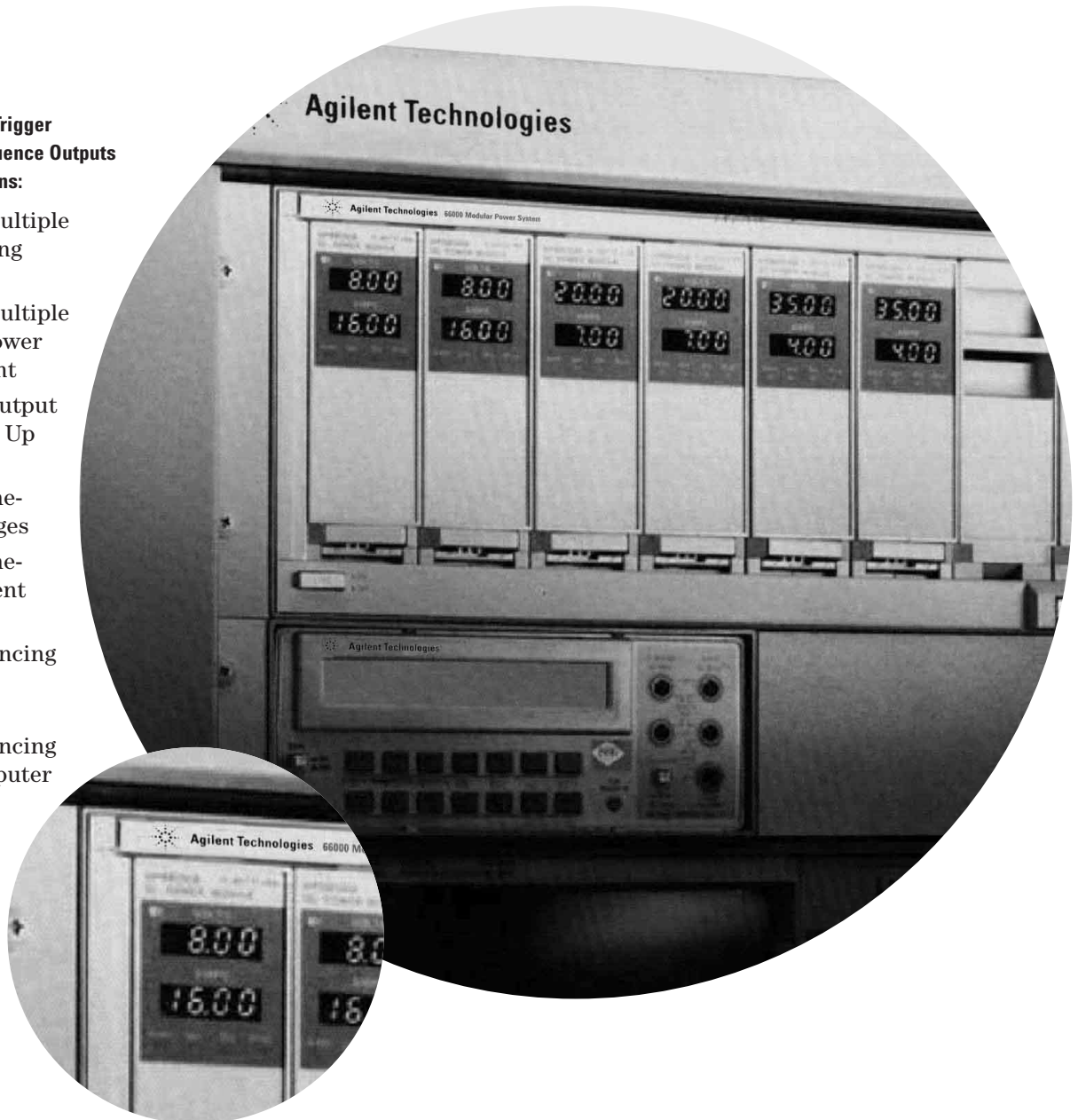


# Agilent 66000 Modular Power System

Product Note

**Using the List and Trigger Capabilities to Sequence Outputs in These Applications:**

- Sequencing Multiple Modules During Power Up
- Sequencing Multiple Modules to Power Down on Event
- Controlling Output Voltage Ramp Up at Turn On
- Providing Time-Varying Voltages
- Providing Time-Varying Current Limiting
- Output Sequencing Paced by the Computer
- Output Sequencing Without Computer Intervention



**Agilent Technologies**



## Introduction

This note provides information on how you can use the advanced programmable features of the Agilent 66000 Modular Power System to address a variety of applications. Although your exact application may not be described here, the capabilities described can be generalized and applied to your specific needs. The programming examples are given in Microsoft Visual Basic 6.0, and the appendix has an HP BASIC programming example.

