 THEN OUTPUT @Hp4142;"FMT1,0"           ! get only measured data
3105   IF NPAR=5 THEN OUTPUT @Hp4142;"FMT1,1"           ! get measured and sweep data
3110   !*****
3115   !*           Set the ranging parameters for SMU           *
3120   !*****
3125   FOR I=1 TO Meas_channel
3130       Module_type((Channel(I)),Type$)
3135       Rng=Range(I)
3140       !
3145       SELECT Meas_mode(I)
3150       CASE 1
3155           IF Rng<>0 THEN CALL Set_v_range((Channel(I)),Rng)

```

```

3160     IF Type$="HP41424" THEN OUTPUT @Hp4142;"RV";Channel(I);",";Rng
3165 CASE 2
3170     IF Type$="HP41424" THEN
3175         DISP "Sweep_miv: HP41424 is not specified for current measure"
3180         BEEP
3185         STOP
3190     ELSE
3195         IF Rng<>0 THEN CALL Set_i_range(Rng)
3200         OUTPUT @Hp4142;"RI";Channel(I);",";Rng
3205     END IF
3210 CASE ELSE
3215     DISP "Sweep_miv: Illegal Measurement mode value"
3220     BEEP
3225     STOP
3230 END SELECT
3235 !
3240 OUTPUT @Hp4142;"*OPC?"
3245 ENTER @Hp4142;A$
3250 S=SPOLL(@Hp4142)
3255 IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_miv")
3260 NEXT I
3265 !*****
3270 !*      Start sweep and get the measured value      *
3275 !*****
3280 OUTPUT @Hp4142;"XE"
3285 OUTPUT @Hp4142;"*OPC?"
3290 ENTER @Hp4142;A$
3295 S=SPOLL(@Hp4142)
3300 IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_miv")
3305 ENTER @Hp4142;Data_buffer$
3310 !*****
3315 !*      Return only measured value      *
3320 !*****
3325 Pointer=1
3330 Chan=1
3335 IF NPAR=4 THEN
3340     FOR I=1 TO (Meas_channel*16)*Data_num STEP Meas_channel*16
3345         Chan=1
3350         FOR J=1 TO Meas_channel*16 STEP 16
3355             Meas$=Data_buffer$(I+(J-1);15)
3360             IF Meas$[1;1]<>"N" THEN Status$(Chan,Pointer)=Meas$[1;1]
3365             Meas_data(Chan,Pointer)=VAL(Meas$[4;12])
3370             Chan=Chan+1
3375         NEXT J
3380         Pointer=Pointer+1
3385     NEXT I
3390 END IF
3395 !*****
3400 !*      Return both measured and source value      *
3405 !*****
3410 IF NPAR=5 THEN
3415     FOR I=1 TO ((Meas_channel+1)*16)*Data_num STEP (Meas_channel+1)*16
3420         Chan=1
3425         FOR J=1 TO Meas_channel*16 STEP 16
3430             Meas$=Data_buffer$(I+(J-1);15)
3435             IF Meas$[1;1]<>"N" THEN Status$(Chan,Pointer)=Meas$[1;1]
3440             Meas_data(Chan,Pointer)=VAL(Meas$[4;12])
3445             Chan=Chan+1
3450         NEXT J
3455         Sweep$=Data_buffer$(I+(J-1);15)
3460         Sweep_value(Pointer)=VAL(Sweep$[4;12])
3465         Pointer=Pointer+1
3470     NEXT I
3475 END IF
3480 !*****
3485 !*      Error channel detect ... Display error message      *

```

```

3490      !*****
3495      FOR I=1 TO Meas_channel
3500          FOR J=1 TO Data_num
3505              IF Status$(I,J)<>"I" THEN
3510                  GOSUB Error_disp
3515              END IF
3520          NEXT J
3525      NEXT I
3530      SUBEXIT
3535      !
3540      Error_disp:      !
3545          DISP "Sweep_miv: ";CHR$(129);" Channel";Channel(I);":";
3550          SELECT Status$(I,J)
3555          CASE "T"
3560              DISP " Another channel has reached compliance ";CHR$(128)
3565          CASE "C"
3570              DISP " This channel has reached compliance ";CHR$(128)
3575          CASE "X"
3580              DISP " This channel is oscillating ";CHR$(128)
3585          CASE "V"
3590              DISP " Measurement data overflow ";CHR$(128)
3595          CASE "F"
3600              DISP " HVU is not settled ";CHR$(128)
3605          END SELECT
3610          RETURN
3615      !
3620      SUBEND

```

Sweep_mode

```
3635 SUB Sweep_mode(INTEGER Return_mode,OPTIONAL INTEGER Stop_mode)
3640 COM @Hp4142
3645 !*****
3650 !*          Set sweep operation mode          *
3655 !*****
3660 SELECT NPAR
3665 CASE 1
3670     OUTPUT @Hp4142;"WM";Return_mode
3675 CASE 2
3680     OUTPUT @Hp4142;"WM";Return_mode;" ";Stop_mode
3685 END SELECT
3690 !
3695 OUTPUT @Hp4142;"*OPC?"
3700 ENTER @Hp4142;A$
3705 S=SPOLL(@Hp4142)
3710 IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_mode")
3715 SUBEND
```

Dpulse_v

```
3730 SUB Dpulse_v(INTEGER Channel,REAL Range,Pbase,Pvalue,OPTIONAL Compliance)
3735 COM @Hp4142
3740 !*****
3745 !*          Setup the pulse voltage measurement parameter      *
3750 !*****
3755 Rng=Range
3760 Module_type((Channel),Type$)
3765 IF Type$="HP41424" THEN
3770     DISP "Dpulse_v: HP41424 is not specified for 2ch pulse channel"
3775     BEEP
3780     STOP
3785 ELSE
3790     IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
3795     SELECT NPAR
3800     CASE 4
3805         OUTPUT @Hp4142;"PDV";Channel;" ";Rng;" ";Pbase;" ";Pvalue
3810     CASE 5
3815         OUTPUT @Hp4142;"PDV";Channel;" ";Rng;" ";Pbase;" ";Pvalue;" ";Compliance
3820     END SELECT
3825 END IF
3830 !
3835 OUTPUT @Hp4142;"*OPC?"
3840 ENTER @Hp4142;A$
3845 S=SPOLL(@Hp4142)
3850 IF BIT(S,5) THEN CALL Detect_error(S,"Dpulse_v")
3855 SUBEND
```

Pulse_v

```
3870 SUB Pulse_v(INTEGER Channel,REAL Range,Pbase,Pvalue,Pwidth,Htime,OPTIONAL Period,Compliance)
3875   COM @Hp4142
3880   !*****
3885   !*   Setup the pulse voltage measurement parameter   *
3890   !*****
3895   Rng=Range
3900   IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
3905   SELECT NPAR
3910   CASE 6
3915     OUTPUT @Hp4142;"PV";Channel;"",Rng;"",Pbase;"",Pvalue;"PT";Htime;"",Pwidth
3920   CASE 7
3925     OUTPUT @Hp4142;"PV";Channel;"",Rng;"",Pbase;"",Pvalue;"PT";Htime;"",Pwidth;"",Period
3930   CASE 8
3935     OUTPUT @Hp4142;"PV";Channel;"",Rng;"",Pbase;"",Pvalue;"",Compliance;"PT";Htime;"",Pwidth;
3940     "","",Period
3945     !
3950     OUTPUT @Hp4142;"*OPC?"
3955     ENTER @Hp4142;A$
3960     S=SPOLL(@Hp4142)
3965     IF BIT(S,5) THEN CALL Detect_error(S,"Pulse_v")
3970 SUBEND
```

Dpulse_i

```
3985 SUB Dpulse_i(INTEGER Channel,REAL Range,Pbase,Pvalue,OPTIONAL Compliance)
3990   COM @Hp4142
3995   !*****
4000   !*   Setup the pulse current measurement parameter   *
4005   !*****
4010   Rng=Range
4015   Module_type((Channel),Type$)
4020   IF Type$="HP41424" THEN
4025     DISP "Dpulse_i: HP41424 is not specified for 2ch pulse channel"
4030     BEEP
4035     STOP
4040   ELSE
4045     IF Rng<>0 THEN CALL Set_i_range(Rng)
4050     SELECT NPAR
4055     CASE 4
4060       OUTPUT @Hp4142;"PDI";Channel;"",Rng;"",Pbase;"",Pvalue
4065     CASE 5
4070       OUTPUT @Hp4142;"PDI";Channel;"",Rng;"",Pbase;"",Pvalue;"",Compliance
4075     END SELECT
4080   END IF
4085   !
4090   OUTPUT @Hp4142;"*OPC?"
4095   ENTER @Hp4142;A$
4100   S=SPOLL(@Hp4142)
4105   IF BIT(S,5) THEN CALL Detect_error(S,"Dpulse_i")
4110 SUBEND
```

Pulse_i

```
4125 SUB Pulse_i(INTEGER Channel,REAL Range,Pbase,Pvalue,Pwidth,Htime,OPTIONAL Period,Compliance)
4130   COM @Hp4142
4135   !*****
4140   !*   Setup the pulse current measurement parameter   *
4145   !*****
4150   Rng=Range
4155   Module_type((Channel),Type$)
4160   IF Type$="HP41424" THEN
4165       DISP "Pulse_i: HP41424 is not specified for current source"
4170       BEEP
4175       STOP
4180   ELSE
4185       IF Rng<>0 THEN CALL Set_i_range(Rng)
4190       SELECT NPAR
4195       CASE 6
4200           OUTPUT @Hp4142;"PI";Channel;"",Rng;"",Pbase;"",Pvalue;"PT";Htime;"",Pwidth
4205       CASE 7
4210           OUTPUT @Hp4142;"PI";Channel;"",Rng;"",Pbase;"",Pvalue;"PT";Htime;"",Pwidth;"",Period
4215       CASE 8
4220           OUTPUT @Hp4142;"PI";Channel;"",Rng;"",Pbase;"",Pvalue;"",Compliance;"PT";Htime;"",Pwidth;
",",Period
4225       END SELECT
4230   END IF
4235   !
4240   OUTPUT @Hp4142;"*OPC?"
4245   ENTER @Hp4142;A$
4250   S=SPOLL(@Hp4142)
4255   IF BIT(S,5) THEN CALL Detect_error(S,"Pulse_i")
4260 SUBEND
```

Dpulse_measure

```

4275 SUB Dpulse_measure(INTEGER Channel,Meas_mode,REAL Meas_data,OPTIONAL INTEGER Primary,REAL Range,
INTEGER Status)
4280 COM @Hp4142
4285 !*****
4290 !* Performs the 2CH pulsed spot measurement -current & voltage - *
4295 !*****
4300 Rng=0
4305 IF NPAR>4 THEN Rng=Range
4310 Module_type((Channel),Type$)
4315 OUTPUT @Hp4142;"MM7,";Channel
4320 OUTPUT @Hp4142;"FLO" ! SMU filter off
4325 !
4330 SELECT Meas_mode
4335 CASE 1
4340 IF Type$="HP41424" THEN
4345 IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
4350 OUTPUT @Hp4142;"RV";Channel;",";Rng
4355 END IF
4360 CASE 2
4365 IF Type$="HP41424" THEN
4370 DISP "Dpulse_measure: HP41424 is not specified for current measure"
4375 BEEP
4380 STOP
4385 ELSE
4390 IF Rng<>0 THEN CALL Set_i_range(Rng)
4395 OUTPUT @Hp4142;"RI";Channel;",";Rng
4400 END IF
4405 CASE ELSE
4410 DISP "Dpulse_measure: Illegal measurement mode value"
4415 BEEP
4420 STOP
4425 END SELECT
4430 !
4435 IF NPAR>3 THEN
4440 IF Primary=0 THEN
4445 OUTPUT @Hp4142;"PDM"
4450 ELSE
4455 OUTPUT @Hp4142;"PDM";Primary
4460 END IF
4465 ELSE
4470 OUTPUT @Hp4142;"PDM"
4475 END IF
4480 OUTPUT @Hp4142;"IE"
4485 OUTPUT @Hp4142;"*OPC?"
4490 ENTER @Hp4142;A$
4495 S=SPOLL(@Hp4142)
4500 IF BIT(S,5) THEN CALL Detect_error(S,"Dpulse_measure")
4505 !*****
4510 !* Execute trigger and get measured data *
4515 !*****
4520 IF NPAR<6 THEN
4525 ENTER @Hp4142;Meas_data
4530 OUTPUT @Hp4142;"FL1" ! SMU filter on
4535 SUBEXIT
4540 ELSE
4545 ENTER @Hp4142;Data$
4550 OUTPUT @Hp4142;"FL1" ! SMU filter on
4555 Meas_data=VAL(Data$[4;12])
4560 Status$=Data$[1;1]
4565 IF Status$="N" THEN Status=0
4570 IF Status$="T" THEN Status=1
4575 IF Status$="C" THEN Status=2

```

```
4580     IF Status$="X" THEN Status=3
4585     IF Status$="V" THEN Status=4
4590     IF Status$="F" THEN Status=7
4595     END IF
4600     !
4605 SUBEND
```

Pulse_measure

```

4620 SUB Pulse_measure(INTEGER Channel,Meas_mode,REAL Meas_data,OPTIONAL Range,INTEGER Status)
4625 COM @Hp4142
4630 !*****
4635 !* Performs the pulsed spot measurement -current & voltage - *
4640 !*****
4645 Rng=0
4650 IF NPAR>3 THEN Rng=Range
4655 Module_type((Channel),Type$)
4660 OUTPUT @Hp4142;"MM3,";Channel
4665 OUTPUT @Hp4142;"FLO" ! SMU filter off
4670 !
4675 SELECT Meas_mode
4680 CASE 1
4685 IF Type$="HP41424" THEN
4690 IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
4695 OUTPUT @Hp4142;"RV";Channel;"",Rng
4700 END IF
4705 CASE 2
4710 IF Type$="HP41424" THEN
4715 DISP "Pulse_measure: HP41424 is not specified for current measure"
4720 BEEP
4725 STOP
4730 ELSE
4735 IF Rng<>0 THEN CALL Set_i_range(Rng)
4740 OUTPUT @Hp4142;"RI";Channel;"",Rng
4745 END IF
4750 CASE ELSE
4755 DISP "Pulse_measure: Illegal measurement mode value"
4760 BEEP
4765 STOP
4770 END SELECT
4775 !
4780 OUTPUT @Hp4142;"IE"
4785 OUTPUT @Hp4142;"*OPC?"
4790 ENTER @Hp4142;A$
4795 S=SPOLL(@Hp4142)
4800 IF BIT(S,5) THEN CALL Detect_error(S,"Pulse_measure")
4805 !*****
4810 !* Execute trigger and get measured data *
4815 !*****
4820 IF NPAR<5 THEN
4825 ENTER @Hp4142;Meas_data
4830 OUTPUT @Hp4142;"FL1" ! SMU filter on
4835 SUBEXIT
4840 ELSE
4845 ENTER @Hp4142;Data$
4850 OUTPUT @Hp4142;"FL1" ! SMU filter on
4855 Meas_data=VAL(Data$[4;12])
4860 Status$=Data$[1;1]
4865 IF Status$="I" THEN Status=0
4870 IF Status$="T" THEN Status=1
4875 IF Status$="C" THEN Status=2
4880 IF Status$="X" THEN Status=3
4885 IF Status$="V" THEN Status=4
4890 IF Status$="F" THEN Status=7
4895 END IF
4900 !
4905 SUBEND

```

Set_piv

```
4920 SUB Set_piv(INTEGER Channel,Swp_mode,REAL Range,Pbase,Start,Stop,No_step,Pwidth,Period,Htime,
OPTIONAL Compliance)
4925   COM @Hp4142
4930   !*****
4935   !*           Setup the pluse sweep parameters           *
4940   !*****
4945   Rng=Range
4950   IF Swp_mode<1 OR Swp_mode>4 THEN
4955     DISP "Set_piv: Illegal sweep mode value"
4960     BEEP
4965     STOP
4970   END IF
4975   !
4980   IF Swp_mode=1 OR Swp_mode=3 THEN
4985     IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
4990     SELECT NPAR
4995     CASE 10
5000       OUTPUT @Hp4142;"PWV";Channel;"",Swp_mode;"",Rng;"",Pbase;"",Start;"",Stop;"",No_step;
";PT";Htime;"",Pwidth;"",Period
5005     CASE 11
5010       OUTPUT @Hp4142;"PWV";Channel;"",Swp_mode;"",Rng;"",Pbase;"",Start;"",Stop;"",No_step;
";",Compliance;"",PT";Htime;"",Pwidth;"",Period
5015     END SELECT
5020   END IF
5025   !
5030   IF Swp_mode=2 OR Swp_mode=4 THEN
5035     Module_type((Channel),Type$)
5040     IF Type$="HP41424" THEN
5045       DISP "Set_piv: HP41424 is not specified for current source"
5050       BEEP
5055       STOP
5060     ELSE
5065       Swp_mode=Swp_mode-1
5070       IF Rng<>0 THEN CALL Set_i_range(Rng)
5075       SELECT NPAR
5080       CASE 10
5085         OUTPUT @Hp4142;"PWI";Channel;"",Swp_mode;"",Rng;"",Pbase;"",Start;"",Stop;"",No_step;
";PT";Htime;"",Pwidth;"",Period
5090       CASE 11
5095         OUTPUT @Hp4142;"PWI";Channel;"",Swp_mode;"",Rng;"",Pbase;"",Start;"",Stop;"",No_step;
";",Compliance;"",PT";Htime;"",Pwidth;"",Period
5100       END SELECT
5105     END IF
5110   END IF
5115   !
5120   OUTPUT @Hp4142;"*OPC?"
5125   ENTER @Hp4142;A$
5130   S=SPOLL(@Hp4142)
5135   IF BIT(S,5) THEN CALL Detect_error(S,"Set_piv")
5140 SUBEND
```

Dsweep_piv

```

5155 SUB Dsweep_piv(INTEGER Channel,Mmode,REAL Range,Meas_data(*),OPTIONAL INTEGER Primary,
REAL Sweep_value(*))
5160   OPTION BASE 1
5165   COM @Hp4142
5170   !*****
5175   !*           Performs the 2CH pulsed sweep measurement           *
5180   !*****
5185   Data_num=SIZE(Meas_data,1)           ! check Meas_data(*) size
5190   OUTPUT @Hp4142;"MM8,";Channel
5195   OUTPUT @Hp4142;"WU?"           ! check number of sweep step
5200   ENTER @Hp4142;No_step
5205   IF No_step<=Data_num THEN Data_num=No_step
5210   !
5215   IF NPAR<6 THEN OUTPUT @Hp4142;"FMT1,0"           ! get only measured data
5220   IF NPAR=6 THEN OUTPUT @Hp4142;"FMT1,1"           ! get measured and sweep data
5225   IF NPAR<6 THEN ALLOCATE Data$(No_step*16)
5230   IF NPAR=6 THEN ALLOCATE Data$[(No_step*16)*2]
5235   !*****
5240   !*           Setup the measurement parameters           *
5245   !*****
5250   Rng=Range
5255   Module_type((Channel),Type$)
5260   OUTPUT @Hp4142;"FLO"           ! SMU filter off
5265   SELECT Mmode
5270   CASE 1
5275     IF Type$="HP41424" THEN
5280       IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
5285       OUTPUT @Hp4142;"RV";Channel;",";Rng
5290     END IF
5295   CASE 2
5300     IF Type$="HP41424" THEN
5305       DISP "Dsweep_piv: HP41424 is not specified for current measure"
5310       BEEP
5315       STOP
5320     ELSE
5325       IF Rng<>0 THEN CALL Set_i_range(Rng)
5330       OUTPUT @Hp4142;"RI";Channel;",";Rng
5335     END IF
5340   CASE ELSE
5345     DISP "Dsweep_piv: Illegal measurement mode value"
5350     BEEP
5355     STOP
5360   END SELECT
5365   IF NPAR>4 THEN
5370     IF Primary=0 THEN
5375       OUTPUT @Hp4142;"PDM"
5380     ELSE
5385       OUTPUT @Hp4142;"PDM";Primary
5390     END IF
5395   ELSE
5400     OUTPUT @Hp4142;"PDM"
5405   END IF
5410   OUTPUT @Hp4142;"*OPC?"
5415   ENTER @Hp4142;A$
5420   S=SPOLL(@Hp4142)
5425   IF BIT(S,5) THEN CALL Detect_error(S,"Dsweep_piv")
5430   !*****
5435   !*           Start sweep and get the measured value           *
5440   !*****
5445   OUTPUT @Hp4142;"IE"
5450   OUTPUT @Hp4142;"*OPC?"
5455   ENTER @Hp4142;A$

```

```

5460 S=SPOLL(@Hp4142)
5465 IF BIT(S,5) THEN CALL Detect_error(S,"Dsweep_piv")
5470 ENTER @Hp4142;Data$
5475 OUTPUT @Hp4142;"FL1" ! SMU filter on
5480 !*****
5485 !* Return only measured value *
5490 !*****
5495 IF NPAR<6 THEN
5500 Pointer=1
5505 FOR I=1 TO Data_num*16 STEP 16
5510 Meas$=Data$[I;15]
5515 IF Meas$[1;1]<>"N" THEN GOSUB Error_disp
5520 Meas_data(Pointer)=VAL(Meas$[4;12])
5525 Pointer=Pointer+1
5530 NEXT I
5535 SUBEXIT
5540 END IF
5545 !*****
5550 !* Return both measured and source value *
5555 !*****
5560 IF NPAR=6 THEN
5565 Pointer=1
5570 FOR I=1 TO (Data_num*16)*2 STEP 32
5575 Meas$=Data$[I;15]
5580 Sweep$=Data$[I+16;15]
5585 IF Meas$[1;1]<>"N" THEN GOSUB Error_disp
5590 Meas_data(Pointer)=VAL(Meas$[4;12])
5595 Sweep_value(Pointer)=VAL(Sweep$[4;12])
5600 Pointer=Pointer+1
5605 NEXT I
5610 END IF
5615 SUBEXIT
5620 !
5625 Error_disp: !
5630 DISP "Dsweep_piv: ";
5635 SELECT Meas$[1;1]
5640 CASE "T"
5645 DISP CHR$(129);" Another channel has reached compliance ";CHR$(128)
5650 CASE "C"
5655 DISP CHR$(129);" This channel has reached compliance ";CHR$(128)
5660 CASE "X"
5665 DISP CHR$(129);" This channel is oscillating ";CHR$(128)
5670 CASE "V"
5675 DISP CHR$(129);" Measurement data overflow ";CHR$(128)
5680 END SELECT
5685 RETURN
5690 !
5695 SUBEND

```

Sweep_piv