These are the capabilities that are discussed and a description of how they can be applied:

<b>1. Sequencing Multiple Modules During Power Up</b>	10
When testing mixed signal devices, $\pm$ bias supply voltages are typically applied before logic bias supply voltages. For a device that is sensitive to when bias voltages are applied, the order of power-up of multiple power modules can be controlled.	
<b>2. Sequencing Multiple Modules to Power Down on Event</b>	15
When testing devices, such as some GaAs and ECL devices that are sensitive to when bias voltages are removed, the order of power-down of multiple power modules can be controlled. The power-down sequence can be initiated by an event, such as a change in power module status, fault condition, detection of a TTL signal, etc.	
<b>3. Controlling Output Voltage Ramp Up at Turn On</b>	20
When control over the rate of voltage ramp-up at turn-on of the power module output is required, the required shape can be approximated by downloading and executing a series of voltage and dwell time points.	
<b>4. Providing Time-Varying Voltages</b>	25
To burn-in devices using thermal or mechanical cycling/stress, cyclical time-varying voltage is provided by programming a set of voltage and dwell time points that repetitively sequence over time.	
<b>5. Providing Time-Varying Current Limiting</b>	30
To provide current limit protection which varies as a function of time, multiple thresholds on current limit are required. Having multiple thresholds can provide a high limit to protect the device under test (DUT) during its power-up in-rush with automatic switchover to a lower limit to protect the DUT during its steady state operation.	
<b>6. Output Sequencing Paced by the Computer</b>	35
When performing bias supply margin testing, throughput can be maximized by eliminating command processing time associated with reprogramming all outputs for each set of limit conditions. Instead, multiple sets of bias limit conditions can be downloaded to the power modules during test system initialization. During the testing, the computer can use a single command to simultaneously signal all power modules to step through each test condition.	
<b>7. Output Sequencing Without Computer Intervention</b>	41
When characterizing devices, the DUT's performance is measured over a range of power supply voltages. This test can be performed without computer intervention by using hardware signals from the measurement instrument to cause the power module to sequence to the next voltage in a preprogrammed List. By buffering these readings in the measurement instrument, the entire test can be executed without computer involvement. For characterizations that require long measurement times, the computer is free to do other tasks. For characterizations that must execute at hardware speeds, the computer is not involved and will not slow down the test loop.	

## Introduction to the Agilent 66000 Modular Power System

The Agilent 66000 Modular Power System (MPS) is a member of the Agilent “One-Box” Solution family of power supplies. The “One-Box” family is the next generation of power supplies and offers many advantages over other power supply solutions.

The MPS consists of the Agilent 66000A Modular Power Mainframe and a number of different Agilent 66100A series dc power modules. Each MPS dc power module contains a dc power supply with automatic CV/CC crossover, voltage and current programmers, a DMM with precision current shunt, and a digital interface. This allows you to directly program voltage and current, read back the actual measured voltage and current, and monitor the status conditions of the power module. Each module is optimized for ATE applications and offers all the built-in features needed for easy integration into your test system.

The MPS dc power modules feature a SCPI (Standard Commands for Programmable Instruments) command set. This industry standard command set simplifies test system development by offering command set commonality between all types of instrumentation so that all instruments performing the same function use the same self-documenting SCPI instructions. For example, measuring the actual output voltage from a dc power module and reading the voltage from an external multimeter use the same SCPI commands. Because you spend less time learning device commands, you can get your application up and running faster.

The following sections of this note explain how to get the most from the advanced programming features of the MPS dc power modules. These explanations are followed by seven practical example applications that show how to apply these advanced features.

## Agilent 66000 MPS Triggering

### How Can I Benefit from Triggers?

With most system power supplies, the only way to make it perform some action is to send it a command over GP-IB. Upon receipt of that command, the power supply processes the command and executes the action. This can cause two problems:

1. When many commands must be processed, the combined command processing time can slow down test execution.
2. When multiple power supplies must be programmed to act in unison, the command processing time can cause the resultant outputs to occur out of synchronization.

The MPS dc power modules offer a solution to these problems by providing the ability to send and receive triggers. With triggers, the new voltage and current levels are sent to the power module before they are needed so that they can be pre-processed. When you want to change the output, you send a trigger and the module immediately changes its output to the new programmed settings, without command processing delays. You no longer need to explicitly program the voltage and current each time you needed to change the output setting.

Triggers can be used when you require multiple bias modules to act in unison (for example, to apply power from three

modules to the DUT simultaneously). Instead of separately programming the voltage and current settings for each of the three modules (and having them reach their voltage settings at different times), you can preprogram the modules to the required correct output. You can then trigger all three at once and they will reach their desired outputs together.

In addition to responding to triggers, the power module can provide a Trigger Out signal. This signal can be used to notify other hardware in your test system that the power module has changed its output. For example, Trigger Out can be connected to the Trigger In of a digital oscilloscope to request it to begin making a measurement on the DUT. With this method of synchronization, a measurement is valid because it is triggered as soon as the power module reaches its programmed voltage but not before it has had time to do so. This method also maximizes throughput because the measurement begins as soon as the module reaches its required output voltage.

**What Actions Can I Trigger?**

The MPS dc power modules support triggering of any of the following, called “trigger actions”:

- a change in output voltage
- a change in output current
- a start of a List sequence (see next section)
- the pace of a List sequence (see next section)

The MPS dc power modules also have a programmable trigger delay. This allows you to insert a time interval between the time a trigger is detected by the module and the time the trigger action occurs.