```

5710 SUB Sweep_piv(INTEGER Channel,Mmode,REAL Range,Meas_data(*),OPTIONAL Sweep_value(*))
5715   OPTION BASE 1
5720   COM @Hp4142
5725   !*****
5730   !*           Performs the pulsed sweep measurement           *
5735   !*****
5740   Data_num=SIZE(Meas_data,1)           ! check Meas_data(*) size
5745   OUTPUT @Hp4142;"MM4,";Channel
5750   OUTPUT @Hp4142;"WNU?"           ! check number of sweep step
5755   ENTER @Hp4142;No_step
5760   IF No_step<=Data_num THEN Data_num=No_step
5765   !
5770   IF NPAR=4 THEN OUTPUT @Hp4142;"FMT1,0"           ! get only measured data
5775   IF NPAR=5 THEN OUTPUT @Hp4142;"FMT1,1"           ! get measured and sweep data
5780   IF NPAR=4 THEN ALLOCATE Data$(No_step*16)
5785   IF NPAR=5 THEN ALLOCATE Data$[(No_step*16)*2]
5790   !*****
5795   !*           Setup the measurement parameters           *
5800   !*****
5805   Rng=Range
5810   Module_type((Channel),Type$)
5815   OUTPUT @Hp4142;"FLO"           ! SMU filter off
5820   SELECT Mmode
5825   CASE 1
5830     IF Type$="HP41424" THEN
5835       IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
5840       OUTPUT @Hp4142;"RV";Channel;",";Rng
5845     END IF
5850   CASE 2
5855     IF Type$="HP41424" THEN
5860       DISP "Sweep_piv: HP41424 is not specified for current measure"
5865       BEEP
5870       STOP
5875     ELSE
5880       IF Rng<>0 THEN CALL Set_i_range(Rng)
5885       OUTPUT @Hp4142;"RI";Channel;",";Rng
5890     END IF
5895   CASE ELSE
5900     DISP "Sweep_piv: Illegal measurement mode value"
5905     BEEP
5910     STOP
5915   END SELECT
5920   !*****
5925   !*           Start sweep and get the measured value           *
5930   !*****
5935   OUTPUT @Hp4142;"XE"
5940   OUTPUT @Hp4142;"*OPC?"
5945   ENTER @Hp4142;A$
5950   S=SPOLL(@Hp4142)
5955   IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_piv")
5960   ENTER @Hp4142;Data$
5965   OUTPUT @Hp4142;"FL1"           ! SMU filter on
5970   !*****
5975   !*           Return only measured value           *
5980   !*****
5985   IF NPAR=4 THEN
5990     Pointer=1
5995     FOR I=1 TO Data_num*16 STEP 16
6000       Meas$=Data$(I;15)
6005       IF Meas$(1;1)<>"N" THEN GOSUB Error_disp
6010       Meas_data(Pointer)=VAL(Meas$(4;12))
6015       Pointer=Pointer+1

```

```

6020     NEXT I
6025     SUBEXIT
6030 END IF
6035 !*****
6040 !*         Return both measured and source value         *
6045 !*****
6050 IF NPAR=5 THEN
6055     Pointer=1
6060     FOR I=1 TO (Data_num*16)*2 STEP 32
6065         Meas$=Data$[I;15]
6070         Sweep$=Data$[I+16;15]
6075         IF Meas$[1;1]<>"N" THEN GOSUB Error_disp
6080         Meas_data(Pointer)=VAL(Meas$[4;12])
6085         Sweep_value(Pointer)=VAL(Sweep$[4;12])
6090         Pointer=Pointer+1
6095     NEXT I
6100 END IF
6105 SUBEXIT
6110 !
6115 Error_disp: !
6120     DISP "Sweep_piv: ";
6125     SELECT Meas$[1;1]
6130     CASE "T"
6135         DISP CHR$(129);" Another channel has reached compliance ";CHR$(128)
6140     CASE "C"
6145         DISP CHR$(129);" This channel has reached compliance ";CHR$(128)
6150     CASE "X"
6155         DISP CHR$(129);" This channel is oscillating ";CHR$(128)
6160     CASE "V"
6165         DISP CHR$(129);" Measurement data overflow ";CHR$(128)
6170     CASE "F"
6175         DISP CHR$(129);" HVU is not settled ";CHR$(128)
6180     END SELECT
6185     RETURN
6190     !
6195 SUBEND

```

Sweep_pbias

```

6210 SUB Sweep_pbias(INTEGER Channel,Mmode,REAL Range,Meas_data(*),OPTIONAL Sweep_value(*))
6215 OPTION BASE 1
6220 COM @Hp4142
6225 !*****
6230 !*          Performs the pulsed sweep measurement          *
6235 !*****
6240 Data_num=SIZE(Meas_data,1)          ! check Meas_data(*) size
6245 OUTPUT @Hp4142;"MM5,";Channel
6250 OUTPUT @Hp4142;"WNU?"          ! check number of sweep step
6255 ENTER @Hp4142;No_step
6260 IF No_step<=Data_num THEN Data_num=No_step
6265 !
6270 IF NPAR=4 THEN OUTPUT @Hp4142;"FMT1,0"          ! get only measured data
6275 IF NPAR=5 THEN OUTPUT @Hp4142;"FMT1,1"          ! get measured and sweep data
6280 IF NPAR=4 THEN ALLOCATE Data$(No_step*16)
6285 IF NPAR=5 THEN ALLOCATE Data$[(No_step*16)*2]
6290 !*****
6295 !*          Setup the measurement parameters          *
6300 !*****
6305 Rng=Range
6310 Module_type((Channel),Type$)
6315 OUTPUT @Hp4142;"FLO"          ! SMU filter off
6320 SELECT Mmode
6325 CASE 1
6330 IF Type$="HP41424" THEN
6335 IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
6340 OUTPUT @Hp4142;"RV";Channel;"",Rng
6345 END IF
6350 CASE 2
6355 IF Type$="HP41424" THEN
6360 DISP "Sweep_pbias: HP41424 is not specified for current measure"
6365 BEEP
6370 STOP
6375 ELSE
6380 IF Rng<>0 THEN CALL Set_i_range(Rng)
6385 OUTPUT @Hp4142;"RI";Channel;"",Rng
6390 END IF
6395 CASE ELSE
6400 DISP "Sweep_pbias: Illegal measurement value"
6405 BEEP
6410 STOP
6415 END SELECT
6420 !*****
6425 !*          Start sweep and get the measured value          *
6430 !*****
6435 OUTPUT @Hp4142;"XE"
6440 OUTPUT @Hp4142;"*OPC?"
6445 ENTER @Hp4142;A$
6450 S=SPOLL(@Hp4142)
6455 IF BIT(S,5) THEN CALL Detect_error(S,"Sweep_pbias")
6460 ENTER @Hp4142;Data$
6465 OUTPUT @Hp4142;"FL1"          ! SMU filter on
6470 !*****
6475 !*          Return only measured value          *
6480 !*****
6485 IF NPAR=4 THEN
6490 Pointer=1
6495 FOR I=1 TO Data_num*16 STEP 16
6500 Meas$=Data$(I;15)
6505 IF Meas$(1;1)<>"N" THEN GOSUB Error_disp
6510 Meas_data(Pointer)=VAL(Meas$(4;12))
6515 Pointer=Pointer+1

```

```

6520     NEXT I
6525     SUBEXIT
6530 END IF
6535 !*****
6540 !*           Return both measured and source value           *
6545 !*****
6550 IF NPAR=5 THEN
6555     Pointer=1
6560     FOR I=1 TO (Data_num*16)*2 STEP 32
6565         Meas$=Data$[I;15]
6570         Sweep$=Data$[I+16;15]
6575         IF Meas$[1;1]<>"N" THEN GOSUB Error_disp
6580         Meas_data(Pointer)=VAL(Meas$[4;12])
6585         Sweep_value(Pointer)=VAL(Sweep$[4;12])
6590         Pointer=Pointer+1
6595     NEXT I
6600 END IF
6605 SUBEXIT
6610 !
6615 Error_disp: !
6620     DISP "Sweep_pbias: ";
6625     SELECT Meas$[1;1]
6630     CASE "T"
6635         DISP CHR$(129);" Another channel has reached compliance ";CHR$(128)
6640     CASE "C"
6645         DISP CHR$(129);" This channel has reached compliance ";CHR$(128)
6650     CASE "X"
6655         DISP CHR$(129);" This channel is oscillating ";CHR$(128)
6660     CASE "V"
6665         DISP CHR$(129);" Measurement data overflow ";CHR$(128)
6670     CASE "F"
6675         DISP CHR$(129);" HVU is not settled ";CHR$(128)
6680     END SELECT
6685     RETURN
6690     !
6695 SUBEND

```

Set_asource

```
6710 SUB Set_asource(INTEGER Channel,REAL Rstart,Rstop,Rspeed,Htime,Dtime,OPTIONAL Compliance)
6715   COM @Hp4142
6720   !*****
6725   !*   Setup the analog feedback measurement parameters   *
6730   !*****
6735   SELECT NPAR
6740   CASE 6
6745     OUTPUT @Hp4142;"ASV";Channel;"",Rstart;"",Rstop;"",Rspeed;"",AT";Htime;"",Dtime
6750   CASE 7
6755     OUTPUT @Hp4142;"ASV";Channel;"",Rstart;"",Rstop;"",Rspeed;"",Compliance;"",AT";Htime;"",Dtime
6760   END SELECT
6765   !
6770   OUTPUT @Hp4142;"*OPC?"
6775   ENTER @Hp4142;A$
6780   S=SPOLL(@Hp4142)
6785   IF BIT(S,5) THEN CALL Detect_error(S,"Set_asource")
6790 SUBEND
```

Set_amonitor

```
6805 SUB Set_amonitor(INTEGER Channel,Mmode,REAL Source,Goal,OPTIONAL Compliance)
6810   COM @Hp4142
6815   !*****
6820   !*   Setup the analog feedback measurement parameters   *
6825   !*****
6830   SELECT Mmode
6835   CASE 1
6840     IF NPAR=4 THEN
6845       OUTPUT @Hp4142;"AVI";Channel;"",Source;"",Goal
6850     ELSE
6855       OUTPUT @Hp4142;"AVI";Channel;"",Source;"",Goal;"",Compliance
6860     END IF
6865   CASE 2
6870     IF NPAR=4 THEN
6875       OUTPUT @Hp4142;"AIV";Channel;"",Source;"",Goal
6880     ELSE
6885       OUTPUT @Hp4142;"AIV";Channel;"",Source;"",Goal;"",Compliance
6890     END IF
6895   CASE ELSE
6900     DISP "Set_amonitor: Illegal measurement mode value"
6905     BEEP
6910     STOP
6915   END SELECT
6920   !
6925   OUTPUT @Hp4142;"*OPC?"
6930   ENTER @Hp4142;A$
6935   S=SPOLL(@Hp4142)
6940   IF BIT(S,5) THEN CALL Detect_error(S,"Set_amonitor")
6945 SUBEND
```

Measure_asearch

```

6960 SUB Measure_asearch(INTEGER Smode,Mmode,REAL Integtime,Sdata,OPTIONAL Mdata,INTEGER Status)
6965   COM @Hp4142
6970   INTEGER Sst,Mst
6975   !*****
6980   !*       Performs the analog feedback measurement       *
6985   !*****
6990   ALLOCATE Data$(32)
6995   OUTPUT @Hp4142;"FMT1;MM6;ASM";Smode;" ";Mmode;" ";Integtime
7000   OUTPUT @Hp4142;"IE"
7005   OUTPUT @Hp4142;"*OPC?"
7010   ENTER @Hp4142;A$
7015   S=SPOLL(@Hp4142)
7020   IF BIT(S,5) THEN CALL Detect_error(S,"Measure_asearch")
7025   ENTER @Hp4142;Data$
7030   !*****
7035   !*       Return search data or sense data       *
7040   !*****
7045   IF Mmode<3 THEN
7050     IF NPAR>4 THEN
7055       DISP "Measure_asearch: This meas mode returns only ";CHR$(129);"Source";CHR$(128);" value."
7060       BEEP
7065       STOP
7070     END IF
7075     Sdata=VAL(Data$(4;12))           ! Setup source data
7080   ELSE
7085     IF NPAR<5 THEN
7090       BEEP
7095       DISP "Measure_asearch: This meas mode must be specified ";CHR$(129);"Source";CHR$(128);" and "
;CHR$(129);"Goal";CHR$(128);" value."
7100       STOP
7105     END IF
7110     Sdata=VAL(Data$(4;12))           ! Setup source data
7115     Mdata=VAL(Data$(20;12))         ! Setup monitor data
7120   END IF
7125   !*****
7130   !*       Error and status data handling       *
7135   !*****
7140   IF NPAR<6 THEN
7145     IF Data$(1;1)<>" " THEN GOSUB Error_disp
7150   ELSE
7155     Sst$=Data$(1;1)
7160     Mst$=Data$(17;1)
7165     IF Sst$=" " THEN Sst=0
7170     IF Sst$="T" THEN Sst=10
7175     IF Sst$="C" THEN Sst=20
7180     IF Sst$="I" THEN Sst=30
7185     IF Sst$="V" THEN Sst=40
7190     IF Sst$="G" THEN Sst=50
7195     IF Sst$="S" THEN Sst=60
7200     IF Sst$="F" THEN Sst=70
7205     !
7210     IF Mst$=" " THEN Mst=0
7215     IF Mst$="T" THEN Mst=1
7220     IF Mst$="C" THEN Mst=2
7225     IF Mst$="I" THEN Mst=3
7230     IF Mst$="V" THEN Mst=4
7235     IF Mst$="F" THEN Mst=7
7240     Status=Sst+Mst
7245   END IF
7250   !
7255   SUBEXIT
7260   !

```

```

7265 Error_disp:  !
7270     DISP "Measure_asearch: ";
7275     SELECT Data$[1;1]
7280     CASE "T"
7285         DISP CHR$(129);" Monitor channel has reached compliance ";CHR$(128)
7290     CASE "C"
7295         DISP CHR$(129);" Source channel has reached compliance ";CHR$(128)
7300     CASE "I"
7305         DISP CHR$(129);" Source channel is Oscillating ";CHR$(128)
7310     CASE "G"
7315         DISP CHR$(129);" No search goal found ";CHR$(128)
7320     CASE "S"
7325         DISP CHR$(129);" Unsettled feedback operation ";CHR$(128)
7330     CASE "V"
7335         DISP CHR$(129);" Source data overflow ";CHR$(128)
7340     CASE "F"
7345         DISP CHR$(129);" HVU does not settle before the measurement";CHR$(128)
7350     END SELECT
7355     BEEP
7360     RETURN
7365     !
7370 SUBEND

```

Para_vth

```

7385 SUB Para_vth(Vd,Id,Id_max,Rstart,Rstop,Ig_max,Cin,Cgd,Integ_time,Rspeed,Dtime,OPTIONAL INTEGER Err,
REAL Hold_time,Gm,Cd)
7390 !*****
7395 !*          Parameter Extraction of Vth measurement          *
7400 !*****
7405 !*          Initialize routine          *
7410 !*****
7415 Id_a=ABS(Id)
7420 Id_max_a=ABS(Id_max)
7425 GOSUB Err_check
7430 CALL V_range(Rstart,Rstop,Vrange,Error)
7435 IF Error THEN GOSUB Err_disp
7440 !
7445 ! SENSE I-RANGE SELECTION
7450 Ir_d=1+INT(LGT(Id_max_a/1.150005))
7455 IF Ir_d<-9.5 THEN Ir_d=-9
7460 Rr_d=1/10^Ir_d
7465 !
7470 ! SEARCH I-RANGE SELECTION
7475 Ir_g=1+INT(LGT(ABS(Ig_max)/1.150005))
7480 IF Ir_g<-9.5 THEN Ir_g=-9
7485 Rr_g=1/10^Ir_g
7490 IF NPAR<13.5 THEN
7495   Gm=20*SQR(Id_a*Id_max_a)
7500 ELSE
7505   Gm=Gm*SQR(Id_a*Id_max_a)/Id_a
7510 END IF
7515 IF NPAR<14.5 THEN
7520   Cd0=3.E-11
7525 ELSE
7530   Cd0=Cd
7535   IF Cd<0 THEN Error=22
7540   IF Cd>2.E-8 THEN Error=22
7545   IF Error THEN GOSUB Err_disp
7550 END IF
7555 !*****
7560 !*          Calculate Integ_time          *
7565 !*****
7570 CALL Tau_read(Ir_g,Ir_d,Tau30,Tau31,Tau50,Tau51,Tau_m,V_slew_m,Cod_m,Tau_s,V_slew_s,Cod_s)
7575 ! Tau3 : V-loop F-response : affected with load capacitance
7580 Tau3=Tau30
7585 IF Cin>1.E-10 THEN Tau3=Tau3+Tau31*(Cin-1.00E-10)/9.00E-10
7590 ! Tau4 : Current through cap. between Sense-SMU and Search-SMU
7595 Tau4=2*Rr_d*Cgd ! 2 IS 6dB GAIN MARGIN
7600 ! Tau5 : F-response of current to AFU error-amp output ; determined
7605 ! with I-range and influenced by load cap. of Sense-SMU
7610 Tau5=Tau50
7615 IF Cd0>1.00E-10 THEN Tau5=Tau5+Tau51*(Cd0-1.E-10)/9.E-10
7620 Tau=Tau2*Tau2+Tau3*Tau3+Tau5*Tau5
7625 Adut=Gm*Rr_d
7630 Integ_time=1.5*SQR(Adut*Adut*Tau+Tau4*Tau4) ! 1.5 IS MARGIN
7635 It_min=1.0E-4/Vrange ! min Integ_time
7640 IF Integ_time<It_min THEN Integ_time=It_min
7645 !*****
7650 !*          Calculate Ramp_speed          *
7655 !*****
7660 IF Vrange>25 THEN ! max Ramp_speed
7665   Sr1=1.E+5
7670 ELSE
7675   Sr1=2.5E+3*Vrange
7680 END IF
7685 Dt=Tau3+Tau5+2.0E-5

```

```

7690 Adut2=Gm_/Id_a
7695 Sr2=ABS(Id_max-Id)/(Cgd+Adut2*Dt/Rr_d)*.8 ! .8 IS MARGIN
7700 Dt_all=Dt+Cgd/Gm_
7705 IF Ir_d<=-5 THEN ! For low current range
7710 Sr_fb0=.1/Integ_time
7715 Sr_n=Sr2*2
7720 IF Cgd=0 THEN Cgd=1.E-15
7725 LOOP
7730 Sr_fb=MIN(Sr_fb0,Sr_n*Gm_*Dt_all*Rr_d/Integ_time)
7735 I_0=Id_a+Sr_n*Gm_*(Tau5+2.0E-5-Tau_m)
7740 IF I_0<0 THEN I_0=0
7745 D_i=Id_a+Sr_n*(Gm_*Dt+Cgd)-I_0
7750 I_fb=Sr_fb*Cgd
7755 I_d_fb=Sr_n*Gm_*Dt+Cgd*(Sr_n+Sr_fb)
7760 T_0=(Dt*Sr_n*Gm_+Cgd*(Sr_n+Sr_fb))/Sr_fb/Gm_+Tau3
7765 ! Im_last=I_0+(D_i+I_fb)-I_d_fb
7770 IF Tau_m<>T_0 THEN
7775 A=(D_i+I_fb)/Tau_m-I_d_fb/(Tau_m-T_0)
7780 B=I_d_fb/(T_0-Tau_m)
7785 IF A/B>0 THEN
7790 T_max=LOG(A/B)/(1/Tau_m-1/T_0)
7795 IF T_max<0 THEN T_max=0
7800 M1=EXP(-T_max/Tau_m)
7805 M2=EXP(-T_max/T_0)
7810 I_max_2=I_0+(D_i+I_fb)*(1-M1)-I_d_fb*(1-T_0/(T_0-Tau_m)*M2-Tau_m/(Tau_m-T_0)*M1)
7815 ELSE
7820 I_max_2=MAX(I_0,Id_a)
7825 END IF
7830 ELSE
7835 I_max_2=MAX(I_0,Id_a)
7840 END IF
7845 EXIT IF I_max_2>Id_max_a
7850 Sr_n=Sr_n*2
7855 I_mm=I_max_2
7860 END LOOP
7865 Sr20=Sr_n/2*(1+(Id_max_a-I_mm)/(I_max_2-I_mm))* .8
7870 Sr21=MAX(Sr2,Sr20)
7875 Sr22=.8*ABS(Id_max-Id)/Cgd ! .8 IS MARGIN
7880 Sr2=MIN(Sr21,Sr22)
7885 END IF
7890 Sr3=.8*ABS(Ig_max)/(Cin+Cod_s+1/Rr_g/580000) ! .8 IS MARGIN
7895 Sr_rec=SQR(ABS(Rstart-Rstop)/8/Integ_time/Dt_all)
7900 Rspeed=MIN(Sr1,Sr2,Sr3,Sr_rec)
7905 !*****
7910 !* Calculate Delay_time *
7915 !*****
7920 Conv_time=1.2*(Rspeed*D1/8*Integ_time+5*Integ_time/Adut)
7925 Dt_defaults=MAX(1.00E-4,Integ_time)
7930 IF Conv_time>Dt_defaults+1.E-4 THEN
7935 Dtime=(INT((Conv_time-Dt_defaults-1.E-4)*1.E+3)+1)*1.E-3
7940 ELSE
7945 Dtime=0
7950 END IF
7955 ! Hold_t1 : Vd setting time
7960 CALL Cal_hold_time(Id_max,Ir_d,Vd,Cd0,Cod_m,Tau_m,V_slew_m,Hold_t1,Wait_d1)
7965 ! Hold_t2 : Vg ( start voltage ) setting time
7970 CALL Cal_hold_time(Ig_max,Ir_g,Rstart,Cin,Cod_s,Tau_s,V_slew_s,Hold_t2,Wait_d2)
7975 ! Ig_comp_time : Period of current through Cgd drive Search SMU
7980 Ig_comp_time=ABS(Vd)*Cgd/Ig_max
7985 IF Ig_comp_time>Wait_d1 THEN
7990 Hold_t3=Ig_comp_time-Wait_d1
7995 ELSE
8000 Hold_t3=0
8005 END IF
8010 IF Hold_t2>0 THEN
8015 T_hold=Hold_t2+Ig_comp_time-Wait_d1

```

```

8020 END IF
8025 IF NPAR>12.5 THEN
8030 Hold_time=1.E-3*(INT((MAX(Hold_t1,Hold_t3,T_hold)*1.E+3)+.95))
8035 END IF
8040 SUBEXIT
8045 !
8050 Err_check: !
8055 !*****
8060 !* Parameter error check *
8065 !*****
8070 Error=0
8075 IF NPAR>11.5 THEN Err=0
8080 IF ABS(Vd)>200 THEN Error=10
8085 IF Error THEN GOSUB Err_disp
8090 IF Id_a<1.E-11 THEN Error=11
8095 IF Id_a>.9 THEN Error=11
8100 IF Error THEN GOSUB Err_disp
8105 IF SGN(Vd)<>SGN(Id) THEN Error=12
8110 IF Error THEN GOSUB Err_disp
8115 IF Id_max_a<1.05*Id_a THEN Error=13
8120 IF Id_max_a>1 THEN Error=13
8125 IF Error THEN GOSUB Err_disp
8130 IF SGN(Id_max)<>SGN(Id) THEN Error=14
8135 IF Error THEN GOSUB Err_disp
8140 IF ABS(Rstart)>200 THEN Error=15
8145 IF Error THEN GOSUB Err_disp
8150 IF ABS(Rstop)>200 THEN Error=16
8155 IF Error THEN GOSUB Err_disp
8160 IF (Rstop-Rstart)*Vd<0 THEN Error=17
8165 IF Error THEN GOSUB Err_disp
8170 IF ABS(Ig_max)>1 THEN Error=19
8175 IF ABS(Ig_max)<1.E-12 THEN Error=19
8180 IF Error THEN GOSUB Err_disp
8185 IF Cin<0 THEN Error=20
8190 IF Cin>2.E-8 THEN Error=20
8195 IF Error THEN GOSUB Err_disp
8200 IF Cgd<0 THEN Error=21
8205 IF Cgd>2.E-8 THEN Error=21
8210 IF Error THEN GOSUB Err_disp
8215 RETURN
8220 !
8225 Err_disp: !
8230 IF NPAR<12 THEN
8235 IF Error=10 THEN DISP "Para_vth: |Vd| > 200 is not allowed."
8240 IF Error=11 THEN DISP "Para_vth: Unsuitable Id. (too small or too large)"
8245 IF Error=12 THEN DISP "Para_vth: Vd and Id must be same polarity."
8250 IF Error=13 THEN DISP "Para_vth: |Id_max| > 1 or |Id_max| < 1.05 * |Id| is not allowed."
8255 IF Error=14 THEN DISP "Para_vth: Id_max and Id must be same polarity."
8260 IF Error=15 THEN DISP "Para_vth: |Start| > 200 is not allowed."
8265 IF Error=16 THEN DISP "Para_vth: |Stop| > 200 is not allowed."
8270 IF Error=17 THEN DISP "Para_vth: (Stop - Start) and Vd must be same polarity."
8275 IF Error=19 THEN DISP "Para_vth: Unsuitable Ig_max. (too small or too large)"
8280 IF Error=20 THEN DISP "Para_vth: Cin must be 0 to 20nF."
8285 IF Error=21 THEN DISP "Para_vth: Cgd must be 0 to 20nF."
8290 ! IF Error=22 THEN DISP "Para_vth: Cd must be 0 to 20nF."
8295 IF Error=30 THEN DISP "Para_vth: |Stop - Start| is too small."
8300 IF Error=31 THEN DISP "Para_vth: |Stop - Start| > 200 is not allowed."
8305 !
8310 BEEP
8315 STOP
8320 ELSE
8325 Err=Error
8330 SUBEXIT
8335 END IF
8340 RETURN
8345 !

```

8350 SUBEND

Para_hfe

```

8365 SUB Para_hfe(Vc,Ic,Ic_max,Rstart,Rstop,Hfe_min,Hfe_max,Ft,Cin,Cbc,Ib,Ib_max,Integ_time,Rspeed,Dtime,
OPTIONAL INTEGER Err,REAL Hold_time,Cc)
8370 !*****
8375 !*          Parameter Extraction of Hfe measurement          *
8380 !*****
8385 Ic_a=ABS(Ic)
8390 Ic_max_a=ABS(Ic_max)
8395 GOSUB Err_check
8400 CALL V_range(Rstart,Rstop,Vrange,Error)
8405 IF Error THEN GOSUB Err_disp
8410 IF WPAR<17.5 THEN
8415   Cc0=3.E-11
8420 ELSE
8425   Cc0=Cc
8430   IF Cc<0 THEN Error=24
8435   IF Cc>2.0E-8 THEN Error=24
8440   IF Error THEN GOSUB Err_disp
8445 END IF
8450 !
8455 Ir_c=1+INT(LGT(ABS(Ic_max)/1.150005))
8460 IF Ir_c<-9.5 THEN Ir_c=-9
8465 Rr_c=1/10Ir_c
8470 !*****
8475 !*          Calculate Ib and Ib_max          *
8480 !*****
8485 Ib=Ic/Hfe_min
8490 Ir_b=1+INT(LGT(ABS(Ib)/1.000005))
8495 IF Ir_b<-9.5 THEN Ir_b=-9
8500 Rr_b=1/10Ir_b
8505 Ib_max=SGN(Ib)*1.15/Rr_b
8510 IF Ib_max>1 THEN Ib_max=1
8515 CALL Tau_read(Ir_b,Ir_c,Tau30,Tau31,Tau50,Tau51,Tau_m,V_slew_m,Cod_m,Tau_s,V_slew_s,Cod_s)
8520 !*****
8525 !*          Calculate Integ_time          *
8530 !*****
8535 Tau1=Hfe_max/(2*PI*Ft)
8540 Tau2=8.E-7*(1+Rr_b*ABS(Ib))/2.5E-2)
8545 IF ABS(Ib)<1.00E-7 THEN Tau2=3.E-6
8550 Tau3=Tau30
8555 C_be=1/.026*ABS(Ic)/2/PI/Ft
8560 IF Cin+C_be>1.E-10 THEN Tau3=Tau3+Tau31*(Cin+C_be-1.E-10)/9.E-10
8565 Tau4=2*Rr_c*Cbc ! 2 IS 6dB GAIN MARGIN
8570 Tau5=Tau50
8575 IF Cc0>1.E-10 THEN Tau5=Tau5+Tau51*(Cc0-1.E-10)/9.E-10
8580 Tau=(Tau1+Tau1+Tau2+Tau2+Tau3+Tau3+Tau5+Tau5)
8585 Adut=40*ABS(Ic*Rr_c)
8590 Gm=Adut/Rr_c*SQR(Ic_a*Ic_max_a)/Ic_a
8595 Integ_time=1.5*SQR(Adut*Adut*Tau+Tau4+Tau4) ! 1.5 IS MARGIN
8600 It_min=1.E-4/Vrange ! min Integ_time @2V range
8605 IF Integ_time<It_min THEN Integ_time=It_min
8610 !*****
8615 !*          Calculate Ramp_speed (=Slew rate)          *
8620 !*****
8625 IF Vrange>25 THEN
8630   Sr1=1.E+5
8635 ELSE
8640   Sr1=2.5E+3*Vrange
8645 END IF
8650 Dt=Tau3+Tau5+2.0E-5+Tau1+Tau2
8655 Adut2=Gm*Rr_c
8660 Sr2=ABS(Ic_max-Ic)/(Cbc+Adut2*Dt/Rr_c)*.8 ! .8 IS MARGIN
8665 Sr_fb=.1/Integ_time

```

```

8670 Dt_all=Dt+Cbc/Gm_
8675 IF Ir_c<-4 THEN
8680   Sr_n=Sr2*2
8685   Sr_fb0=Sr_fb
8690   IF Cbc=0 THEN Cbc=1.E-15
8695   LOOP
8700     Sr_fb=MIN(Sr_fb0,Sr_n*(Gm_*Dt_all+Cbc)*Rr_c/Integ_time)
8705     I_0=Ic_a+Sr_n*Gm_*(Tau5+2.0E-5-Tau_m)
8710     IF I_0<0 THEN I_0=0
8715     D_i=Ic_a+Sr_n*(Gm_*Dt+Cbc)-I_0
8720     I_fb=Sr_fb+Cbc
8725     I_c_fb=Sr_n*Gm_*Dt+Cbc*(Sr_n+Sr_fb)
8730     T_0=(Dt*Sr_n*Gm_+Cbc*(Sr_n+Sr_fb))/Sr_fb/Gm_+Tau3+Tau1+Tau2
8735     IF Tau_m<>T_0 THEN
8740       A=(D_i+I_fb)/Tau_m-I_c_fb/(Tau_m-T_0)
8745       B=I_c_fb/(T_0-Tau_m)
8750       IF A/B>0 THEN
8755         T_max=LOG(A/B)/(1/Tau_m-1/T_0)
8760         IF T_max<0 THEN T_max=0
8765         M1=EXP(-T_max/Tau_m)
8770         M2=EXP(-T_max/T_0)
8775         I_max_2=I_0+(D_i+I_fb)*(1-M1)-I_c_fb*(1-T_0/(T_0-Tau_m)*M2-Tau_m/(Tau_m-T_0)*M1)
8780       ELSE
8785         I_max_2=MAX(I_0,Ic_a)
8790       END IF
8795     ELSE
8800       I_max_2=MAX(I_0,Ic_a)
8805     END IF
8810     EXIT IF I_max_2>ABS(Ic_max)
8815     Sr_n=Sr_n*2
8820     I_mm=I_max_2
8825   END LOOP
8830   Sr20=Sr_n/2*(1+(ABS(Ic_max)-I_mm)/(I_max_2-I_mm))*.8
8835   Sr21=MAX(Sr2,Sr20)
8840   Sr22=.8*ABS(Ic_max-Ic)/Cbc
8845   Sr2=MIN(Sr21,Sr22)
8850 END IF
8855 C_be=1/.026*ABS(Ic)/2/PI/Ft
8860 Sr3=.8*ABS(Ib_max-Ib)/(Cin+Cod_s+1/Rr_b/580000+C_be) ! .8 IS MARGIN
8865 Sr_rec=SQR(ABS(Rstart-Rstop)/8/Integ_time/Dt_all)
8870 Rspeed=MIN(Sr1,Sr2,Sr3,Sr_rec)
8875 !*****
8880 !*          Calculate Delay_time Hold_time          *
8885 !*****
8890 Conv_time=(Rspeed*Dt_all/8*Integ_time+5*Integ_time/Adut)*1.2
8895 Max_di=Rspeed*(Dt_all*40*ABS(Ib)+Cod_s+Cin+Cbc+1/Rr_b/580000+C_be)
8900 Ib_min=Ic/Hfe_max
8905 M_tau=MAX(ABS(LOG(ABS(Ib_min/10/Max_di))),3)
8910 Dtt=Conv_time+M_tau*Tau_s
8915 Dt_defaults=MAX(Integ_time,1.00E-4)
8920 IF Dtt>Dt_defaults+1.E-4 THEN
8925   Dtime=(INT((Dtt-Dt_defaults-1.E-4)*1.E+3)+1)*1.E-3
8930 ELSE
8935   Dtime=0
8940 END IF
8945 CALL Cal_hold_time(Ic_max,Ir_c,Vc,Cc0,Cod_m,Tau_m,V_slew_m,H_time,Wait_def)
8950 IF H_time>0 THEN
8955   H_time=1.E-3*INT(H_time*1.E+3)+1.E-3
8960 END IF
8965 Wait_base=ABS(Vc*Cbc/Ib_max)
8970 IF Wait_base>H_time+Wait_def THEN
8975   H_time=(INT((Wait_base-Wait_def)*1.E+3+1))*1.E-3
8980 END IF
8985 IF NPAR>16.5 THEN Hold_time=H_time
8990 SUBEXIT
8995 !

```

```

9000 Err_check:  !
9005      !*****
9010      !*           Parameter error check           *
9015      !*****
9020      Error=0
9025      IF NPAR>15.5 THEN Err=0
9030      IF ABS(Vc)>200 THEN Err=10
9035      IF Error THEN GOSUB Err_disp
9040      IF ABS(Ic)<1.E-11 THEN Err=11
9045      IF ABS(Ic)>.9 THEN Err=11
9050      IF Error THEN GOSUB Err_disp
9055      IF SGN(Vc)<>SGN(Ic) THEN Err=12
9060      IF Error THEN GOSUB Err_disp
9065      IF ABS(Ic_max)<1.05*ABS(Ic) THEN Err=13
9070      IF ABS(Ic_max)>1 THEN Err=13
9075      IF Error THEN GOSUB Err_disp
9080      IF SGN(Ic_max)<>SGN(Ic) THEN Err=14
9085      IF Error THEN GOSUB Err_disp
9090      IF ABS(Rstart)>200 THEN Err=15
9095      IF Error THEN GOSUB Err_disp
9100      IF ABS(Rstop)>200 THEN Err=16
9105      IF Error THEN GOSUB Err_disp
9110      IF (Rstop-Rstart)*Vc<0 THEN Err=17
9115      IF Error THEN GOSUB Err_disp
9120      IF Hfe_min<1 THEN Err=19
9125      IF Hfe_max<1 THEN Err=19
9130      IF Error THEN GOSUB Err_disp
9135      IF Hfe_max<=Hfe_min THEN Err=20
9140      IF Error THEN GOSUB Err_disp
9145      IF Ft<=0 THEN Err=21
9150      IF Error THEN GOSUB Err_disp
9155      IF Cin<0 THEN Err=22
9160      IF Cin>2.0E-8 THEN Err=22
9165      IF Error THEN GOSUB Err_disp
9170      IF Cbc<0 THEN Err=23
9175      IF Cbc>2.0E-8 THEN Err=23
9180      IF Error THEN GOSUB Err_disp
9185      RETURN
9190      !
9195 Err_disp: !
9200      IF NPAR<16 THEN
9205          IF Error=10 THEN DISP "Para_hfe: |Vc| > 200 is not allowed."
9210          IF Error=11 THEN DISP "Para_hfe: Unsuitable Ic. (too small or too large)"
9215          IF Error=12 THEN DISP "Para_hfe: Vc and Ic must be same polarity."
9220          IF Error=13 THEN DISP "Para_hfe: |Ic_max| > 1 or |Ic_max| < 1.05 * |Ic| is not allowed."
9225          IF Error=14 THEN DISP "Para_hfe: Ic_max and Ic must be same polarity."
9230          IF Error=15 THEN DISP "Para_hfe: |Start| > 200 is not allowed."
9235          IF Error=16 THEN DISP "Para_hfe: |Stop| > 200 is not allowed."
9240          IF Error=17 THEN DISP "Para_hfe: (Stop - Start) and Vc must be same polarity."
9245          IF Error=19 THEN DISP "Para_hfe: Hfe_min < 1 or Hfe_max < 1 is not allowed."
9250          IF Error=20 THEN DISP "Para_hfe: Hfe_max <= Hfe_min is not allowed."
9255          IF Error=21 THEN DISP "Para_hfe: Ft <= 0 is not allowed."
9260          IF Error=22 THEN DISP "Para_hfe: Cin must be 0 to 20nF."
9265          IF Error=23 THEN DISP "Para_hfe: Cbc must be 0 to 20nF."
9270          IF Error=30 THEN DISP "Para_hfe: |Stop - Start| is too small."
9275          IF Error=31 THEN DISP "Para_hfe: |Stop - Start| > 200 is not allowed."
9280          !
9285          BEEP
9290          STOP
9295      ELSE
9300          Err=Error
9305          SUBEXIT
9310      END IF
9315      RETURN
9320      !

```

9325 SUBEED

Set_v_range

```
9340 SUB Set_v_range(INTEGER Channel,REAL Range)
9345 !*****
9350 !*           Setup voltage range paramater           *
9355 !*****
9360 Module_type((Channel),Type$) ! check installed module type
9365 !
9370 SELECT Type$
9375 CASE "HP41420"
9380     IF Range<2 THEN Range=-999
9385     IF Range>200 THEN Range=999
9390 CASE "HP41421"
9395     IF Range<2 THEN Range=-999
9400     IF Range>100 THEN Range=999
9405 CASE "HP41424"
9410     IF Range<.2 THEN Range=-999
9415     IF Range>40 THEN Range=999
9420 CASE "HP41422"
9425     IF Range<2 THEN Range=-999
9430     IF Range>20 THEN Range=999
9435 CASE "HP41423"
9440     IF Range<100 THEN Range=-999
9445     IF Range>1000 THEN Range=999
9450 END SELECT
9455 !
9460 IF Range<>999 AND Range<>-999 THEN
9465     SELECT ABS(Range)
9470     CASE .2,2,20,40,100,200,500,1000
9475     IF ABS(Range)=.2 THEN Rng=10
9480     IF ABS(Range)=2 THEN Rng=11
9485     IF ABS(Range)=20 THEN Rng=12
9490     IF ABS(Range)=40 THEN Rng=13
9495     IF ABS(Range)=100 THEN Rng=14
9500     IF ABS(Range)=200 THEN Rng=15
9505     IF ABS(Range)=500 THEN Rng=16
9510     IF ABS(Range)=1000 THEN Rng=17
9515     CASE ELSE
9520     Rng=0
9525     END SELECT
9530     Range=Rng
9535 END IF
9540 SUBEND
```

Set_i_range

```
9555 SUB Set_i_range(REAL Range)
9560 !*****
9565 !*          Setup the current range parameater          *
9570 !*****
9575 IF Range>0 THEN
9580   SELECT Range
9585   CASE 1.E-9 TO 10
9590     Rselect=((20+LGT(Range))-INT((20+LGT(Range))))
9595     IF Rselect=0 THEN Range=20+LGT(Range)
9600     IF Rselect<>0 THEN Range=0
9605   CASE ELSE
9610     IF Range>10 THEN Range=999
9615     IF Range<1.E-9 THEN Range=-999
9620   END SELECT
9625 END IF
9630 !
9635 IF Range<0 THEN
9640   SELECT Range
9645   CASE -10 TO -1.E-9
9650     Rselect=((20-LGT(ABS(Range)))-INT((20-LGT(ABS(Range)))))
9655     IF Rselect=0 THEN Range=-20-LGT(ABS(Range))
9660     IF Rselect<>0 THEN Range=0
9665   CASE ELSE
9670     IF Range>-1.E-9 THEN Range=999
9675     IF Range<-10 THEN Range=-999
9680   END SELECT
9685 END IF
9690 SUBEND
```

Module_type

```
9705 SUB Module_type(INTEGER Chan,Type$)
9710   COM @Hp4142
9715   !*****
9720   !*           Check installed module type           *
9725   !*****
9730   IF Chan>=11 AND Chan<=18 THEN Chan=Chan-10
9735   IF Chan>=21 AND Chan<=28 THEN Chan=Chan-20
9740   ALLOCATE A$(100),Unt$(8)[8]
9745   !
9750   OUTPUT @Hp4142;"UNT?"
9755   ENTER @Hp4142;A$
9760   K=1
9765   FOR I=1 TO 8
9770     IF A$(K,1)="0" THEN
9775       Unt$(I)=A$(K,1)
9780       K=K+4
9785     ELSE
9790       Unt$(I)=A$(K,7)
9795       K=K+11
9800     END IF
9805   NEXT I
9810   !*****
9815   !*           Return module type           *
9820   !*****
9825   Type$=Unt$(Chan)
9830   !
9835 SUBEND
```

Detect_error

```
9850 SUB Detect_error(Spoll_value,Module_name$)
9855   COM @Hp4142
9860   !*****
9865   !*           Emergency error process           *
9870   !*****
9875   IF BIT(Spoll_value,7) OR BIT(Spoll_value,3) THEN
9880     IF BIT(Spoll_value,7) THEN
9885       DISP CHR$(129);" 4142B SHUT DOWN !! ";CHR$(128)
9890     ELSE
9895       DISP CHR$(129);" 4142B NOT INTERLOCKED !! ";CHR$(128)
9900     END IF
9905     BEEP
9910     STOP
9915   END IF
9920   !*****
9925   !*           Normal error process           *
9930   !*****
9935   OUTPUT @Hp4142;"ERR?"
9940   ENTER @Hp4142;E(1)
9945   DISP "4142B ERROR";E(1);" In ";Module_name$
9950   BEEP
9955   STOP
9960 SUBEND
```

Tau_read

```
9975 SUB Tau_read(S_range,M_range,Tau30,Tau31,Tau50,Tau51,Tau_m,V_slew_m,Cod_m,Tau_s,V_slew_s,Cod_s)
9980 SELECT S_range
9985 CASE 0,-1,-2,-3
9990     Tau30=1.2E-6+1/10^S_range*1.E-9
9995     Tau31=2.E-7
10000     Cod_s=3.500E-9
10005     Tau_s=3.E-6
10010     V_slew_s=2.E-5
10015 CASE -4,-5
10020     Tau30=6.1E-6-1.4E-5*(S_range+4)
10025     Tau31=1.E-6-2.E-6*(S_range+4)
10030     Cod_s=3.500E-9
10035     Tau_s=1.E-5
10040     V_slew_s=1.65E-3
10045     IF S_range=-4 THEN V_slew_s=1.5E-4
10050 CASE -6
10055     Tau30=1.2E-5
10060     Tau31=2.0E-5
10065     Cod_s=2.20E-10
10070     Tau_s=5.E-5
10075     V_slew_s=1.11E-3
10080 CASE -7
10085     Tau30=1.3E-5
10090     Tau31=3.2E-5
10095     Cod_s=7.0E-11
10100     Tau_s=2.E-4
10105     V_slew_s=3.68E-3
10110 CASE -8
10115     Tau30=1.8E-5
10120     Tau31=8.2E-5
10125     Cod_s=4.E-12
10130     Tau_s=9.E-4
10135     V_slew_s=.004
10140 CASE -9
10145     Tau30=4.8E-5
10150     Tau31=3.70E-4
10155     Cod_s=2.E-12
10160     Tau_s=3.E-3
10165     V_slew_s=.024
10170 END SELECT
10175 !
10180 SELECT M_range
10185 CASE 0,-1,-2,-3,-4
10190     Tau50=3.E-6
10195     Tau51=1.E-7
10200     Tau_m=3.E-6
10205     V_slew_m=2.0E-5
10210     IF M_range=-4 THEN V_slew_m=1.5E-4
10215     Cod_m=3.500E-9
10220 CASE -5
10225     Tau50=2.0E-5
10230     Tau51=1.E-6
10235     Tau_m=1.E-5
10240     V_slew_m=1.650E-3
10245     Cod_m=3.500E-9
10250 CASE -6
10255     Tau50=1.0E-5
10260     Tau51=8.E-6
10265     Tau_m=5.E-5
10270     V_slew_m=1.11E-3
10275     Cod_m=2.20E-10
10280 CASE -7
```

```
10285      Tau50=1.5E-5
10290      Tau51=2.0E-5
10295      Tau_m=2.E-4
10300      V_slew_m=3.68E-3
10305      Cod_m=7.0E-11
10310      CASE -8
10315      Tau50=2.2E-5
10320      Tau51=8.0E-5
10325      Tau_m=9.0E-4
10330      V_slew_m=4.E-3
10335      Cod_m=4.E-12
10340      CASE -9
10345      Tau50=8.0E-5
10350      Tau51=2.70E-4
10355      Tau_m=2.0E-3
10360      V_slew_m=2.4E-2
10365      Cod_m=2.E-12
10370      END SELECT
10375 SUBEND
```

Cal_hold_time

```
10390 SUB Cal_hold_time(I_s_max,I_range,V_set,Cap_s,Cod,Tau_m,V_slew,Hold_time,Wait_def)
10395   V_set_a=ABS(V_set)
10400   V_range=2
10405   IF V_set_a>2 THEN V_range=20
10410   IF V_set_a>20 THEN V_range=40
10415   IF V_set_a>40 THEN V_range=100
10420   IF V_set_a>100 THEN V_range=200
10425   Wait_def=3*Tau_m+(INT(V_set_a/V_range/2*13)+1)/16*V_slew*V_range/2
10430   Cap_dummy=10*I_range/580000
10435   Hold_time=(Cap_s+Cod+Cap_dummy)*V_set_a/I_s_max+4*Tau_m-Wait_def
10440   IF Hold_time<0 THEN Hold_time=0
10445 SUBEND
10450!
10455!
```

V_range

```
10460 SUB V_range(Rstart,Rstop,Vrange>Error_n)
10465   V_diff=ABS(Rstop-Rstart)
10470   Max_v=MAX(V_diff,ABS(Rstart),ABS(Rstop))
10475   Vrange=0
10480   SELECT Max_v
10485   CASE <=2
10490     IF V_diff>=.1 THEN Vrange=2
10495   CASE <2,<=20
10500     IF V_diff>=1 THEN Vrange=20
10505   CASE <20,<=40
10510     IF V_diff>=2 THEN Vrange=40
10515   CASE <40,<=100
10520     IF V_diff>=5 THEN Vrange=100
10525   CASE ELSE
10530     IF V_diff>=10 THEN Vrange=200
10535   END SELECT
10540   Error_n=0
10545   IF Vrange<1 THEN Error_n=30
10550   IF V_diff>200 THEN Error_n=31
10555 SUBEND
```

Set_bdm

```
10585 SUB Set_bdm(INTEGER Channel,REAL Vrange,Vstart,Vstop,Thold,Tsettle,OPTIONAL I_compliance)
10590   !
10595   COM @Hp4142
10600   !*****
10605   !*           Setup parameters           *
10610   !*****
10615   Rng=Vrange
10620   IF Rng<>0 THEN CALL Set_v_range((Channel),Rng)
10625   !
10630   SELECT NPAR
10635   CASE 6
10640     OUTPUT @Hp4142;"BDV";Channel,Rng,Vstart,Vstop
10645   CASE 7
10650     OUTPUT @Hp4142;"BDV";Channel,Rng,Vstart,Vstop,I_compliance
10655   END SELECT
10660   !
10665   OUTPUT @Hp4142;"BDT";Thold,Tsettle
10670   !
10675   OUTPUT @Hp4142;"*OPC?"
10680   ENTER @Hp4142;A$
10685   S=SPOLL(@Hp4142)
10690   IF BIT(S,5) THEN CALL Detect_error(S,"Set_bdm")
10695   !
10700 SUBEND
```

Measure_bdm

```
10720 SUB Measure_bdm(INTEGER Channel,Judge_mode,Measure_mode,REAL Measure_value,OPTIONAL INTEGER Status)
10725   !
10730   COM @Hp4142
10735   !*****
10740   !*           Setup measurement parameters           *
10745   !*****
10750   OUTPUT @Hp4142;"BDM";Judge_mode,Measure_mode
10755   OUTPUT @Hp4142;"MM";9,Channel
10760   OUTPUT @Hp4142;"IE"
10765   !
10770   OUTPUT @Hp4142;"*OPC?"
10775   ENTER @Hp4142;A$
10780   S=SPOLL(@Hp4142)
10785   IF BIT(S,5) THEN CALL Detect_error(S,"Measure_bdm")
10790   !*****
10795   !*           Get measure data and status data           *
10800   !*****
10805   IF NPAR<5 THEN
10810     ENTER @Hp4142;Measure_value
10815   ELSE
10820     ENTER @Hp4142;Data$
10825     Measure_value=VAL(Data$[4;12])
10830     Status$=Data$[1;1]
10835     IF Status$="N" THEN Status=0
10840     IF Status$="T" THEN Status=1
10845     IF Status$="C" THEN Status=2
10850     IF Status$="X" THEN Status=3
10855     IF Status$="V" THEN Status=4
10860     IF Status$="F" THEN Status=7
10865     IF Status$="G" THEN Status=8
10870     IF Status$="S" THEN Status=9
10875   END IF
10880   !
10885 SUBEND
```

Set_pol

```
10905 SUB Set_pol(INTEGER Channel,Mode)
10910  !
10915  COM @Hp4142
10920  !*****
10925  !*          Setup polaraty parameter          *
10930  !*****
10935  OUTPUT @Hp4142;"POL";Channel,Mode
10940  !
10945  OUTPUT @Hp4142;"*OPC?"
10950  ENTER @Hp4142;A$
10955  S=SPOLL(@Hp4142)
10960  IF BIT(S,5) THEN CALL Detect_error(S,"Set_pol")
10965  !
10970 SUBEND
```

Connect_relay1

```
10990 SUB Connect_relay1(OPTIONAL INTEGER Channel)
10995  !
11000  COM @Hp4142
11005  !*****
11010  !*          OUTPUT RELAY CHANNEL          *
11015  !*****
11020  SELECT NPAR
11025  CASE 0
11030    OUTPUT @Hp4142;"ERC";1,0,0
11035  CASE 1
11040    OUTPUT @Hp4142;"ERC";1,Channel,0
11045  END SELECT
11050  !
11055  OUTPUT @Hp4142;"*OPC?"
11060  ENTER @Hp4142;A$
11065  S=SPOLL(@Hp4142)
11070  IF BIT(S,5) THEN CALL Detect_error(S,"Connect_relay1")
11075  !
11080 SUBEND
```

Connect_relay2

```
11100 SUB Connect_relay2(INTEGER Mode,OPTIONAL INTEGER Port1,Port2,Port3,Port4,Port5,Port6,Port7,Port8,
Port9,Port10,Port11,Port12,Port13,Port14,Port15,Port16)
11105   !
11110   COM @Hp4142
11115   INTEGER Bit_val
11120   Bit_val=0
11125   Value=0
11130   !*****
11135   !*          ERROR CHECK AND SET BIT          *
11140   !*****
11145   IF NPAR>1 THEN
11150     IF Port1<1 OR 16<Port1 THEN GOSUB Connect_erc_err
11155     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port1-1)))
11160   END IF
11165   !
11170   IF NPAR>2 THEN
11175     IF Port2<1 OR 16<Port2 THEN GOSUB Connect_erc_err
11180     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port2-1)))
11185   END IF
11190   !
11195   IF NPAR>3 THEN
11200     IF Port3<1 OR 16<Port3 THEN GOSUB Connect_erc_err
11205     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port3-1)))
11210   END IF
11215   !
11220   IF NPAR>4 THEN
11225     IF Port4<1 OR 16<Port4 THEN GOSUB Connect_erc_err
11230     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port4-1)))
11235   END IF
11240   !
11245   IF NPAR>5 THEN
11250     IF Port5<1 OR 16<Port5 THEN GOSUB Connect_erc_err
11255     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port5-1)))
11260   END IF
11265   !
11270   IF NPAR>6 THEN
11275     IF Port6<1 OR 16<Port6 THEN GOSUB Connect_erc_err
11280     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port6-1)))
11285   END IF
11290   !
11295   IF NPAR>7 THEN
11300     IF Port7<1 OR 16<Port7 THEN GOSUB Connect_erc_err
11305     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port7-1)))
11310   END IF
11315   !
11320   IF NPAR>8 THEN
11325     IF Port8<1 OR 16<Port8 THEN GOSUB Connect_erc_err
11330     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port8-1)))
11335   END IF
11340   !
11345   IF NPAR>9 THEN
11350     IF Port9<1 OR 16<Port9 THEN GOSUB Connect_erc_err
11355     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port9-1)))
11360   END IF
11365   !
11370   IF NPAR>10 THEN
11375     IF Port10<1 OR 16<Port10 THEN GOSUB Connect_erc_err
11380     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port10-1)))
11385   END IF
11390   !
11395   IF NPAR>11 THEN
11400     IF Port11<1 OR 16<Port11 THEN GOSUB Connect_erc_err
```

```

11405     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port11-1)))
11410 END IF
11415 !
11420 IF NPAR>12 THEN
11425     IF Port12<1 OR 16<Port12 THEN GOSUB Connect_erc_err
11430     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port12-1)))
11435 END IF
11440 !
11445 IF NPAR>13 THEN
11450     IF Port13<1 OR 16<Port13 THEN GOSUB Connect_erc_err
11455     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port13-1)))
11460 END IF
11465 !
11470 IF NPAR>14 THEN
11475     IF Port14<1 OR 16<Port14 THEN GOSUB Connect_erc_err
11480     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port14-1)))
11485 END IF
11490 !
11495 IF NPAR>15 THEN
11500     IF Port15<1 OR 16<Port15 THEN GOSUB Connect_erc_err
11505     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port15-1)))
11510 END IF
11515 !
11520 IF NPAR>16 THEN
11525     IF Port16<1 OR 16<Port16 THEN GOSUB Connect_erc_err
11530     Bit_val=BINIOR(Bit_val,SHIFT(1,-(Port16-1)))
11535 END IF
11540 !*****
11545 !*          OUTPUT RELAY BIT          *
11550 !*****
11555 IF Bit_val<0 THEN
11560     Value=65535-BINCOMP(Bit_val)
11565 ELSE
11570     Value=Bit_val
11575 END IF
11580 !
11585 OUTPUT @Hp4142;"ERC";2,Value,Mode
11590 !
11595 OUTPUT @Hp4142;"*OPC?"
11600 ENTER @Hp4142;A$
11605 S=SPOLL(@Hp4142)
11610 IF BIT(S,5) THEN CALL Detect_error(S,"Connect_relay2")
11615 !
11620 SUBEXIT
11625 !*****
11630 !*          SELECT BIT ERROR          *
11635 !*****
11640 Connect_erc_err:
11645 DISP "CONNECT_RELAY2: PORT SELECT IS 1 TO 16  "
11650 BEEP
11655 STOP
11660 RETURN
11665 !
11670 !
11675 SUBEND

```

GRAPHICS File

Dump_screen

```
30  SUB Dump_screen(OPTIONAL INTEGER Expand_output)
35  !*****
40  !*          Dumps the graphic screen to a printer          *
45  !*****
50  INTEGER Dump_device
55  INTEGER Key_labels
60  Dump_device=701          ! HP-IB address of printer
65  !*****
70  !*          Setup the printer          *
75  !*****
80  ASSIGN @Printer TO Dump_device          ! Assign I/O path to printer
85  ON TIMEOUT SC(@Printer),20 GOTO No_printer
90  IF NPAR=1 THEN
95      SELECT Expand_output
100     CASE 1
105         DUMP DEVICE IS Dump_device,EXPANDED
110     CASE 0
115         DUMP DEVICE IS Dump_device
120     CASE ELSE
125         Disp_error("Dump_screen: Illegal parameter has been passed")
130         STOP
135     END SELECT
140 END IF
145 !
150 IF NPAR=0 THEN
155     DUMP DEVICE IS Dump_device
160 END IF
165 !*****
170 !*          Dump the screen          *
175 !*****
180 DUMP GRAPHICS          ! Dump the graphics page
185 OFF TIMEOUT          ! Disable timeout
190 SUBEXIT
195 !
200 No_printer: !
205 Disp_error("Dump_screen: Printer is not responding")
210 ASSIGN @Printer TO *          ! Deassign I/O path
215 OFF TIMEOUT          ! Disable timeout
220 SUBEND
```

Init_computer

```

235 SUB Init_computer(OPTIONAL INTEGER Separate_pages)
240 !*****
245 !*      Initializes the computer and its display      *
250 !*****
255 INTEGER Alpha_planes
260 INTEGER Graphics_planes(1:1)
265 !*****
270 !*      Clear the display      *
275 !*****
280 OUTPUT KBD;CHR$(255);"K";      ! Clear the screen
285 GINIT      ! Initialize graphics page
290 GCLEAR      ! Clear the graphics page
295 ALPHA ON      ! Enable alpha page
300 GRAPHICS ON      ! Enable graphics page
305 CONTROL CRT,12;1      ! Turn off softkey labels
310 !*****
315 !*      Set keyboard auto-repeat parameters      *
320 !*****
325 CONTROL KBD,3;2.5      ! Auto-repeat interval = 25 ms
330 CONTROL KBD,4;20      ! Auto-repeat delay = 200 ms
335 !*****
340 !*      Check the display color or monochrome      *
345 !*****
350 Crt_type$=TRIM$(SYSTEM$("CRT ID"))      ! Get the CRT ID
355 !*****
360 !*      Intialize the monochrome display both bit-mapped and      *
365 !*      non-bit-mapped display      *
370 !*      *
375 !*      @ non-bit-mapped CRT ... HP9000 200 series 9836/9816      *
380 !*      @ bit-mapped CRT ... HP9000 300 series 310/320      *
385 !*****
390 IF (Crt_type$[7,9]=" GB") OR (Crt_type$[7,9]=" G") THEN
395   IF NPAR THEN
400     Disp_error("Init_computer: Argument is not specified for the monochrome display")
405     STOP
410   ELSE
415     SUBEXIT
420   END IF
425 END IF
430 !*****
435 !*      Intialize the non-bit-mapped color display      *
440 !*      - for HP9000 200 series 9836C -      *
445 !*****
450 IF Crt_type$[7,9]="CG" THEN
455   IF NPAR THEN
460     Disp_error("Init_computer: Argument is not specified for the non-bit-mapped color display")
465     STOP
470   ELSE
475     PLOTTER IS CRT,"INTERNAL";COLOR MAP      ! Define the colors and pens
480     CONTROL 1,5;138      ! Set yellow alpha characters
485     SET PEN 0 INTENSITY 0,2/15,7/15      ! Blue background
490   END IF
495 END IF
500 !*****
505 !*      Intialize the bit-mapped color display      *
510 !*      - for HP9000 300 series color CRT -      *
515 !*****
520 IF Crt_type$[7,9]="CGB" THEN
525   PLOTTER IS CRT,"INTERNAL";COLOR MAP
530   IF NPAR THEN
535     IF Separate_pages=1 THEN
540       SET PEN 8 INTENSITY 1,1,0      ! Yellow

```

```

545     CONTROL CRT,5;0           ! Turn off the cursor (temp.)
550     CONTROL CRT,18;IVAL("1000",2) ! Select plane 4 for alpha
555     CONTROL CRT,5;8           ! Yellow alpha
560     Graphics_planes(1)=IVAL("0111",2) ! Select planes 1 - 3 for
565     GESCAPE CRT,7,Graphics_planes(*) ! graphics.
570 END IF
575 !*****
580 !*           Alpha and graphics will overlap           *
585 !*****
590 IF Separate_pages=0 THEN
595     STATUS 1,18;Alpha_planes ! Read alpha write-enable mask
600     IF Alpha_planes<>IVAL("1111",2) THEN
605         CONTROL CRT,18;IVAL("1111",2)
610         Graphics_planes(1)=IVAL("1111",2)
615         GESCAPE CRT,7,Graphics_planes(*)
620         CONTROL CRT,5;3 ! Yellow alpha
625     END IF
630 END IF
635 !
640 IF (Separate_pages<0) OR (Separate_pages>1) THEN
645     Disp_error("Init_computer: Illegal argument has been passed")
650     STOP
655 END IF
660 END IF
665 SET PEN 0 INTENSITY 0,2/15,5/15 ! Blue background
670 END IF
675 SUBEND

```

Wbuild_file

```

690 SUB Wbuild_file(Filename$,Meas_data*),File_comment$,Variable_names$(*))
695 !*****
700 !* Stores data in a file that conforms to the Basic Statistics and *
705 !* Data Manipulation (BSDM) format. The HP4062B uses a similar method.*
710 !* The data can be evaluated by the HP98820A Statistical Library. *
715 !*****
720 ALLOCATE Subfile_names$(1:20)[10]
725 ALLOCATE Var_names$(1:50)[10]
730 INTEGER I
735 INTEGER Meas_data_base ! Measurement data base
740 INTEGER Var_names_base ! Variable name base
745 REAL Number_obs ! Number of observations
750 REAL Number_subfiles ! Number of subfiles
755 REAL Number_var ! Number of variables
760 REAL Subfile_char(1:20)
765 !
770 Meas_data_base=BASE(Meas_data,1)
775 Var_names_base=BASE(Variable_names$,1)
780 !*****
785 !* Check the length of the title *
790 !*****
795 IF LEN(File_comment$)>80 THEN
800 Disp_error(" Wbuild_file: File comment is longer than 80 characters")
805 STOP
810 END IF
815 !*****
820 !* Check the size of the data array *
825 !*****
830 Number_var=SIZE(Meas_data,1)
835 IF Number_var>50 THEN
840 Disp_error("Wbuild_file: Too Many Variables (over 50)")
845 STOP
850 END IF
855 !
860 Number_obs=SIZE(Meas_data,2)
865 IF Number_var*Number_obs>1500 THEN
870 Disp_error("Wbuild_file: More than 1500 data points were provided")
875 STOP
880 END IF
885 !*****
890 !* Check the number of variable names *
895 !*****
900 SELECT SIZE(Variable_names$,1)
905 CASE <Number_var
910 Disp_error("Wbuild_file: Not Enough Variable Names")
915 STOP
920 CASE >Number_var
925 Disp_error("Wbuild_file: Too many variable names")
930 STOP
935 END SELECT
940 !*****
945 !* Check the length of the variable names *
950 !*****
955 IF LEN(Variable_names$(Var_names_base))>10 THEN
960 Disp_error("Wbuild_file: The variable names are longer than 10 characters")
965 STOP
970 END IF
975 !*****
980 !* Process the variable names *
985 !*****
990 FOR I=1 TO Number_var
995 Var_names$(I)=Variable_names$(Var_names_base+I)[1;MIN(10,LEN(Variable_names$(Var_names_base)))]

```

```

1000  NEXT I
1005  !
1010  FOR I=Number_var+1 TO 50
1015    Var_names$(I)=""
1020  NEXT I
1025  !*****
1030  !* Initialize the subfile variables. Although the statistical library *
1035  !* allows subfiles, they are not used in this subprogram.          *
1040  !*****
1045  Number_subfiles=0
1050  MAT Subfile_names$= ("" )
1055  MAT Subfile_char= (0)
1060  !*****
1065  !* Create the data file and assign an I/O path                      *
1070  !*****
1075  ON ERROR GOTO Duplicate_file      ! Prevent duplicate filenames
1080  CREATE BDAT Filename$,2+(8*Number_var*Number_obs DIV 1280),1280
1085  OFF ERROR
1090  ASSIGN @File TO Filename$        ! Assign I/O path
1095  !*****
1100  !* Store the data in the file                                         *
1105  !*****
1110  OUTPUT @File,1;File_comment$[1;MIN(80,LEN(File_comment$))],Number_obs,Number_var,Var_names$(*),
Number_subfiles,Subfile_names$(*),Subfile_char(*)
1115  OUTPUT @File,2                    ! Point to record 2
1120  OUTPUT @File;Meas_data(*)         ! Store data
1125  ASSIGN @File TO *                ! Close the file
1130  SUBEXIT
1135  !
1140 Duplicate_file:
1145  OFF ERROR
1150  ALLOCATE E$[80]
1155  E$=ERRM$
1160  Disp_error("Wbuild_file: "&E$[19;37]&" ")
1165  STOP
1170 SUBEND

```

Lingraph

```
1185 SUB Lingraph(X_min,X_max,Y_min,Y_max,X_axes_name$,Y_axes_name$,Graph_name$,OPTIONAL INTEGER Grid,
Quadrant_mode,Right_subtitle$,Left_subtitle$)
1190 !*****
1195 !*          Set the parameters of linear scale graph          *
1200 !*****
1205 ALLOCATE Titles$(1:5)[40]
1210 INTEGER Grid_pattern
1215 INTEGER Quadrant
1220 !*****
1225 !*                      Error check                      *
1230 !*****
1235 IF (Y_min=Y_max) OR (X_min=X_max) THEN
1240   Disp_error("Lingraph: Minimum and maximun values are duplicate  ( check the X or Y value )")
1245   STOP
1250 END IF
1255 IF LEN(X_axes_name$)>30 OR LEN(Y_axes_name$)>30 OR LEN(Graph_name$)>35 THEN
1260   Disp_error("Lingraph: Graph name or axes name is too long")
1265   STOP
1270 END IF
1275 IF NPAR>=8 THEN
1280   IF Grid<0 OR Grid>2 THEN
1285     Disp_error("Lingraph: Illegal Grid control parameter")
1290     STOP
1295   END IF
1300 END IF
1305 IF NPAR>=9 THEN
1310   IF Quadrant_mode<0 OR Quadrant_mode>4 THEN
1315     Disp_error("Lingraph: Illegal Quadrant mode parameter")
1320     STOP
1325   END IF
1330 END IF
1335 IF NPAR>=10 THEN
1340   IF LEN(Right_subtitle$)>20 THEN
1345     Disp_error("Lingraph: Right_subtitle is too long")
1350     STOP
1355   END IF
1360 END IF
1365 IF NPAR=11 THEN
1370   IF LEN(Left_subtitle$)>20 THEN
1375     Disp_error("Lingraph: Left_subtitle is too long")
1380     STOP
1385   END IF
1390 END IF
1395 !
1400 Xmin=X_min
1405 Xmax=X_max
1410 Ymin=Y_min
1415 Ymax=Y_max
1420 Titles$(1)=Graph_name$
1425 Titles$(2)=""
1430 Titles$(3)=""
1435 Titles$(4)=X_axes_name$
1440 Titles$(5)=Y_axes_name$
1445 Grid_pattern=0
1450 Quadrant=0
1455 !
1460 IF NPAR>=8 THEN Grid_pattern=Grid
1465 IF NPAR>=9 THEN Quadrant=Quadrant_mode
1470 IF NPAR>=10 THEN Titles$(3)=Right_subtitle$
1475 IF NPAR=11 THEN Titles$(2)=Left_subtitle$
1480 !*****
1485 !*                      Darw linear scale graph                      *
```

```
1490      !*****
1495      Draw_lin(Xmin,Xmax,Ymin,Ymax,Titles$(*),Grid_pattern,Quadrant)
1500 SUBEND
```

Loggraph

```

1515 SUB Loggraph(X_min,X_max,Y_min,Y_max,X_axes_name$,Y_axes_name$,Graph_name$,OPTIONAL INTEGER
      Graph_mode,Grid,Quadrant_mode,Right_subtitle$,Left_subtitle$)
1520 !*****
1525 !*          Set the parameters of log scale graph          *
1530 !*****
1535 ALLOCATE Titles$(1:5)[50]
1540 INTEGER Graph_pattern
1545 INTEGER Grid_pattern
1550 INTEGER Quadrant
1555 INTEGER Y_polarity          ! Polarity flag of y-axis
1560 INTEGER Y_val_exchange      ! Exchange flag of Y_min and Y_max
1565 INTEGER X_polarity          ! Polarity flag of x-axis
1570 INTEGER X_val_exchange      ! Exchange flag of X_min and X_max
1575 !*****
1580 !*          Error check          *
1585 !*****
1590 Graph_pattern=1
1595 IF NPAR>=8 THEN Graph_pattern=Graph_mode
1600 !
1605 IF Graph_pattern=1 OR Graph_pattern=3 THEN
1610   IF Y_min=0 OR Y_max=0 THEN
1615     Disp_error("Loggraph: 0 is not available for Y_min or Y_max value")
1620     STOP
1625   END IF
1630   IF Y_max<0 AND Y_min>0 OR Y_max>0 AND Y_min<0 THEN
1635     Disp_error("Loggraph: Y_min and Y_max must be same polarity")
1640     STOP
1645   END IF
1650 END IF
1655 !
1660 IF Graph_pattern=2 OR Graph_pattern=3 THEN
1665   IF X_min=0 OR X_max=0 THEN
1670     Disp_error("Loggraph: 0 is not available for X_min or X_max value")
1675     STOP
1680   END IF
1685   IF X_max<0 AND X_min>0 OR X_max>0 AND X_min<0 THEN
1690     Disp_error("Loggraph: X_min and X_max must be same polarity")
1695     STOP
1700   END IF
1705 END IF
1710 !
1715 IF Y_min=Y_max OR X_min=X_max THEN
1720   Disp_error("Loggraph: Minimum and maximum values are duplicate ( check the X or Y value )")
1725   STOP
1730 END IF
1735 IF LEN(X_axes_name$)>30 OR LEN(Y_axes_name$)>30 OR LEN(Graph_name$)>35 THEN
1740   Disp_error("Loggraph: Graph name or axes name is too long")
1745   STOP
1750 END IF
1755 IF NPAR>=8 THEN
1760   IF Graph_mode>3 OR Graph_mode<1 THEN
1765     Disp_error("Loggraph: Illegal Graph mode parameter")
1770     STOP
1775   END IF
1780 END IF
1785 IF NPAR>=9 THEN
1790   IF Grid<0 OR Grid>2 THEN
1795     Disp_error("Loggraph: Illegal Grid control parameter")
1800     STOP
1805   END IF
1810 END IF
1815 IF NPAR>=10 THEN

```

```

1820     IF Quadrant_mode<0 OR Quadrant_mode>4 THEN
1825         Disp_error("Loggraph: Illegal Quadrant mode parameter")
1830         STOP
1835     END IF
1840 END IF
1845 IF NPAR>=11 THEN
1850     IF LEN(Right_subtitle$)>20 THEN
1855         Disp_error("Loggraph: Right_subtitle is too long")
1860         STOP
1865     END IF
1870 END IF
1875 IF NPAR=12 THEN
1880     IF LEN(Left_subtitle$)>20 THEN
1885         Disp_error("Loggraph: Left_subtitle is too long")
1890         STOP
1895     END IF
1900 END IF
1905 !
1910 Xmin=X_min
1915 Xmax=X_max
1920 Ymin=Y_min
1925 Ymax=Y_max
1930 Titles$(1)=Graph_name$
1935 Titles$(2)=""
1940 Titles$(3)=""
1945 Titles$(4)=X_axes_name$
1950 Titles$(5)=Y_axes_name$
1955 Grid_pattern=0
1960 Y_polarity=0
1965 Y_val_exchange=0
1970 X_polarity=0
1975 X_val_exchange=0
1980 !
1985 IF Graph_pattern=1 THEN
1990     IF Ymax<Ymin THEN Y_val_exchange=1
1995     IF Ymax<0 AND Ymin<0 THEN Y_polarity=1
2000     IF NPAR>=9 THEN Grid_pattern=Grid
2005     IF NPAR>=10 THEN Quadrant=Quadrant_mode
2010     IF NPAR>=11 THEN Titles$(3)=Right_subtitle$
2015     IF NPAR=12 THEN Titles$(2)=Left_subtitle$
2020 END IF
2025 !
2030 IF Graph_pattern=2 THEN
2035     IF Xmax<Xmin THEN X_val_exchange=1
2040     IF Xmax<0 AND Xmin<0 THEN X_polarity=1
2045     IF NPAR>=9 THEN Grid_pattern=Grid
2050     IF NPAR>=10 THEN Quadrant=Quadrant_mode
2055     IF NPAR>=11 THEN Titles$(3)=Right_subtitle$
2060     IF NPAR=12 THEN Titles$(2)=Left_subtitle$
2065 END IF
2070 !
2075 IF Graph_pattern=3 THEN
2080     IF Ymax<Ymin THEN Y_val_exchange=1
2085     IF Ymax<0 AND Ymin<0 THEN Y_polarity=1
2090     IF Xmax<Xmin THEN X_val_exchange=1
2095     IF Xmax<0 AND Xmin<0 THEN X_polarity=1
2100     IF NPAR>=9 THEN Grid_pattern=Grid
2105     IF NPAR>=10 THEN Quadrant=Quadrant_mode
2110     IF NPAR>=11 THEN Titles$(3)=Right_subtitle$
2115     IF NPAR=12 THEN Titles$(2)=Left_subtitle$
2120 END IF
2125 !*****
2130 !*                Darw log scale graph                *
2135 !*****
2140 IF Graph_pattern=1 THEN
2145     Draw_log1(Xmin,Xmax,Ymin,Ymax,Titles$(*),Grid_pattern,Quadrant,Y_polarity,Y_val_exchange)

```

```

2150     END IF
2155     !
2160     IF Graph_pattern=2 THEN
2165         Draw_log2(Xmin,Xmax,Ymin,Ymax,Titles$(*),Grid_pattern,Quadrant,X_polarity,X_val_exchange)
2170     END IF
2175     !
2180     IF Graph_pattern=3 THEN
2185         Draw_log3(Xmin,Xmax,Ymin,Ymax,Titles$(*),Grid_pattern,Quadrant,X_polarity,X_val_exchange,
Y_polarity,Y_val_exchange)
2190     END IF
2195 SUBEND

```

Draw_lin

```
2210 SUB Draw_lin(Xmin,Xmax,Ymin,Ymax,Titles$(*),INTEGER Grid_pattern,Quadrant)
2215 !*****
2220 !*      Draws horizontal and vertical axes with linear scales      *
2225 !*****
2230 INTEGER I
2235 INTEGER Minor_ticks
2240 INTEGER Quarter_screen
2245 INTEGER Title_base
2250 !*****
2255 !* Determine the plot area and the number of minor tick marks per *
2260 !* major division.                                           *
2265 !*****
2270 Quarter_screen=Quadrant
2275 SELECT Quadrant
2280 CASE 0
2285     Minor_ticks=5
2290 CASE 1,2,3,4
2295     Minor_ticks=4
2300 END SELECT
2305 GRAPHICS ON                      ! Enable graphics page
2310 WINDOW 0,100,0,100              ! Fixes a BASIC 3.01 bug
2315 !*****
2320 !*      Zoom to the plotting area and write the titles      *
2325 !*****
2330 Zoom_plot_area(Quarter_screen)    ! Zoom to the plotting area.
2335 Title_base=BASE(Titles$,1)
2340 Write_title(Titles$(Title_base),Titles$(Title_base+1),Titles$(Title_base+2),Quarter_screen)
2345 !*****
2350 !*      Zoom to the graph area and write the axes names      *
2355 !*****
2360 Zoom_graph_area(Quarter_screen)    ! Zoom to the graph area.
2365 Write_axes_name(Titles$(Title_base+3),Titles$(Title_base+4),Quarter_screen)
2370 !*****
2375 !*      Scale the axes                                          *
2380 !*****
2385 Calc_lin_axis(Xmin,Xmax,Xaxis_min,Xaxis_max,Xmajor_div,Quarter_screen)
2390 Calc_lin_axis(Ymin,Ymax,Yaxis_min,Yaxis_max,Ymajor_div,Quarter_screen)
2395 !*****
2400 !*      Draw the axes                                          *
2405 !*****
2410 WINDOW Xaxis_min,Xaxis_max,Yaxis_min,Yaxis_max
2415 LINE TYPE 1                      ! Solid line
2420 PEN 1                             ! White
2425 IF Quarter_screen THEN
2430     AXES 0,0,Xaxis_max,Yaxis_max
2435     AXES 0,0,Xaxis_min,Yaxis_min
2440 ELSE
2445     AXES 0,0,Xaxis_max,Yaxis_max
2450     AXES 0,0,Xaxis_min,Yaxis_min
2455 END IF
2460 !*****
2465 !*      Label the tick marks                                  *
2470 !*****
2475 Label_lin_xaxis(Xaxis_min,Xaxis_max,Xmajor_div,Yaxis_min,Yaxis_max,Minor_ticks,Quarter_screen)
2480 Label_lin_yaxis(Xaxis_min,Xaxis_max,Yaxis_min,Yaxis_max,Ymajor_div,Minor_ticks,Quarter_screen)
2485 !*****
2490 !*      Draw the grid if necessary                              *
2495 !*****
2500 IF Grid_pattern THEN
2505     LINE TYPE 4
2510     IF Grid_pattern=2 THEN LINE TYPE 1
2515     !*****
```

```

2520      !*              draw vertical grid              *
2525      !*****
2530      IF Yaxis_min*Yaxis_max<0 THEN
2535          FOR Y1=0 TO Yaxis_max STEP Ymajor_div
2540              IF Y1<>Yaxis_max THEN
2545                  MOVE Xaxis_min,Y1
2550                  DRAW Xaxis_max,Y1
2555              END IF
2560          NEXT Y1
2565          FOR Y1=0-Ymajor_div TO Yaxis_min STEP -Ymajor_div
2570              IF Y1<>Yaxis_min THEN
2575                  MOVE Xaxis_min,Y1
2580                  DRAW Xaxis_max,Y1
2585              END IF
2590          NEXT Y1
2595      ELSE
2600          FOR Y1=Yaxis_min TO Yaxis_max STEP Ymajor_div
2605              IF Y1<>Yaxis_min AND Y1<>Yaxis_max THEN
2610                  MOVE Xaxis_min,Y1
2615                  DRAW Xaxis_max,Y1
2620              END IF
2625          NEXT Y1
2630      END IF
2635      !*****
2640      !*              draw horizontal grid              *
2645      !*****
2650      IF Xaxis_min*Xaxis_max<0 THEN
2655          FOR X1=0 TO Xaxis_max STEP Xmajor_div
2660              IF X1<>Xaxis_max THEN
2665                  MOVE X1,Yaxis_min
2670                  DRAW X1,Yaxis_max
2675              END IF
2680          NEXT X1
2685          FOR X1=0-Xmajor_div TO Xaxis_min STEP -Xmajor_div
2690              IF X1<>Xaxis_min THEN
2695                  MOVE X1,Yaxis_min
2700                  DRAW X1,Yaxis_max
2705              END IF
2710          NEXT X1
2715      ELSE
2720          FOR X1=Xaxis_min TO Xaxis_max STEP Xmajor_div
2725              IF X1<>Xaxis_min AND X1<>Xaxis_max THEN
2730                  MOVE X1,Yaxis_min
2735                  DRAW X1,Yaxis_max
2740              END IF
2745          NEXT X1
2750      END IF
2755      END IF
2760      !*****
2765      !*              Move the pen to the bottom left corner      *
2770      !*****
2775      LINE TYPE 1
2780      MOVE Xaxis_min,Yaxis_min
2785      SUBEND

```

Draw_log1

```
2800 SUB Draw_log1(Xmin,Xmax,Ymin,Ymax,Titles$(*),INTEGER Grid_pattern,Quadrant,Y_polarity,Y_val_exchange)
2805 !*****
2810 !*   The vertical axis will have a logarithmic (base 10) scale,      *
2815 !*   while the horizontal axis will have a linear scale.            *
2820 !*****
2825   INTEGER I,Mantissa
2830   INTEGER Title_base,Quarter_screen
2835   INTEGER Xminor_tick
2840   INTEGER Yaxis_min,Yaxis_max
2845   INTEGER Yminor_tick,Ymajor_tick
2850   INTEGER Y1,Y2
2855 !*****
2860 !*   Determine the plot area and number of x-axis minor tick marks  *
2865 !*   per major division.                                           *
2870 !*****
2875   Quarter_screen=Quadrant
2880   SELECT Quadrant
2885   CASE 0
2890     Xminor_tick=5
2895   CASE 1,2,3,4
2900     Xminor_tick=4
2905   END SELECT
2910   GRAPHICS ON                      ! Enable graphics page
2915   WINDOW 0,100,0,100              ! Fixes a BASIC 3.01 bug
2920 !*****
2925 !*   Zoom to the plotting area and write the titles                *
2930 !*****
2935   Zoom_plot_area(Quarter_screen)   ! Zoom to the plotting area.
2940   Title_base=BASE(Titles$,1)
2945   Write_title(Titles$(Title_base),Titles$(Title_base+1),Titles$(Title_base+2),Quarter_screen)
2950 !*****
2955 !*   Zoom to the graph area and write the axes name                *
2960 !*****
2965   Zoom_graph_area(Quarter_screen) ! Zoom to the graph area.
2970   Write_axes_name(Titles$(Title_base+3),Titles$(Title_base+4),Quarter_screen)
2975 !*****
2980 !*   Scale the axes                                                  *
2985 !*****
2990   Calc_lin_axis(Xmin,Xmax,Xaxis_min,Xaxis_max,Xmajor_div,Quarter_screen)
2995   Calc_log_axis(Ymin,Ymax,Yaxis_min,Yaxis_max,Ymajor_tick,Yminor_tick,Quarter_screen)
3000 !*****
3005 !*   Draw the axes                                                  *
3010 !*****
3015   WINDOW Xaxis_min,Xaxis_max,Yaxis_min,Yaxis_max
3020   LINE TYPE 1
3025   PEN 1                          ! White
3030   AXES 0,0,Xaxis_max,Yaxis_max
3035   AXES 0,0,Xaxis_min,Yaxis_min
3040 !*****
3045 !*   Label the tick marks                                           *
3050 !*****
3055   Label_lin_xaxis((Xaxis_min),(Xaxis_max),(Xmajor_div),(Yaxis_min),(Yaxis_max),Xminor_tick,
Quarter_screen)
3060   Label_log_yaxis((Xaxis_min),(Xaxis_max),(Yaxis_min),(Yaxis_max),(Ymajor_tick),(Yminor_tick),
Quarter_screen,Y_polarity,Y_val_exchange,Grid_pattern)
3065 !*****
3070 !*   Draw the grid if necessary  -- draw vertical grid --          *
3075 !*****
3080   IF Grid_pattern THEN
3085     GOSUB Line_type
3090     IF Xaxis_min*Xaxis_max<0 THEN
3095       FOR I1=0 TO Xaxis_max STEP Xmajor_div
```

```

3100     IF X1<>Xaxis_max THEN
3105         MOVE X1,Yaxis_min
3110         DRAW X1,Yaxis_max
3115     END IF
3120     NEXT X1
3125     FOR X1=0-Xmajor_div TO Xaxis_min STEP -Xmajor_div
3130         IF X1<>Xaxis_min THEN
3135             MOVE X1,Yaxis_min
3140             DRAW X1,Yaxis_max
3145         END IF
3150     NEXT X1
3155     ELSE
3160         FOR X1=Xaxis_min TO Xaxis_max STEP Xmajor_div
3165             IF X1<>Xaxis_min AND X1<>Xaxis_max THEN
3170                 MOVE X1,Yaxis_min
3175                 DRAW X1,Yaxis_max
3180             END IF
3185         NEXT X1
3190     END IF
3195     !*****
3200     !*      Draw the grid if necessary    -- draw horizontal grid -- *
3205     !*****
3210     LINE TYPE 1
3215     MOVE Xaxis_min,Yaxis_min
3220     IF Yminor_tick=0 THEN GOSUB Line_type
3225     IF Ymin>Ymax THEN Ymajor_tick=-Ymajor_tick
3230     FOR Y1=Yaxis_min TO Yaxis_max STEP Ymajor_tick
3235         IF Y1<>Yaxis_min AND Y1<>Yaxis_max THEN
3240             MOVE Xaxis_min,Y1
3245             DRAW Xaxis_max,Y1
3250         END IF
3255     NEXT Y1
3260     !*****
3265     !*      Draw the grid line(s) between each decade      *
3270     !*****
3275     GOSUB Line_type
3280     IF Yminor_tick<>0 THEN
3285         MOVE Xaxis_min,Yaxis_min
3290         Y_start=MIN(Yaxis_min,Yaxis_max)
3295         Y_stop=MAX(Yaxis_min,Yaxis_max)
3300         IF Y_polarity=0 THEN Y_stop=Y_stop-1
3305         IF Y_polarity=1 THEN Y_start=Y_start+1
3310     !
3315     FOR Exponent=Y_start TO Y_stop
3320         FOR Mantissa=2 TO 9
3325             IF Y_polarity=0 THEN Y_coordinate=LGT(Mantissa*(10^Exponent))
3330             IF Y_polarity=1 THEN Y_coordinate=-LGT(Mantissa*(10^(-Exponent)))
3335             MOVE Xaxis_min,Y_coordinate
3340             DRAW Xaxis_max,Y_coordinate
3345         NEXT Mantissa
3350     NEXT Exponent
3355     END IF
3360     END IF
3365     LINE TYPE 1
3370     !*****
3375     !*      Move the pen to the bottom left corner      *
3380     !*****
3385     MOVE Xaxis_min,Yaxis_min
3390     SUBEXIT
3395     !
3400 Line_type: !
3405     LINE TYPE 4
3410     IF Grid_pattern=2 THEN LINE TYPE 1
3415     RETURN

```

3420 SUBEND

Draw_log2

```

3435 SUB Draw_log2(Xmin,Xmax,Ymin,Ymax,Titles$(*),INTEGER Grid_pattern,Quadrant,X_polarity,X_val_exchange)
3440 !*****
3445 !*   The horizontal axis will have a logarithmic (base 10) scale,   *
3450 !*   while the vertical axis will have a linear scale.           *
3455 !*****
3460   INTEGER I,Mantissa
3465   INTEGER Title_base,Quarter_screen
3470   INTEGER Yminor_tick
3475   INTEGER Xaxis_min,Xaxis_max
3480   INTEGER Xminor_tick,Xmajor_tick
3485   INTEGER X1,X2
3490 !*****
3495 !*   Determine the plot area and number of x-axis minor tick marks *
3500 !*   per major division.                                           *
3505 !*****
3510   Quarter_screen=Quadrant
3515   SELECT Quadrant
3520     CASE 0
3525       Yminor_tick=5
3530     CASE 1,2,3,4
3535       Yminor_tick=4
3540   END SELECT
3545   GRAPHICS ON                      ! Enable graphics page
3550   WINDOW 0,100,0,100              ! Fixes a BASIC 3.01 bug
3555 !*****
3560 !*   Zoom to the plotting area and write the titles               *
3565 !*****
3570   Zoom_plot_area(Quarter_screen)    ! Zoom to the plotting area.
3575   Title_base=BASE(Titles$,1)
3580   Write_title(Titles$(Title_base),Titles$(Title_base+1),Titles$(Title_base+2),Quarter_screen)
3585 !*****
3590 !*   Zoom to the graph area and write the axes name              *
3595 !*****
3600   Zoom_graph_area(Quarter_screen)   ! Zoom to the graph area.
3605   Write_axes_name(Titles$(Title_base+3),Titles$(Title_base+4),Quarter_screen)
3610 !*****
3615 !*   Scale the axes                                               *
3620 !*****
3625   Calc_lin_axis(Ymin,Ymax,Yaxis_min,Yaxis_max,Ymajor_div,Quarter_screen)
3630   Calc_log_axis(Xmin,Xmax,Xaxis_min,Xaxis_max,Xmajor_tick,Xminor_tick,Quarter_screen)
3635 !*****
3640 !*   Draw the axes                                               *
3645 !*****
3650   WINDOW Xaxis_min,Xaxis_max,Yaxis_min,Yaxis_max
3655   LINE TYPE 1
3660   PEN 1                          ! White
3665   AXES 0,0,Xaxis_max,Yaxis_max
3670   AXES 0,0,Xaxis_min,Yaxis_min
3675 !*****
3680 !*   Label the tick marks                                         *
3685 !*****
3690   Label_lin_yaxis((Xaxis_min),(Xaxis_max),(Yaxis_min),(Yaxis_max),(Ymajor_div),Yminor_tick,
Quarter_screen)
3695   Label_log_xaxis((Xaxis_min),(Xaxis_max),(Yaxis_min),(Yaxis_max),(Xmajor_tick),(Xminor_tick),
Quarter_screen,X_polarity,X_val_exchange,Grid_pattern)
3700 !*****
3705 !*   Draw the grid if necessary  -- draw vertical grid --      *
3710 !*****
3715   IF Grid_pattern THEN
3720     GOSUB Line_type
3725     IF Yaxis_min*Yaxis_max<0 THEN
3730       FOR Y1=0 TO Yaxis_max STEP Ymajor_div

```

```

3735     IF Y1<>Yaxis_max THEN
3740         MOVE Xaxis_min,Y1
3745         DRAW Xaxis_max,Y1
3750     END IF
3755     NEXT Y1
3760     FOR Y1=0-Ymajor_div TO Yaxis_min STEP -Ymajor_div
3765         IF Y1<>Yaxis_min THEN
3770             MOVE Xaxis_min,Y1
3775             DRAW Xaxis_max,Y1
3780         END IF
3785     NEXT Y1
3790     ELSE
3795         FOR Y1=Yaxis_min TO Yaxis_max STEP Ymajor_div
3800             IF Y1<>Yaxis_min AND Y1<>Yaxis_max THEN
3805                 MOVE Xaxis_min,Y1
3810                 DRAW Xaxis_max,Y1
3815             END IF
3820         NEXT Y1
3825     END IF
3830 *****
3835     !*   Draw the grid if necessary   -- draw horizontal grid -- *
3840 *****
3845     LINE TYPE 1
3850     MOVE Xaxis_min,Yaxis_min
3855     IF Xminor_tick=0 THEN GOSUB Line_type
3860     IF Xmin>Xmax THEN Xmajor_tick=-Xmajor_tick
3865     FOR X1=Xaxis_min TO Xaxis_max STEP Xmajor_tick
3870         IF X1<>Xaxis_min AND X1<>Xaxis_max THEN
3875             MOVE X1,Yaxis_min
3880             DRAW X1,Yaxis_max
3885         END IF
3890     NEXT X1
3895 *****
3900     !*   Draw the grid line(s) between each decade   *
3905 *****
3910     GOSUB Line_type
3915     IF Xminor_tick<>0 THEN
3920         MOVE Xaxis_min,Yaxis_min
3925         X_start=MIN(Xaxis_min,Xaxis_max)
3930         X_stop=MAX(Xaxis_min,Xaxis_max)
3935         IF X_polarity=0 THEN X_stop=X_stop-1
3940         IF X_polarity=1 THEN X_start=X_start+1
3945     !
3950     FOR Exponent=X_start TO X_stop
3955         FOR Mantissa=2 TO 9
3960             IF X_polarity=0 THEN X_coordinate=LGT(Mantissa*(10^Exponent))
3965             IF X_polarity=1 THEN X_coordinate=-LGT(Mantissa*(10^(-Exponent)))
3970             MOVE X_coordinate,Yaxis_min
3975             DRAW X_coordinate,Yaxis_max
3980         NEXT Mantissa
3985     NEXT Exponent
3990     END IF
3995     END IF
4000     LINE TYPE 1
4005 *****
4010     !*   Move the pen to the bottom left corner   *
4015 *****
4020     MOVE Xaxis_min,Yaxis_min
4025     SUBEXIT
4030     !
4035     Line_type: !
4040     LINE TYPE 4
4045     IF Grid_pattern=2 THEN LINE TYPE 1
4050     RETURN

```

4055 SUBEND

Draw_log3

```
4070 SUB Draw_log3(Xmin,Xmax,Ymin,Ymax,Titles$(*),INTEGER Grid_pattern,Quadrant,X_polarity,X_val_exchange,
Y_polarity,Y_val_exchange)
4075 !*****
4080 !*   The vertical axis and horizontal axis will have logarithmic   *
4085 !*   (base 10) scale.                                           *
4090 !*****
4095   INTEGER I
4100   INTEGER Mantissa
4105   INTEGER Quarter_screen
4110   INTEGER Title_base
4115   INTEGER Xaxis_min,Xaxis_max
4120   INTEGER Xminor_tick,Xmajor_tick
4125   INTEGER X1,X2
4130   INTEGER Yaxis_min,Yaxis_max
4135   INTEGER Yminor_tick,Ymajor_tick
4140   INTEGER Y1,Y2
4145 !*****
4150 !*   Determine the plot area and number of x-axis minor tick marks *
4155 !*   per major division.                                           *
4160 !*****
4165   Quarter_screen=Quadrant
4170   SELECT Quadrant
4175   CASE 0
4180     Xminor_tick=5
4185     Yminor_tick=5
4190   CASE 1,2,3,4
4195     Xminor_tick=4
4200     Yminor_tick=4
4205   END SELECT
4210   GRAPHICS ON                      ! Enable graphics page
4215   WINDOW 0,100,0,100              ! Fixes a BASIC 3.01 bug
4220 !*****
4225 !*   Zoom to the plotting area and write the titles               *
4230 !*****
4235   Zoom_plot_area(Quarter_screen)   ! Zoom to the plotting area.
4240   Title_base=BASE(Titles$,1)
4245   Write_title(Titles$(Title_base),Titles$(Title_base+1),Titles$(Title_base+2),Quarter_screen)
4250 !*****
4255 !*   Zoom to the graph area and write the axes name              *
4260 !*****
4265   Zoom_graph_area(Quarter_screen) ! Zoom to the graph area.
4270   Write_axes_name(Titles$(Title_base+3),Titles$(Title_base+4),Quarter_screen)
4275 !*****
4280 !*   Scale the axes                                               *
4285 !*****
4290   Calc_log_axis(Xmin,Xmax,Xaxis_min,Xaxis_max,Xmajor_tick,Xminor_tick,Quarter_screen)
4295   Calc_log_axis(Ymin,Ymax,Yaxis_min,Yaxis_max,Ymajor_tick,Yminor_tick,Quarter_screen)
4300 !*****
4305 !*   Draw the axes                                               *
4310 !*****
4315   WINDOW Xaxis_min,Xaxis_max,Yaxis_min,Yaxis_max
4320   LINE TYPE 1
4325   PEN 1                          ! White
4330   AXES 0,0,Xaxis_max,Yaxis_max
4335   AXES 0,0,Xaxis_min,Yaxis_min
4340 !*****
4345 !*   Label the tick marks                                         *
4350 !*****
4355   Label_log_xaxis((Xaxis_min),(Xaxis_max),(Yaxis_min),(Yaxis_max),(Xmajor_tick),(Xminor_tick),
Quarter_screen,X_polarity,X_val_exchange,Grid_pattern)
4360   Label_log_yaxis((Xaxis_min),(Xaxis_max),(Yaxis_min),(Yaxis_max),(Ymajor_tick),(Yminor_tick),
Quarter_screen,Y_polarity,Y_val_exchange,Grid_pattern)
```

```

4365 !*****
4370 !*      Draw the grid if necessary  -- draw vertical grid --  *
4375 !*****
4380 IF Grid_pattern THEN
4385     LINE TYPE 1
4390     IF Xminor_tick=0 THEN GOSUB Line_type
4395     MOVE Xaxis_min,Yaxis_min
4400     IF Xmin>Xmax THEN Xmajor_tick=-Xmajor_tick
4405     FOR X1=Xaxis_min TO Xaxis_max STEP Xmajor_tick
4410         IF X1<>Xaxis_min AND X1<>Xaxis_max THEN
4415             MOVE X1,Yaxis_min
4420             DRAW X1,Yaxis_max
4425         END IF
4430     NEXT X1
4435 !*****
4440 !*      Draw the grid line(s) between each decade      *
4445 !*****
4450     IF Xminor_tick<>0 THEN
4455         GOSUB Line_type
4460         IF Yminor_tick=0 THEN GOSUB Line_type
4465         MOVE Xaxis_min,Yaxis_min
4470         X_start=MIN(Xaxis_min,Xaxis_max)
4475         X_stop=MAX(Xaxis_min,Xaxis_max)
4480         !
4485         FOR Exponent=X_start TO X_stop
4490             FOR Mantissa=2 TO 9
4495                 IF X_polarity=0 THEN X_coordinate=LGT(Mantissa*(10^Exponent))
4500                 IF X_polarity=1 THEN X_coordinate=-LGT(Mantissa*(10^(-Exponent)))
4505                 MOVE X_coordinate,Yaxis_min
4510                 DRAW X_coordinate,Yaxis_max
4515             NEXT Mantissa
4520         NEXT Exponent
4525     END IF
4530 !*****
4535 !*      Draw the grid if necessary  -- draw horizontal grid --  *
4540 !*****
4545     LINE TYPE 1
4550     IF Yminor_tick=0 THEN GOSUB Line_type
4555     MOVE Xaxis_min,Yaxis_min
4560     IF Ymin>Ymax THEN Ymajor_tick=-Ymajor_tick
4565     FOR Y1=Yaxis_min TO Yaxis_max STEP Ymajor_tick
4570         IF Y1<>Yaxis_min AND Y1<>Yaxis_max THEN
4575             MOVE Xaxis_min,Y1
4580             DRAW Xaxis_max,Y1
4585         END IF
4590     NEXT Y1
4595 !*****
4600 !*      Draw the grid line(s) between each decade      *
4605 !*****
4610     IF Yminor_tick<>0 THEN
4615         GOSUB Line_type
4620         IF Xminor_tick=0 THEN GOSUB Line_type
4625         MOVE Xaxis_min,Yaxis_min
4630         Y_start=MIN(Yaxis_min,Yaxis_max)
4635         Y_stop=MAX(Yaxis_min,Yaxis_max)
4640         !
4645         FOR Exponent=Y_start TO Y_stop
4650             FOR Mantissa=2 TO 9
4655                 IF Y_polarity=0 THEN Y_coordinate=LGT(Mantissa*(10^Exponent))
4660                 IF Y_polarity=1 THEN Y_coordinate=-LGT(Mantissa*(10^(-Exponent)))
4665                 MOVE Xaxis_min,Y_coordinate
4670                 DRAW Xaxis_max,Y_coordinate
4675             NEXT Mantissa
4680         NEXT Exponent
4685     END IF
4690 END IF

```

```

4695  LINE TYPE 1
4700  !*****
4705  !*          Move the pen to the bottom left corner          *
4710  !*****
4715  MOVE Xaxis_min,Yaxis_min
4720  SUBEXIT
4725  !
4730 Line_type: !
4735  LINE TYPE 4
4740  IF Grid_pattern=2 THEN LINE TYPE 1
4745  RETURN
4750 SUBEND

```

Disp_error

```
4765 SUB Disp_error(Error_message$)
4770   BEEP
4775   !*****
4780   !*           Displays the given error message on the CRT           *
4785   !*****
4790   INTEGER Alpha_color
4795   INTEGER Alpha_planes
4800   !*****
4805   !*           For bit-mapped color display                           *
4810   !*****
4815   Crt_type$=TRIM$(SYSTEM$("CRT ID"))
4820   IF Crt_type$[7;9]="CGB" THEN
4825       STATUS 1,5;Alpha_color           ! Determine current alpha color
4830       STATUS 1,18;Alpha_planes        ! Read alpha write-enable mask
4835       IF Alpha_planes<>IVAL("1111",2) THEN
4840           DISP Error_message$
4845       ELSE
4850           DISP CHR$(137);Error_message$
4855           CONTROL 1,5;Alpha_color      ! Restore previous alpha color
4860       END IF
4865   ELSE
4870   !*****
4875   !*           For non-bit-mapped color display and monochrome display *
4880   !*****
4885       DISP Error_message$
4890   END IF
4895 SUBEND
```

Zoom_plot_area

```
4910 SUB Zoom_plot_area(INTEGER Quadrant)
4915 !*****
4920 !*           Zooms to the plotting area           *
4925 !*****
4930 X_left=0
4935 X_center=50*RATIO
4940 X_right=100*RATIO
4945 !*****
4950 !* If desired, leave space for main titles at the top of *
4955 !* the screen. *
4960 !*****
4965 Title_offset=0
4970 Y_bottom=0
4975 Y_center=(100-Title_offset)/2
4980 Y_top=100-Title_offset
4985 !*****
4990 !*           Zoom to the plotting area           *
4995 !*****
5000 SELECT Quadrant
5005 CASE 1
5010     VIEWPORT X_center,X_right,Y_center,Y_top
5015 CASE 2
5020     VIEWPORT X_left,X_center,Y_center,Y_top
5025 CASE 3
5030     VIEWPORT X_left,X_center,Y_bottom,Y_center
5035 CASE 4
5040     VIEWPORT X_center,X_right,Y_bottom,Y_center
5045 CASE ELSE
5050     VIEWPORT X_left,X_right,Y_bottom,Y_top
5055 END SELECT
5060 !
5065 PEN 7           ! Magenta
5070 FRAME           ! Frame the quadrant
5075 PEN 1           ! White
5080 SUBEND
```

Write_title

```
5095 SUB Write_title(Main_title$,Left_title$,Right_title$,INTEGER Quadrant)
5100 !*****
5105 !* Writes a title and two subtitles at the top of the current viewport *
5110 !*****
5115 INTEGER X_coordinate
5120 INTEGER Y_coordinate
5125 LINE TYPE 1                ! Solid line
5130 DEG                        ! All angles in degrees
5135 LDIR 0                     ! 0 degrees rotation
5140 !*****
5145 !* Determine the character size of the main title *
5150 !*****
5155 IF Quadrant THEN
5160 Character_size=3.25
5165 ELSE
5170 Character_size=5.1
5175 END IF
5180 !*****
5185 !* Write the main title *
5190 !*****
5195 CSIZE Character_size
5200 PEN 3                      ! Yellow
5205 LORG 6
5210 WINDOW 0,100,0,100
5215 MOVE 50,100
5220 LABEL Main_title$
5225 !*****
5230 !* Calculate the vertical position of the subtitles *
5235 !*****
5240 WHERE X_coordinate,Y_coordinate
5245 Y_coordinate=.90*Y_coordinate+10
5250 CSIZE .7*Character_size    ! Reduce character size
5255 !*****
5260 !* Write the subtitles *
5265 !*****
5270 LORG 1
5275 MOVE 1,Y_coordinate-5
5280 LABEL Left_title$
5285 LORG 7
5290 MOVE 99,Y_coordinate-5
5295 LABEL Right_title$
5300 SUBEND
```

Zoom_graph_area

```
5315 SUB Zoom_graph_area(INTEGER Quadrant)
5320 !*****
5325 !*                Zooms to the graph area                *
5330 !*****
5335 X_lef=0
5340 X_cen=50*RATIO
5345 X_rig=100*RATIO
5350 !*****
5355 !* If desired, leave space for main titles at the top of the screen *
5360 !*****
5365 Title_offset=0
5370 Y_bot=0
5375 Y_cen=(100-Title_offset)/2
5380 Y_top=100-Title_offset
5385 !*****
5390 !* Determine the offsets of the graph area from the plotting area *
5395 !*****
5400 IF Quadrant THEN
5405     Lef_off=11.3*RATIO                ! Space for y-axis labels
5410     Rig_off=4.5*RATIO                ! Space for x-axis labels
5415     Bot_off=8.5                    ! Space for x-axis labels
5420     Top_off=8.25                   ! Space for graph's title
5425 ELSE
5430     Lef_off=16*RATIO
5435     Rig_off=7*RATIO
5440     Bot_off=12
5445     Top_off=14
5450 END IF
5455 !*****
5460 !*                Zoom to the graph area                *
5465 !*****
5470 SELECT Quadrant
5475 CASE 1
5480     VIEWPORT X_cen+Lef_off,X_rig-Rig_off,Y_cen+Bot_off,Y_top-Top_off
5485 CASE 2
5490     VIEWPORT X_lef+Lef_off,X_cen-Rig_off,Y_cen+Bot_off,Y_top-Top_off
5495 CASE 3
5500     VIEWPORT X_lef+Lef_off,X_cen-Rig_off,Y_bot+Bot_off,Y_cen-Top_off
5505 CASE 4
5510     VIEWPORT X_cen+Lef_off,X_rig-Rig_off,Y_bot+Bot_off,Y_cen-Top_off
5515 CASE ELSE
5520     VIEWPORT X_lef+Lef_off,X_rig-Rig_off,Y_bot+Bot_off,Y_top-Top_off
5525 END SELECT
5530 SUBEND
```

Write_axes_name

```
5545 SUB Write_axes_name(Xaxis_name$,Yaxis_name$,INTEGER Quadrant)
5550 !*****
5555 !*           Writes the names of the axes           *
5560 !*****
5565 LINE TYPE 1                      ! Solid line
5570 DEG                             ! All angles in degrees
5575 LONG 6
5580 PEN 4                            ! Green
5585 !*****
5590 !*   Determine the size and location of the axes labels   *
5595 !*****
5600 IF Quadrant THEN
5605     Character_size=3              ! Character size of axes labels
5610     Left_offset=-32.5             ! Horiz. offset of y-axis label
5615     Bottom_offset=-13.5          ! Vert. offset of x-axis label
5620 ELSE
5625     Character_size=3.8
5630     Left_offset=-19.5
5635     Bottom_offset=-8.5
5640 END IF
5645 !
5650 CLIP OFF                          ! Turn off soft clipping
5655 WINDOW 0,100,0,100
5660 CSIZE Character_size
5665 !
5670 LDIR 0
5675 MOVE 50,Bottom_offset
5680 LABEL Xaxis_name$                ! Label x-axis
5685 LDIR 90
5690 MOVE Left_offset,50
5695 LABEL Yaxis_name$                ! Label y-axis
5700 CLIP ON                          ! Turn on soft clipping
5705 SUBEND
```

Calc_lin_axis

```
5720 SUB Calc_lin_axis(Min_val,Max_val,Axis_min,Axis_max,Major_div,INTEGER Quadrant)
5725 !*****
5730 !* Calculate the minimum and maximum values of a linear scale axis *
5735 !*****
5740 INTEGER Exponent
5745 !*****
5750 !* Calculate the major division *
5755 !*****
5760 Data_min=MIN(Min_val,Max_val)
5765 Data_max=MAX(Min_val,Max_val)
5770 !
5775 IF Quadrant THEN
5780 Approx_division=(Data_max-Data_min)/(4/1.2)
5785 ELSE
5790 Approx_division=(Data_max-Data_min)/5
5795 END IF
5800 !
5805 Exponent=INT(LGT(Approx_division))
5810 Mantissa=Approx_division/(10.0^Exponent)
5815 !
5820 SELECT Mantissa
5825 CASE <=1.35
5830 Major_div=1
5835 CASE <=2.4
5840 Major_div=2
5845 CASE <=3.6
5850 Major_div=3
5855 CASE <=7.2
5860 Major_div=5
5865 CASE ELSE
5870 Major_div=10
5875 END SELECT
5880 !
5885 Major_div=DROUND(Major_div*10^Exponent,1)
5890 IF Min_val>Max_val THEN Major_div=-Major_div
5895 !*****
5900 !* Calculate the minimum and maximum values of the axis *
5905 !*****
5910 IF Min_val<0 THEN
5915 IF ABS(Max_val)>=1 THEN Axis_min=PROUND(Min_val,Exponent)
5920 IF ABS(Max_val)<1 THEN Axis_min=PROUND(Min_val,Exponent)
5925 ELSE
5930 IF ABS(Max_val)>=1 THEN Axis_min=PROUND(Min_val,Exponent)
5935 IF ABS(Max_val)<1 THEN Axis_min=PROUND(Min_val,Exponent)
5940 END IF
5945 !
5950 IF Max_val<0 THEN
5955 IF ABS(Max_val)>=1 THEN Axis_max=PROUND(Max_val,Exponent)
5960 IF ABS(Max_val)<1 THEN Axis_max=PROUND(Max_val,Exponent)
5965 ELSE
5970 IF ABS(Max_val)>=1 THEN Axis_max=PROUND(Max_val,Exponent)
5975 IF ABS(Max_val)<1 THEN Axis_max=PROUND(Max_val,Exponent)
5980 END IF
5985 SUBEND
```

Label_lin_xaxis

```

5995 SUB Label_lin_xaxis(X_min,X_max,Xmajor_div,Y_min,Y_max,INTEGER Xminor_tick,Quadrant)
6000 !*****
6005 !*           Labels the linear scaled x-axis           *
6010 !*****
6015 LINE TYPE 1                                ! Solid line
6020 LDIR 0                                      ! 0 degrees rotation
6025 LORG 6
6030 CLIP OFF                                ! Soft clip off
6035 !*****
6040 !* Determine the vertical position and size of the axis labels *
6045 !*****
6050 IF Quadrant THEN
6055   Y_coordinate=Y_min-.028*(Y_max-Y_min)
6060   Y_coord_1=Y_min-.1*(Y_max-Y_min)
6065   CSIZE 2.6                                ! use small characters.
6070 ELSE
6075   Y_coordinate=Y_min-.02*(Y_max-Y_min)
6080   Y_coord_1=Y_min-.07*(Y_max-Y_min)
6085   CSIZE 3.0
6090 END IF
6095 Tick_length=.018*(Y_max-Y_min)
6100 !*****
6105 !*           Draw minor ticks           *
6110 !*****
6115 IF X_min*X_max<0 THEN
6120   FOR X_coordinate=0 TO X_max STEP Xmajor_div/Xminor_tick
6125     MOVE X_coordinate,Y_min
6130     IDRAW 0,Tick_length*.5
6135   NEXT X_coordinate
6140   !
6145   FOR X_coordinate=0-Xmajor_div/Xminor_tick TO X_min STEP -Xmajor_div/Xminor_tick
6150     MOVE X_coordinate,Y_min
6155     IDRAW 0,Tick_length*.5
6160   NEXT X_coordinate
6165 ELSE
6170   FOR X_coordinate=X_min+Xmajor_div/Xminor_tick TO X_max STEP Xmajor_div/Xminor_tick
6175     MOVE X_coordinate,Y_min
6180     IDRAW 0,Tick_length*.5
6185   NEXT X_coordinate
6190 END IF
6195 !*****
6200 !*           Write the labels and draw major ticks           *
6205 !*****
6210 Max_val=MAX(ABS(X_min),ABS(X_max))
6215 Digit=INT(LGT(ABS(Max_val)))-INT(LGT(ABS(Xmajor_div)))
6220 IF Digit<4 THEN
6225   IF Quadrant=0 THEN
6230     B=ABS(X_max/(Xmajor_div/Xminor_tick) MOD Xminor_tick)
6235     IF B>1 OR B=0 THEN
6240       GOSUB Calc_axis
6245       X_coordinate=X_max
6250       MOVE X_coordinate,Y_coordinate
6255       GOSUB Write_label
6260     END IF
6265     !
6270     B=ABS(X_min/(Xmajor_div/Xminor_tick) MOD Xminor_tick)
6275     IF B>1 OR B=0 THEN
6280       GOSUB Calc_axis
6285       X_coordinate=X_min
6290       MOVE X_coordinate,Y_coordinate
6295       GOSUB Write_label
6300     END IF

```

```

6305     END IF
6310     !
6315     IF X_min*X_max<0 THEN
6320         GOSUB Calc_axis
6325         X_position=0
6330         FOR I=0 TO INT(ABS(X_max/Xmajor_div))
6335             X_coordinate=PROUND(X_position,INT(LGT(ABS(Xmajor_div))))
6340             MOVE X_coordinate,Y_coordinate
6345             GOSUB Write_label
6350             GOSUB Draw_x_tick
6355             X_position=X_position+Xmajor_div
6360         NEXT I
6365         !
6370         GOSUB Calc_axis
6375         X_position=0-Xmajor_div
6380         FOR I=0 TO INT(ABS(X_min/Xmajor_div))-1
6385             X_coordinate=PROUND(X_position,INT(LGT(ABS(Xmajor_div))))
6390             MOVE X_coordinate,Y_coordinate
6395             GOSUB Write_label
6400             GOSUB Draw_x_tick
6405             X_position=X_position-Xmajor_div
6410         NEXT I
6415     ELSE
6420         GOSUB Calc_axis
6425         IF X_min<>0 THEN
6430             IF ABS(X_max/X_min)=2 OR ABS(X_max/X_min)=.5 THEN
6435                 X_stop=X_max+Xmajor_div
6440             ELSE
6445                 X_stop=X_max
6450             END IF
6455         ELSE
6460             X_stop=X_max
6465         END IF
6470         !
6475         FOR X_coordinate=X_min TO X_stop STEP Xmajor_div
6480             MOVE X_coordinate,Y_coordinate
6485             GOSUB Write_label
6490             GOSUB Draw_x_tick
6495         NEXT X_coordinate
6500     END IF
6505 ELSE
6510     GOSUB Calc_axis
6515     FOR X_coordinate=X_min TO X_max STEP Xmajor_div
6520         MOVE X_coordinate,Y_coordinate
6525         IF X_coordinate<>X_min AND X_coordinate<>X_max THEN GOSUB Draw_x_tick
6530     NEXT X_coordinate
6535     !
6540     FOR X_coordinate=X_min TO X_max STEP (X_max-X_min)
6545         GOSUB Calc_axis
6550         MOVE X_coordinate,Y_coordinate
6555         GOSUB Write_label
6560     NEXT X_coordinate
6565     GOSUB Write_sublabel
6570 END IF
6575 !*****
6580 !*                Write exponent value                *
6585 !*****
6590 MOVE X_max,Y_coord_1
6595 LOG 6
6600 A$=VAL$(Exponent)
6605 LABEL "(xE"&A$&"")"
6610 CLIP ON
6615 SUBEXIT
6620 !
6625 Write_label:!
6630 X_coordinate=PROUND(X_coordinate,INT(LGT(ABS(Xmajor_div)))-2)

```

```

6635 IF Quadrant THEN
6640 IF Digit<4 THEN
6645 LABEL USING "K";DROUND((X_coordinate/(10^Exponent)),5)
6650 ELSE
6655 LABEL USING "K";DROUND((X_coordinate/(10^Exponent)),15)
6660 END IF
6665 ELSE
6670 IF Digit<4 THEN
6675 LABEL USING "K";DROUND((X_coordinate/(10^Exponent)),6)
6680 ELSE
6685 LABEL USING "K";DROUND((X_coordinate/(10^Exponent)),15)
6690 END IF
6695 END IF
6700 RETURN
6705 !
6710 Write_sublabel: !
6715 MOVE X_min+(X_max-X_min)/2,Y_coordinate
6720 LABEL USING "K";Xmajor_div/Xminor_tick;"/div"
6725 RETURN
6730 !
6735 Draw_x_tick: !
6740 MOVE X_coordinate,Y_min
6745 IDRAW 0,Tick_length
6750 !
6755 MOVE X_coordinate,Y_max
6760 IDRAW 0,-Tick_length
6765 RETURN
6770 !
6775 Calc_axis: !
6780 A1=MAX(ABS(X_min),ABS(X_max))
6785 !
6790 IF A1<-1 OR A1>1 THEN
6795 FOR I=1 TO 102
6800 IF (A1/(10^(3*(I-1))))<1000 THEN
6805 Exponent=3*(I-1)
6810 I=103
6815 END IF
6820 NEXT I
6825 END IF
6830 !
6835 IF A1>=-1 AND A1<=1 THEN
6840 FOR I=1 TO 102
6845 IF (A1/((10^(-3*(I-1)))))>=1 THEN
6850 Exponent=-3*(I-1)
6855 I=103
6860 END IF
6865 NEXT I
6870 END IF
6875 RETURN
6880 SUBEND

```

Label_lin_yaxis

```

6895 SUB Label_lin_yaxis(X_min,X_max,Y_min,Y_max,Ymajor_div,INTEGER Minor_ticks,Quadrant)
6900 !*****
6905 !*           Labels the linear scaled y-axis           *
6910 !*****
6915 IF Quadrant THEN           ! If a quadrant is specified,
6920     CSIZE 2.6               ! use small characters.
6925 ELSE
6930     CSIZE 3
6935 END IF
6940 !
6945 LINE TYPE 1                ! Solid line
6950 LDIR 0                     ! 0 degrees rotation
6955 LORG 8
6960 CLIP OFF                   ! Soft clip off
6965 !*****
6970 !*           Calculate the horizontal position of the axis labels       *
6975 !*****
6980 X_coordinate=X_min-.015*(X_max-X_min)
6985 X_coord_1=X_min+.005*(X_max-X_min)
6990 Tick_length=.01*(X_max-X_min)
6995 !*****
7000 !*           Draw minor ticks           *
7005 !*****
7010 IF Y_min*Y_max<0 THEN
7015     FOR Y_coordinate=0 TO Y_max STEP Ymajor_div/Minor_ticks
7020         MOVE X_min,Y_coordinate
7025         IDRAW Tick_length*.5,0
7030     NEXT Y_coordinate
7035 !
7040     FOR Y_coordinate=0-Ymajor_div/Minor_ticks TO Y_min STEP -Ymajor_div/Minor_ticks
7045         MOVE X_min,Y_coordinate
7050         IDRAW Tick_length*.5,0
7055     NEXT Y_coordinate
7060 ELSE
7065     FOR Y_coordinate=Y_min+Ymajor_div/Minor_ticks TO Y_max STEP Ymajor_div/Minor_ticks
7070         MOVE X_min,Y_coordinate
7075         IDRAW Tick_length*.5,0
7080     NEXT Y_coordinate
7085 END IF
7090 !*****
7095 !*           Write the labels and draw major ticks           *
7100 !*****
7105 Max_val=MAX(ABS(Y_min),ABS(Y_max))
7110 Digit=INT(LGT(ABS(Max_val)))-INT(LGT(ABS(Ymajor_div)))
7115 IF Digit<4 THEN
7120     IF Quadrant=0 THEN
7125         B=ABS(Y_max/(Ymajor_div/Minor_ticks) MOD Minor_ticks)
7130         IF B>1 THEN
7135             GOSUB Calc_axis
7140             Y_coordinate=Y_max
7145             MOVE X_coordinate,Y_coordinate
7150             GOSUB Write_label
7155         END IF
7160         !
7165         B=ABS(Y_min/(Ymajor_div/Minor_ticks) MOD Minor_ticks)
7170         IF B>1 THEN
7175             GOSUB Calc_axis
7180             Y_coordinate=Y_min
7185             MOVE X_coordinate,Y_coordinate
7190             GOSUB Write_label
7195         END IF
7200     END IF

```

```

7205      !
7210      IF Y_min*Y_max<0 THEN
7215          GOSUB Calc_axis
7220          Y_position=0
7225          FOR I=0 TO INT(ABS(Y_max/Ymajor_div))
7230              Y_coordinate=PROUND(Y_position,INT(LGT(ABS(Ymajor_div))))
7235              MOVE X_coordinate,Y_coordinate
7240              GOSUB Write_label
7245              GOSUB Draw_y_tick
7250              Y_position=Y_position+Ymajor_div
7255          NEXT I
7260      !
7265      GOSUB Calc_axis
7270      Y_position=0-Ymajor_div
7275      FOR I=0 TO INT(ABS(Y_min/Ymajor_div))-1
7280          Y_coordinate=PROUND(Y_position,INT(LGT(ABS(Ymajor_div))))
7285          MOVE X_coordinate,Y_coordinate
7290          GOSUB Write_label
7295          GOSUB Draw_y_tick
7300          Y_position=Y_position-Ymajor_div
7305      NEXT I
7310      ELSE
7315          GOSUB Calc_axis
7320          IF Y_min<>0 THEN
7325              IF ABS(Y_max/Y_min)=2 OR ABS(Y_max/Y_min)=.5 THEN
7330                  Y_stop=Y_max+Ymajor_div
7335              ELSE
7340                  Y_stop=Y_max
7345              END IF
7350          ELSE
7355              Y_stop=Y_max
7360          END IF
7365      !
7370      FOR Y_coordinate=Y_min TO Y_stop STEP Ymajor_div
7375          MOVE X_coordinate,Y_coordinate
7380          GOSUB Write_label
7385          GOSUB Draw_y_tick
7390      NEXT Y_coordinate
7395      END IF
7400      ELSE
7405          GOSUB Calc_axis
7410          FOR Y_coordinate=Y_min TO Y_max STEP Ymajor_div
7415              MOVE X_coordinate,Y_coordinate
7420              IF Y_coordinate<>Y_min AND Y_coordinate<>Y_max THEN GOSUB Draw_y_tick
7425          NEXT Y_coordinate
7430      !
7435          FOR Y_coordinate=Y_min TO Y_max STEP (Y_max-Y_min)
7440              GOSUB Calc_axis
7445              MOVE X_coordinate,Y_coordinate
7450              GOSUB Write_label
7455          NEXT Y_coordinate
7460          GOSUB Write_sublabel
7465      END IF
7470      !*****
7475      !*              Write exponent value              *
7480      !*****
7485      MOVE X_coord_1,Y_max+.025*(Y_max-Y_min)
7490      LONG 7
7495      A$=VAL$(Exponent)
7500      LABEL "(xE"&A$&")"
7505      CLIP ON
7510      SUBEXIT
7515      !
7520      Write_label:      !
7525      Y_coordinate=PROUND(Y_coordinate,INT(LGT(ABS(Ymajor_div)))-2)
7530      IF Quadrant THEN

```

```

7535     IF Digit<4 THEN
7540         LABEL USING "K";DROUND(Y_coordinate/(10^Exponent),5)
7545     ELSE
7550         LABEL USING "K";DROUND(Y_coordinate/(10^Exponent),15)
7555     END IF
7560 ELSE
7565     IF Digit<4 THEN
7570         LABEL USING "K";DROUND(Y_coordinate/(10^Exponent),6)
7575     ELSE
7580         LABEL USING "K";DROUND(Y_coordinate/(10^Exponent),15)
7585     END IF
7590 END IF
7595 RETURN
7600 !
7605 Write_sublabel: !
7610     MOVE X_coordinate,Y_min+(Y_max-Y_min)/2
7615     LDIR 3.1415926/2
7620     LONG 4
7625     LABEL USING "K";Ymajor_div/Minor_ticks;"/div"
7630     LDIR 0
7635     LONG 8
7640     RETURN
7645     !
7650 Draw_y_tick: !
7655     MOVE X_min,Y_coordinate
7660     IDRAW Tick_length,0
7665     !
7670     MOVE X_max,Y_coordinate
7675     IDRAW -Tick_length,0
7680     RETURN
7685     !
7690 Calc_axis: !
7695     A1=MAX(ABS(Y_min),ABS(Y_max))
7700     !
7705     IF A1<-1 OR A1>1 THEN
7710         FOR I=1 TO 102
7715             IF (A1/(10^(3*(I-1))))<1000 THEN
7720                 Exponent=3*(I-1)
7725                 I=103
7730             END IF
7735         NEXT I
7740     END IF
7745     !
7750     IF A1>=-1 AND A1<=1 THEN
7755         FOR I=1 TO 102
7760             IF (A1/((10^(-3*(I-1)))))>=1 THEN
7765                 Exponent=-3*(I-1)
7770                 I=102
7775             END IF
7780         NEXT I
7785     END IF
7790     RETURN
7795 SUBEND

```

Calc_log_axis

```
7810 SUB Calc_log_axis(Min_val,Max_val,INTEGER Axis_min,Axis_max,Major_tick,Minor_tick,Quadrant)
7815 !*****
7820 !* Calculate the minimum number of decades that will span the data, *
7825 !* and determine the tick mark spacing. *
7830 !*****
7835 IF Min_val>0 OR Max_val>0 THEN
7840     Data_min=Min_val
7845     Data_max=Max_val
7850     Axis_min=LGT(Data_min)
7855     Axis_max=LGT(Data_max)
7860     !
7865     IF Min_val<Max_val THEN
7870         IF LGT(Data_min)<Axis_min THEN Axis_min=Axis_min-1
7875         IF LGT(Data_max)>Axis_max THEN Axis_max=Axis_max+1
7880     END IF
7885     !
7890     IF Min_val>Max_val THEN
7895         IF LGT(Data_min)>Axis_min THEN Axis_min=Axis_min+1
7900         IF LGT(Data_max)<Axis_max THEN Axis_max=Axis_max-1
7905     END IF
7910 END IF
7915 !
7920 IF Min_val<0 OR Max_val<0 THEN
7925     Data_min=ABS(Min_val)
7930     Data_max=ABS(Max_val)
7935     Axis_min=-LGT(Data_min)
7940     Axis_max=-LGT(Data_max)
7945     !
7950     IF Min_val<Max_val THEN
7955         IF -LGT(Data_min)<Axis_min THEN Axis_min=Axis_min-1
7960         IF -LGT(Data_max)>Axis_max THEN Axis_max=Axis_max+1
7965     END IF
7970     !
7975     IF Min_val>Max_val THEN
7980         IF -LGT(Data_min)>Axis_min THEN Axis_min=Axis_min+1
7985         IF -LGT(Data_max)<Axis_max THEN Axis_max=Axis_max-1
7990     END IF
7995 END IF
8000 !*****
8005 !* Determine the number of major and minor tick marks *
8010 !*****
8015 IF Axis_min*Axis_max>0 THEN
8020     Draw_type=MAX(ABS(Axis_min),ABS(Axis_max))-MIN(ABS(Axis_min),ABS(Axis_max))
8025 ELSE
8030     Draw_type=MAX(Axis_min,Axis_max)-MIN(Axis_min,Axis_max)
8035 END IF
8040 !
8045 SELECT Draw_type
8050 CASE 1
8055     Major_tick=1
8060     IF Quadrant THEN
8065         Minor_tick=3
8070     ELSE
8075         Minor_tick=10
8080     END IF
8085 CASE 2
8090     Major_tick=1
8095     IF Quadrant THEN
8100         Minor_tick=1
8105     ELSE
8110         Minor_tick=3
8115     END IF
```

```

8120 CASE 3
8125     Major_tick=1
8130     IF Quadrant THEN
8135         Minor_tick=1
8140     ELSE
8145         Minor_tick=2
8150     END IF
8155 CASE 4 TO 5
8160     Major_tick=1
8165     IF Quadrant THEN
8170         Minor_tick=0
8175         IF Draw_type=4 THEN Major_tick=1
8180         IF Draw_type=5 THEN Major_tick=2
8185     ELSE
8190         Minor_tick=1
8195     END IF
8200 CASE 6 TO 9
8205     IF Quadrant THEN
8210         IF Draw_type=6 OR Draw_type=7 THEN Major_tick=3
8215         IF Draw_type=8 OR Draw_type=9 THEN Major_tick=3
8220     ELSE
8225         Major_tick=1
8230     END IF
8235     Minor_tick=0
8240 CASE 10 TO 16
8245     IF Quadrant THEN
8250         Major_tick=5
8255     ELSE
8260         Major_tick=2
8265     END IF
8270     Minor_tick=0
8275 CASE 17 TO 30
8280     IF Quadrant THEN
8285         IF Draw_type<21 THEN Major_tick=5
8290         IF Draw_type>=21 THEN Major_tick=10
8295     ELSE
8300         Major_tick=5
8305     END IF
8310     Minor_tick=0
8315 CASE 31 TO 40
8320     Major_tick=10
8325     Minor_tick=0
8330 CASE 41 TO 100
8335     IF Quadrant THEN
8340         IF Draw_type<50 THEN Major_tick=15
8345         IF Draw_type>=50 THEN Major_tick=25
8350     ELSE
8355         Major_tick=20
8360     END IF
8365     Minor_tick=0
8370 CASE 101 TO 200
8375     IF Quadrant THEN
8380         IF Draw_type<=150 THEN Major_tick=50
8385         IF Draw_type>150 THEN Major_tick=100
8390     ELSE
8395         Major_tick=50
8400     END IF
8405     Minor_tick=0
8410 CASE ELSE
8415     Major_tick=100
8420     Minor_tick=0
8425 END SELECT
8430 SUBEND

```

Label_log_yaxis

```

8445 SUB Label_log_yaxis(REAL X_min,X_max,INTEGER Y_min,Y_max,Major_tick,Minor_tick,Quadrant,Y_polarity,
Y_val_exchange,Grid_pattern)
8450 !*****
8455 !*           Labels the log scaled y-axis           *
8460 !*****
8465 INTEGER Exponent
8470 INTEGER Mantissa
8475 !*****
8480 !*   Determine the character size and minor tick mark length   *
8485 !*****
8490 IF Quadrant THEN
8495   CSIZE 2.6
8500   Tick_length=.0155*(X_max-X_min)
8505 ELSE
8510   CSIZE 3
8515   Tick_length=.0062*(X_max-X_min)
8520 END IF
8525 LINE TYPE 1           ! Solid line
8530 LDIR 0                 ! 0 degrees rotation
8535 LONG 8
8540 CLIP OFF               ! Soft clip off
8545 !*****
8550 !*   Calculate the horizontal position of the axis labels   *
8555 !*****
8560 X_coordinate=X_min-.01*(X_max-X_min)
8565 !*****
8570 !*           Label the major tick marks           *
8575 !*****
8580 IF ABS(ABS(Y_min)-ABS(Y_max))<=40 THEN
8585   IF Y_min<Y_max THEN Y_step=1
8590   IF Y_min>Y_max THEN Y_step=-1
8595 END IF
8600 !
8605 IF Y_val_exchange=1 THEN Major_tick=-Major_tick
8610 FOR Exponent=Y_min TO Y_max STEP Y_step
8615   IF Exponent<>Y_min AND Exponent<>Y_max THEN GOSUB Draw_y_tick
8620   GOSUB Draw_y_tick
8625 NEXT Exponent
8630 !
8635 FOR Exponent=Y_min TO Y_max STEP Major_tick
8640   IF ABS(Exponent)>=0 AND ABS(Exponent)<=9 THEN Using$="MDESZ"
8645   IF ABS(Exponent)>=10 AND ABS(Exponent)<=99 THEN Using$="MDESZZ"
8650   IF ABS(Exponent)>=100 AND ABS(Exponent)<=999 THEN Using$="MDESZZZ"
8655   !
8660   IF Exponent<>Y_min AND Exponent<>Y_max THEN GOSUB Draw_y_tick
8665   MOVE X_coordinate,Exponent
8670   IF Y_polarity=0 THEN
8675     LABEL USING Using$;PROUND(10^Exponent,Exponent)
8680   ELSE
8685     LABEL USING Using$;-(1/PROUND(10^Exponent,Exponent))
8690   END IF
8695 NEXT Exponent
8700 !*****
8705 !*           Label the minor tick marks           *
8710 !*****
8715 Y_start=MIN(Y_min,Y_max)
8720 Y_stop=MAX(Y_min,Y_max)
8725 IF Y_polarity=0 THEN Y_stop=Y_stop-1
8730 IF Y_polarity=1 THEN Y_start=Y_start+1
8735 !
8740 IF Minor_tick<>0 THEN
8745   FOR Exponent=Y_start TO Y_stop

```

```

8750     FOR Mantissa=1 TO 10
8755     IF Y_polarity=0 THEN Y_coordinate=PROUND(Mantissa*(10^Exponent),Exponent)
8760     IF Y_polarity=1 THEN Y_coordinate=PROUND(Mantissa*(10^(-Exponent)),-Exponent)
8765     IF Grid_pattern=0 THEN GOSUB Write_label
8770     NEXT Mantissa
8775     NEXT Exponent
8780     END IF
8785     !
8790     CLIP ON
8795     SUBEXIT
8800     !
8805 Write_label: !
8810     IF Y_polarity=0 THEN
8815         MOVE X_min,LGT(Y_coordinate)
8820         IDRAW Tick_length*.8,0
8825         !
8830         MOVE X_max,LGT(Y_coordinate)
8835         IDRAW -Tick_length*.8,0
8840     END IF
8845     !
8850     IF Y_polarity=1 THEN
8855         MOVE X_min,-LGT(Y_coordinate)
8860         IDRAW Tick_length*.8,0
8865         !
8870         MOVE X_max,-LGT(Y_coordinate)
8875         IDRAW -Tick_length*.8,0
8880     END IF
8885     RETURN
8890     !
8895 Draw_y_tick: !
8900     MOVE X_min,Exponent
8905     IDRAW Tick_length*1.5,0
8910     MOVE X_max,Exponent
8915     IDRAW -Tick_length*1.5,0
8920     RETURN
8925 SUBEND

```

Label_log_xaxis

```
8940 SUB Label_log_xaxis(INTEGER X_min,X_max,REAL Y_min,Y_max,INTEGER Major_tick,Minor_tick,Quadrant,
X_polarity,X_val_exchange,Grid_pattern)
8945 !*****
8950 !*          Labels the log scaled x-axis          *
8955 !*****
8960 INTEGER Exponent
8965 INTEGER Mantissa
8970 !*****
8975 !*    Determine the character size and minor tick mark length    *
8980 !*****
8985 IF Quadrant THEN
8990     CSIZE 2.6
8995     Tick_length=.0155*(Y_max-Y_min)
9000 ELSE
9005     CSIZE 3
9010     Tick_length=.0085*(Y_max-Y_min)
9015 END IF
9020 LINE TYPE 1                ! Solid line
9025 LDIR 0                    ! 0 degrees rotation
9030 LORG 6
9035 CLIP OFF                  ! Soft clip off
9040 !*****
9045 !*    Calculate the horizontal position of the axis labels    *
9050 !*****
9055 Y_coordinate=Y_min-.015*(Y_max-Y_min)
9060 !*****
9065 !*          Label the major tick marks          *
9070 !*****
9075 IF ABS(ABS(X_min)-ABS(X_max))<=40 THEN
9080     IF X_min<X_max THEN X_step=1
9085     IF X_min>X_max THEN X_step=-1
9090 END IF
9095 !
9100 IF X_val_exchange=1 THEN Major_tick=-Major_tick
9105 FOR Exponent=X_min TO X_max STEP X_step
9110     IF Exponent<>X_min AND Exponent<>X_max THEN GOSUB Draw_x_tick
9115 NEXT Exponent
9120 !
9125 FOR Exponent=X_min TO X_max STEP Major_tick
9130     IF ABS(Exponent)>=0 AND ABS(Exponent)<=9 THEN Using$="MDESZ"
9135     IF ABS(Exponent)>=10 AND ABS(Exponent)<=99 THEN Using$="MDESZZ"
9140     IF ABS(Exponent)>=100 AND ABS(Exponent)<=999 THEN Using$="MDESZZZ"
9145     !
9150     IF Exponent<>X_min AND Exponent<>X_max THEN GOSUB Draw_x_tick
9155     MOVE Exponent,Y_coordinate
9160     IF X_polarity=0 THEN
9165         LABEL USING Using$;PROUND(10^Exponent,Exponent)
9170     ELSE
9175         LABEL USING Using$;-(1/PROUND(10^Exponent,Exponent))
9180     END IF
9185 NEXT Exponent
9190 !*****
9195 !*          Label the minor tick marks          *
9200 !*****
9205 X_start=MIN(X_min,X_max)
9210 X_stop=MAX(X_min,X_max)
9215 IF X_polarity=0 THEN X_stop=X_stop-1
9220 IF X_polarity=1 THEN X_start=X_start+1
9225 !
9230 IF Minor_tick<>0 THEN
9235     FOR Exponent=X_start TO X_stop
9240         FOR Mantissa=2 TO 9
```

```

9245         IF X_polarity=0 THEN X_coordinate=PROUND(Mantissa*(10^Exponent),Exponent)
9250         IF X_polarity=1 THEN X_coordinate=PROUND(Mantissa*(10^(-Exponent)),-Exponent)
9255         IF Grid_pattern=0 THEN GOSUB Write_label
9260     NEXT Mantissa
9265     NEXT Exponent
9270 END IF
9275 !
9280 CLIP ON
9285 SUBEXIT
9290 !
9295 Write_label: !
9300     IF X_polarity=0 THEN
9305         MOVE LGT(X_coordinate),Y_min
9310         IDRAW 0,Tick_length
9315         !
9320         MOVE LGT(X_coordinate),Y_max
9325         IDRAW 0,-Tick_length
9330     END IF
9335     !
9340     IF X_polarity=1 THEN
9345         MOVE -LGT(X_coordinate),Y_min
9350         IDRAW 0,Tick_length
9355         !
9360         MOVE -LGT(X_coordinate),Y_max
9365         IDRAW 0,-Tick_length
9370     END IF
9375     RETURN
9380     !
9385 Draw_x_tick: !
9390     MOVE Exponent,Y_min
9395     IDRAW 0,Tick_length*1.7
9400     MOVE Exponent,Y_max
9405     IDRAW 0,-Tick_length*1.7
9410     RETURN
9415 SUBEND

```

PARA_4142 File

FNHfe

```
30 DEF FNHfe(INTEGER Channel(*),REAL Vc,Ic,Vbstart,Vbstop,OPTIONAL Vbspeed,Integtime,Hfe_val)
35 OPTION BASE 1
40 INTEGER Ch1,Ch2,Ch3,St
45 !-----
50 Ch_num=SIZE(Channel,1)
55 Init_hp4142
60 Ch_sw_off
65 Vbrate=500 ! RAMP SPEED DEFAULT
70 Integ=5.0E-2 ! INTEG TIME DEFAULT
75 Hfe=50 ! Hfe DFAULT
80 Htime=1 ! HOLD TIME DEFAULT
85 Dtime=1.5 ! DELAY TIME DEFAULT
90 !
95 IF NPAR>5 THEN
100 Vbrate=Vbspeed
105 IF NPAR=7 THEN Integ=Integtime
110 IF NPAR=8 THEN Hfe=Hfe_val
115 END IF
120 !
125 IF Ch_num>2 THEN
130 Ch3=Channel(3)
135 SELECT Channel(3)
140 CASE 1 TO 8
145 Ch_sw_on(Ch3) ! SET SMU 0(V)
150 Force_v(Ch3,0,0,1.E-1) ! HPSMU allows 1(A) compliance
155 CASE ELSE
160 IF Channel(3)<>99 THEN
165 DISP "FNHfe: Illegal channel number"
170 BEEP
175 STOP
180 END IF
185 END SELECT
190 END IF
195 !-----
200 Ch1=Channel(1)
205 Ch2=Channel(2)
210 Ch_sw_on(Ch1,Ch2)
215 Set_amonitor(Ch1,1,Vc,Ic,Ic*1.1)
220 Set_asource(Ch2,Vbstart,Vbstop,Vbrate,Htime,Dtime,(Ic*1.1)/Hfe)
225 Measure_asearch(1,4,Integ,Ibmeas,Icmeas,St)
230 Ch_sw_off
235 !-----
240 IF St=0 THEN
245 Hfe_meas=Icmeas/Ibmeas
250 ELSE
255 Hfe_meas=-9999999.99999 ! OUT OF MEASUREMENT
260 END IF
265 !-----
270 RETURN Hfe_meas
275 FNEED
<\figure>
```

<s2>FNBvcbo

<figure text nonumber smaller>

<figuretext>

```
290 DEF FNBvcbo(INTEGER Channel(*),REAL Ic,OPTIONAL Vlim,Time)
295 OPTION BASE 1
300 INTEGER N,Ch1,Ch2,Ch3,St
```

```

305  N=NPARR
310  !-----
315  Init_hp4142
320  Ch_num=SIZE(Channel,1)
325  Vlimit=20                      ! VOLTAGE LIMIT DEFAULT
330  Twait=.4                      ! WAIT TIME DEFAULT
335  IF N>2 THEN
340      IF Vlim<>0 THEN Vlimit=ABS(Vlim)
345      IF N>3 THEN Twait=Time    ! SET WAIT TIME
350  END IF
355  !-----
360  IF Ch_num>1 THEN
365      Ch2=Channel(2)
370      SELECT Channel(2)
375      CASE 1 TO 8
380          Comp=1.E-1              ! SET HPSMU COMPLIANCE
385          Module_type((Ch2),Type$)
390          IF Type$="HP41423" THEN Comp=1.E-2    ! SET HVU COMPLIANCE
395          IF Type$="HP41423" AND Ic>=0 THEN
400              Set_pol(Ch2,1)
405          ELSE
410              Ch_sw_on(Ch2)
415          END IF
420          Force_v(Ch2,0,0,Comp)
425      CASE ELSE
430          IF Channel(2)<>99 THEN
435              DISP "FWBvcbo: Illegal channel number"
440              BEEP
445              STOP
450          END IF
455      END SELECT
460      !
465      IF Ch_num=3 THEN
470          Ch3=Channel(3)
475          Ch_sw_off(Ch3)
480      END IF
485  END IF
490  !-----
495  WAIT .01
500  Ch1=Channel(1)
505  Module_type((Ch1),Type$)
510  IF Type$="HP41423" AND Ic<0 THEN
515      Set_pol(Ch1,1)
520  ELSE
525      Ch_sw_on(Ch1)
530  END IF
535  Force_i(Ch1,Ic,0,Vlimit)      ! SET CURRENT
540  WAIT Twait
545  Measure_v(Ch1,Vmeas,0,St)    ! MEASURE Bvcbo
550  Ch_sw_off
555  WAIT .01
560  RETURN Vmeas
565  FWEEND

```

FNBvces

```

580 DEF FNBvces(INTEGER Channel(*),REAL Ic,OPTIONAL Vlim,Time)
585   OPTION BASE 1
590   INTEGER N,Ch1,Ch2
595   !-----
600   Init_hp4142
605   Ch_num=SIZE(Channel,1)
610   N=NPAR
615   Vlimit=20                      ! Vlim DEFAULT
620   Twait=.4                      ! WAIT TIME DEFAULT
625   IF N>2 THEN
630     IF Vlim<>0 THEN Vlimit=ABS(Vlim)
635     IF N>3 THEN Twait=Time
640   END IF
645   !-----
650   IF Ch_num>1 THEN
655     Ch2=Channel(2)
660     SELECT Channel(2)
665     CASE 1 TO 8
670       Comp=1.E-1                  ! SET HPSMU COMPLIANCE
675       Module_type((Ch2),Type$)
680       IF Type$="HP41423" THEN Comp=1.E-2 ! SET HVU COMPLIANCE
685       IF Type$="HP41423" AND Ic>=0 THEN
690         Set_pol(Ch2,1)
695       ELSE
700         Ch_sw_on(Ch2)
705       END IF
710       Force_v(Ch2,0,0,Comp)      ! HPSMU allows 1(A) compliance
715     CASE ELSE
720       IF Channel(2)<>99 THEN
725         DISP "FNBvces: Illegal channel number"
730         BEEP
735         STOP
740       END IF
745     END SELECT
750   END IF
755   !-----
760   WAIT .01
765   Ch1=Channel(1)
770   Module_type((Ch1),Type$)
775   IF Type$="HP41423" AND Ic<0 THEN
780     Set_pol(Ch1,1)
785   ELSE
790     Ch_sw_on(Ch1)
795   END IF
800   Force_i(Ch1,Ic,0,Vlimit)      ! SET COLLECTOR CURRENT
805   WAIT Twait
810   Measure_v(Ch1,Vmeas,0)       ! MEASURE Bvces
815   Ch_sw_off
820   WAIT .01
825   RETURN Vmeas
830 FNEED

```

FNIds

```

845 DEF FNIds(INTEGER Channel(*),REAL Vds,Vgs,OPTIONAL Idslim,Vsubs)
850   OPTION BASE 1
855   INTEGER N,St,Ch1,Ch2,Ch3,Ch4
860   !-----
865   Ch_num=SIZE(Channel,1)
870   Init_hp4142
875   N=MPAR
880   Vsubs1=0                                ! VSUBS DEFAULT VALUE
885   Ilimit=1.E-6                            ! CURRENT LIMIT SMU DEFAULT VALUE
890   IF N>3 THEN
895       Ilimit=ABS(Idslim)
900       IF N=5 THEN Vsubs1=Vsubs
905   END IF
910   Ch1=Channel(1)
915   Ch2=Channel(2)
920   Module_type((Ch1),Type$)
925   IF Type$="HP41423" AND Vds<0 THEN
930       Set_pol(Ch1,1)
935   ELSE
940       Ch_sw_on(Ch1)
945   END IF
950   Module_type((Ch2),Type$)
955   IF Type$="HP41423" AND Vgs<0 THEN
960       Set_pol(Ch2,1)
965   ELSE
970       Ch_sw_on(Ch2)
975   END IF
980   IF Ch_num=4 AND MPAR=5 THEN
985       Ch4=Channel(4)
990       Module_type((Ch4),Type$)
995       IF Type$="HP41423" AND Vsubs1<0 THEN
1000           Set_pol(Ch4,1)
1005       ELSE
1010           Ch_sw_on(Ch4)
1015       END IF
1020   END IF
1025   !-----
1030   IF Ch_num>2 THEN
1035       Ch3=Channel(3)
1040       SELECT Channel(3)
1045       CASE 1 TO 8
1050           Module_type((Ch3),Type$)
1055           IF Type$="HP41423" THEN
1060               Comp=1.E-2
1065           ELSE
1070               Comp=1.E-1
1075           END IF
1080           IF Type$="HP41423" AND Vds>=0 THEN
1085               Set_pol(Ch3,1)
1090           ELSE
1095               Ch_sw_on(Ch3)
1100           END IF
1105           Ch_sw_on(Ch3)
1110           Force_v(Ch3,0,0,Comp)
1115       CASE ELSE
1120           IF Channel(3)<>99 THEN
1125               DISP "FNIds: Illegal channel number"
1130               BEEP
1135               STOP
1140           END IF
1145       END SELECT
1150   !

```

```

1155     IF Ch_num=4 AND NPAR=5 THEN
1160         Ch4=Channel(4)
1165         Module_type((Ch4),Type$)
1170         IF Type$="HP41423" THEN
1175             Comp=1.E-2
1180         ELSE
1185             Comp=1.E-1
1190         END IF
1195         Force_v(Ch4,Vsubs1,0,Comp)      ! HPSMU allows 1(A) compliance
1200     END IF
1205 END IF
1210 !-----
1215 Force_v(Ch1,Vds,0,Ilimit)              ! SET DRAIN-SOURCE VOLTAGE
1220 Force_v(Ch2,Vgs,0,1.0E-5)             ! SET GATE-SOURCE VOLTAGE
1225 Measure_i(Ch1,Ids,0,St)               ! MEASURE DRAIN-SOURCE CURRENT
1230 !
1235 IF St>0 THEN Ids=-9999999.99999
1240 Ch_sw_off
1245 RETURN Ids
1250 F$END

```

FNVth1

```

1265 DEF FNVth1(INTEGER Channel(*),REAL Ids,OPTIONAL Vlim,Vsubs)
1270   OPTION BASE 1
1275   INTEGER N,St,Ch1,Ch2,Ch3
1280   !-----
1285   Ch_num=SIZE(Channel,1)
1290   Init_hp4142
1295   N=NPAR
1300   Iforce=ABS(Ids)
1305   Vlimit=20                      ! VLIM DEFAULT
1310   Vsubs1=0                      ! VSUBS DEFAULT
1315   IF N>2 THEN
1320     IF Vlim<>0 THEN Vlimit=ABS(Vlim)
1325     IF N=4 THEN Vsubs1=Vsubs
1330   END IF
1335   !
1340   Twait=.01                      ! WAIT TIME SELECTION
1345   IF Iforce<=2.E-5 THEN
1350     Twait=.1
1355     IF Iforce<=3.E-8 THEN
1360       Twait=1
1365       IF Iforce<=5.E-9 THEN Twait=2
1370     END IF                      ! IF Iforce<=3E-8
1375   END IF                      ! IF Iforce<=2E-5
1380   !-----
1385   IF Ch_num>1 THEN
1390     Ch2=Channel(2)
1395     SELECT Channel(2)
1400     CASE 1 TO 8
1405       Ch_sw_on(Ch2)              ! SET SMU 0(V)
1410       Force_v(Ch2,0,0,1.E-1)    ! HPSMU allows 1(A) compliance
1415     CASE ELSE
1420       IF Channel(2)<>99 THEN
1425         DISP "FNVth1: Illegal channel number"
1430         BEEP
1435         STOP
1440       END IF
1445     END SELECT
1450     !
1455     IF Ch_num=3 AND NPAR=4 THEN
1460       Ch3=Channel(3)
1465       Ch_sw_on(Ch3)              ! SET SMU 0(V)
1470       Force_v(Ch3,Vsubs1,0,1.E-1) ! HPSMU allows 1(A) compliance
1475     END IF
1480   END IF
1485   !-----
1490   Ch1=Channel(1)
1495   Ch_sw_on(Ch1)
1500   Force_i(Ch1,Ids,0,Vlimit)      ! SET DRAIN-SOURCE CURRENT
1505   WAIT Twait
1510   Measure_v(Ch1,Vgs,0,St)        ! MEASURE GATE-SOURCE VOLTAGE
1515   Vth=Vgs
1520   IF St THEN Vth=-9999999.99999
1525   Ch_sw_off
1530   RETURN Vth
1535 FVEND

```

FNVth2

```

1550 DEF FNVth2(INTEGER Channel(*),REAL Ids1,Ids2,OPTIONAL Vlim,Vsubs)
1555 OPTION BASE 1
1560 INTEGER N,I,St1,St2,Ch1,Ch2,Ch3
1565 REAL Twait(1:2)
1570 !-----
1575 Init_hp4142
1580 Ch_num=SIZE(Channel,1)
1585 N=MPAR
1590 Vlimit=20 ! Vlim DEFAULT
1595 Vsubs1=0 ! Vsubs DEFAULT
1600 IF N>3 THEN
1605 IF Vlim<>0 THEN Vlimit=ABS(Vlim)
1610 IF N=5 THEN Vsubs1=Vsubs
1615 END IF
1620 Iforce=ABS(Ids1)
1625 FOR I=1 TO 2
1630 Twait(I)=.01 ! WAIT TIME SELECTION
1635 IF Iforce<=2.E-5 THEN
1640 Twait(I)=.1
1645 IF Iforce<=3.E-8 THEN
1650 Twait(I)=1
1655 IF Iforce<=5.E-9 THEN Twait(I)=2
1660 END IF ! IF Iforce<=3E-8
1665 END IF ! IF Iforce<=2E-5
1670 Iforce=ABS(Ids2)
1675 NEXT I
1680 !-----
1685 IF Ch_num>1 THEN
1690 Ch2=Channel(2)
1695 SELECT Channel(2)
1700 CASE 1 TO 8
1705 Ch_sw_on(Ch2) ! SET SMU O(V)
1710 Force_v(Ch2,0,0,1.E-1) ! HPSMU allows 1(A) compliance
1715 CASE ELSE
1720 IF Channel(2)<>99 THEN
1725 DISP "FNVth2: Illegal channel number"
1730 BEEP
1735 STOP
1740 END IF
1745 END SELECT
1750 !
1755 IF Ch_num=3 AND MPAR=5 THEN
1760 Ch3=Channel(3)
1765 Ch_sw_on(Ch3) ! SET SMU O(V)
1770 Force_v(Ch3,Vsubs1,0,1.E-1) ! HPSMU allows 1(A) compliance
1775 END IF
1780 END IF
1785 !-----
1790 Ch1=Channel(1)
1795 Ch_sw_on(Ch1)
1800 Force_i(Ch1,Ids1,0,Vlimit) ! SET Ids1
1805 WAIT Twait(1)
1810 Measure_v(Ch1,Vgs1,0,St1) ! MEASURE Vgs1
1815 Force_i(Ch1,Ids2,0,Vlimit) ! SET Ids2
1820 WAIT Twait(2)
1825 Measure_v(Ch1,Vgs2,0,St2) ! MEASURE Vgs2
1830 IF Ids2-Ids1=0 THEN Ids1=Ids2*.99
1835 Vth=(Vgs1*ABS(Ids2)-Vgs2*ABS(Ids1)+(Vgs1-Vgs2)*SQR(ABS(Ids2)*ABS(Ids1)))/(ABS(Ids2)-ABS(Ids1))
! CALCULATE Vth
1840 IF St1 OR St2 THEN
1845 Vth=-9999999.99999
1850 ELSE

```

```
1855     Vth=PROUND(Vth,-3)
1860   END IF
1865   Ch_sw_off
1870   RETURN Vth
1875 FWEEND
```

FNVth3

```

1890 DEF FNVth3(INTEGER Channel(*),REAL Ids,Vds,Vgstart,Vgstop,Gate_i_comp,OPTIONAL Vgspeed,Integtime,
Vsubs)
1895   OPTION BASE 1
1900   INTEGER St,Ch1,Ch2,Ch3,Ch4
1905   !-----
1910   Init_hp4142
1915   Ch_num=SIZE(Channel,1)
1920   Vgrate=500                                ! RAMP SPEED DEFAULT
1925   Integ=5.E-3                               ! INTEG TIME DEFAULT
1930   Htime=1                                   ! HOLD TIME DEFAULT
1935   Dtime=1.5                               ! DELAY TIME DEFAULT
1940   !
1945   IF NPAR>6 THEN
1950       Vgrate=Vgspeed
1955       IF NPAR>=8 THEN Integ=Integtime
1960       IF NPAR=9 THEN Vsubs1=Vsubs
1965   END IF
1970   !
1975   IF Ch_num>2 THEN
1980       Ch3=Channel(3)
1985       SELECT Channel(3)
1990       CASE 1 TO 8
1995           Ch_sw_on(Ch3)                    ! SET SMU 0(V)
2000           Force_v(Ch3,0,0,1.E-1)          ! HPSMU allows 1(A) compliance
2005       CASE ELSE
2010           IF Channel(3)<>99 THEN
2015               DISP "FNVth3: Illegal channel number"
2020               BEEP
2025               STOP
2030           END IF
2035       END SELECT
2040       !
2045       IF Ch_num=4 AND NPAR=9 THEN
2050           Ch4=Channel(4)
2055           Ch_sw_on(Ch4)                    ! SET SMU 0(V)
2060           Force_v(Ch4,Vsubs1,0,1.E-1)      ! HPSMU allows 1(A) compliance
2065       END IF
2070   END IF
2075   !-----
2080   Ch1=Channel(1)
2085   Ch2=Channel(2)
2090   Ch_sw_on(Ch1,Ch2)
2095   Set_amonitor(Ch1,1,Vds,Ids,Ids*1.1)
2100   Set_asource(Ch2,Vgstart,Vgstop,Vgrate,Htime,Dtime,Gate_i_comp)
2105   Measure_asearch(1,3,Integ,Vth,Id,St)
2110   Ch_sw_off
2115   IF St THEN Vth=-9999999.99999           ! OUT OF MEASUREMENT
2120   RETURN Vth
2125 FVEND

```

FNR_measure

```

2140 DEF FNR_measure(INTEGER High_i,Low_i,High_v,Low_v,Meas_mode,REAL Itest,OPTIONAL Vlim)
2145   INTEGER St1,St2
2150   !*****
2155   !*                SMU two terminals measurement                *
2160   !*****
2165   Init_hp4142
2170   Ch_sw_off
2175   IF Meas_mode=1 THEN
2180     Iforce=ABS(Itest)                ! SOURCE CURRENT
2185     Module_type((Low_i),Type$)
2190     IF (Iforce>=1.E-9 AND Type$<>"HP41423") OR (Iforce>=1.E-8 AND Type$="HP41423") THEN
2195       IF Type$="HP41423" THEN
2200         Twait=.1
2205       ELSE
2210         Twait=.01                    ! WAIT TIME SELECTION
2215       END IF
2220       IF Iforce<=2.E-5 THEN          ! 2E-5>=|Itest|>3E-8
2225         Twait=.4
2230       IF Iforce<=3.E-8 THEN          ! 3E-8>=|Itest|>5E-9
2235         Twait=5
2240       IF Iforce<5.E-9 THEN Twait=10  ! 5E-9>=|Itest|>=1E-9
2245       END IF
2250     END IF
2255     Vlimit=20
2260     IF NPAR=7 THEN Vlimit=Vlim       ! VOLTAGE LIMIT
2265     !
2270     Module_type((High_i),Type$)
2275     IF Type$="HP41423" AND Itest>=0 THEN
2280       Set_pol(High_i,1)
2285     ELSE
2290       Ch_sw_on(High_i)
2295     END IF
2300     IF Type$="HP41423" THEN
2305       Force_range=100
2310       Comp=1.E-2
2315     ELSE
2320       Force_range=20
2325       Comp=1.E-1
2330     END IF
2335     Force_v(High_i,0,Force_range,Comp)    ! SET SMU OV
2340     Module_type((Low_i),Type$)
2345     IF Type$="HP41423" AND Itest<0 THEN
2350       Set_pol(Low_i,1)
2355     ELSE
2360       Ch_sw_on(Low_i)
2365     END IF
2370     Force_i(Low_i,Itest,0,Vlimit)         ! SET CURRENT
2375     WAIT Twait
2380     Measure_v(Low_i,Vmeas,0)              ! MEASURE VOLTAGE
2385     Measure_i(High_i,Imeas,0,St1)         ! MEASURE CURRENT
2390     IF ABS(Imeas)>=9.E-10 AND ABS(Vmeas)>=1.E-2 AND St1<>2 THEN
2395       R=DROUND(-Vmeas/Imeas,5)           ! CALCULATE R
2400     ELSE
2405       R=-9999999.99999
2410     END IF
2415     Ch_sw_off
2420   ELSE
2425     R=-9999999.99999                    ! OUT OF MEASUREMENT
2430   END IF                                ! IF ABS(Iforce)>=1.E-9
2435   RETURN R
2440 END IF
2445   !*****

```

```

2450      !*                SMU four terminals measurement                *
2455      !*****
2460      IF Meas_mode=2 THEN
2465          Iforce=ABS(Itest)                ! SOURCE CURRENT
2470          IF Iforce>=1.E-6 THEN
2475              Vlimit=20                    ! V LIM DEFAULT
2480              IF WPAR=7 THEN Vlimit=Vlim    ! VOLTAGE LIMIT
2485              !
2490              Module_type((High_i),Type$)
2495              IF Type$="HP41423" THEN
2500                  Twait=.1
2505              ELSE
2510                  Twait=.01
2515              END IF
2520              IF Iforce<=2.E-5 THEN Twait=.4    ! WAIT TIME SELECTION
2525              Module_type((Low_i),Type$)
2530              IF Type$="HP41423" AND Itest>=0 THEN
2535                  Set_pol(Low_i,1)
2540              ELSE
2545                  Ch_sw_on(Low_i)
2550              END IF
2555              IF Type$="HP41423" THEN
2560                  Force_range=100
2565                  Comp=1.E-2
2570              ELSE
2575                  Force_range=20
2580                  Comp=1.E-1
2585              END IF
2590              Force_v(Low_i,0,Force_range,Comp)    ! SET I MEASUREMENT MODE
2595              Ch_sw_on(Low_v,High_v)
2600              Module_type((High_i),Type$)
2605              IF Type$="HP41423" AND Itest<0 THEN
2610                  Set_pol(High_i,1)
2615              ELSE
2620                  Ch_sw_on(High_i)
2625              END IF
2630              Force_i(Low_v,1.E-10,0,20)          ! SET V MEASUREMENT MODE
2635              Force_i(High_v,1.E-10,0,Vlimit)      ! SET V MEASUREMENT MODE
2640              Force_i(High_i,Itest,0,Vlimit)        ! SET CURRENT
2645              WAIT Twait
2650              Measure_i(Low_i,I meas,0,St1)         ! MEASURE CURRENT
2655              Measure_v(Low_v,Vm2,0)               ! MEASURE V2
2660              Measure_v(High_v,Vm1,0)              ! MEASURE V1
2665              IF ABS(I meas)>=9.E-7 AND ABS(Vm1)>=1.E-2 AND St1<>2 THEN
2670                  R=DROUND(-(Vm1-Vm2)/I meas,5)    ! CALCULATE R
2675              ELSE
2680                  R=-9999999.99999
2685              END IF
2690              Ch_sw_off
2695              ELSE
2700                  R=-9999999.99999                ! OUT OF MEASUREMENT
2705              END IF                                ! IF Iforce>=1.E-6
2710              RETURN R
2715      END IF
2720      !*****
2725      !*                VM differential measurement                *
2730      !*****
2735      IF Meas_mode=3 THEN
2740          Iforce=ABS(Itest)                ! SOURCE CURRENT
2745          Module_type((High_i),Type$)
2750          IF (Iforce>=1.E-9 AND Type$<>"HP41423") OR (Iforce>=1.E-8 AND Type$="HP41423") THEN
2755              IF Type$="HP41423" THEN
2760                  Twait=.1
2765              ELSE
2770                  Twait=.01
2775              END IF

```

```

2780     IF Iforce<=2.E-5 THEN
2785         Twait=.4
2790     IF Iforce<=3.E-8 THEN
2795         Twait=5
2800     IF Iforce<5.E-9 THEN Twait=10
2805     END IF
2810 END IF
2815 Vlimit=20                                ! VLIM DEFAULT
2820 !
2825 IF NPAR=7 THEN Vlimit=Vlim                ! VOLTAGE LIMIT
2830 Module_type((Low_i),Type$)
2835 IF Type$="HP41423" AND Itest>=0 THEN
2840     Set_pol(Low_i,1)
2845 ELSE
2850     Ch_sw_on(Low_i)
2855 END IF
2860 IF Type$="HP41423" THEN
2865     Force_range=100
2870     Comp=1.E-2
2875 ELSE
2880     Force_range=20
2885     Comp=1.E-1
2890 END IF
2895 Force_v(Low_i,0,Force_range,Comp)          ! SET I MEASUREMENT MODE
2900 Module_type((High_i),Type$)
2905 IF Type$="HP41423" AND Itest<0 THEN
2910     Set_pol(High_i,1)
2915 ELSE
2920     Ch_sw_on(High_i)
2925 END IF
2930 Force_i(High_i,Itest,0,Vlimit)              ! SET CURRENT
2935 WAIT Twait
2940 Measure_i(Low_i,Imeas,0,St1)                ! MEASURE CURRENT
2945 Set_vm(High_v,2)                            ! SET VM DIFF MODE
2950 Measure_v(High_v,Vmeas,0,St2)              ! MEASURE VOLTAGE
2955 Set_vm(High_v,1)                            ! SET VM NORMAL MODE
2960 IF ABS(Imeas)>=9.E-7 AND ABS(Vmeas)>=1.E-2 AND St1<>2 THEN
2965     R=DROUND(Vmeas/Imeas,5)                  ! CALCULATE R
2970 ELSE
2975     R=-99999999.99999
2980 END IF
2985 Ch_sw_off
2990 ELSE
2995     R=-99999999.99999                        ! OUT OF MEASUREMENT
3000 END IF
3005 RETURN R
3010 END IF
3015 FEND

```