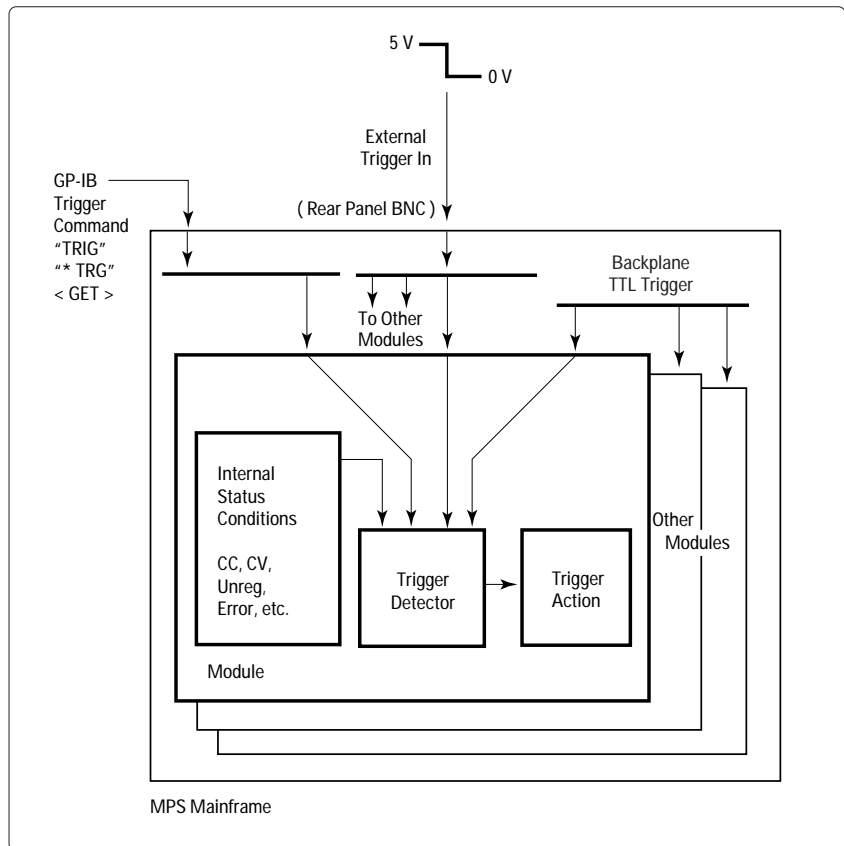
**From Where Do I Receive Triggers?**

The MPS dc power modules can receive triggers from the following trigger sources:

**The GP-IB.** The computer can send trigger commands to the modules. There is a short command processing time associated with this source.

**External Trigger In.** This is the MPS mainframe TRIGGER IN connector. It is a BNC connector wired to a backplane bus that provides a trigger input common to all modules. TRIGGER IN accepts TTL levels, with the falling edge detected as the trigger.

**Status Conditions.** A change in a module’s status can be used to generate a trigger. For example, you can use the constant current status condition to trigger the module to change its output voltage.



**Figure 1.** MPS Trigger Sources

**TTL Trigger.** The TTL Trigger bus is a backplane bus that can be used as a trigger input by any module. (See next section for how to drive the backplane TTL Trigger bus.)

**How Do I Generate Triggers?**

Any module can be programmed to drive the backplane TTL Trigger bus. By means of this bus, a module can trigger all other modules that have been programmed to respond to the TTL Trigger bus as their trigger source (see the preceding paragraph).

The TTL Trigger bus is wired to the MPS mainframe TRIGGER OUT BNC connector. The TRIGGER OUT signal is a 20-microsecond, negative-true TTL pulse. Whenever a module drives the backplane TTL Trigger bus, it also drives the TRIGGER OUT connector on the MPS mainframe.

The MPS dc power modules can drive the TTL Trigger line in response to:

**The GP-IB.** When the computer sends a trigger command to a module, it can drive the TTL Trigger bus. This permits you to trigger one module from the computer and have that module “echo” the trigger command over the TTL Trigger bus, simultaneously triggering all other modules.

**External Trigger In.** A module can drive the TTL Trigger bus in response to a trigger received from the mainframe TRIGGER IN connector.

**Status Conditions.** A change in a module’s status can be used to drive the TTL Trigger bus. This allows a change in status in one module to trigger other modules. For example, you can use a current limit condition in one module to trigger the other modules to perform a trigger action.

**How Do I Enable the MPS to Respond to Triggers?**

The default state of the dc power module is the idle state, where trigger detection is disabled.

Before it can respond to a trigger, the module must be placed in the “initiated” state (by a computer command). Once initiated, the module can detect a trigger from the selected source. This is the only time when the module responds to a trigger; no trigger detection occurs in the idle state (above) or in the “triggered” state (next paragraph).

Once the trigger is detected, the module moves into a triggered state and performs the trigger action (after waiting any programmed trigger delay time). Upon completion of the trigger action, the module returns to the idle state.

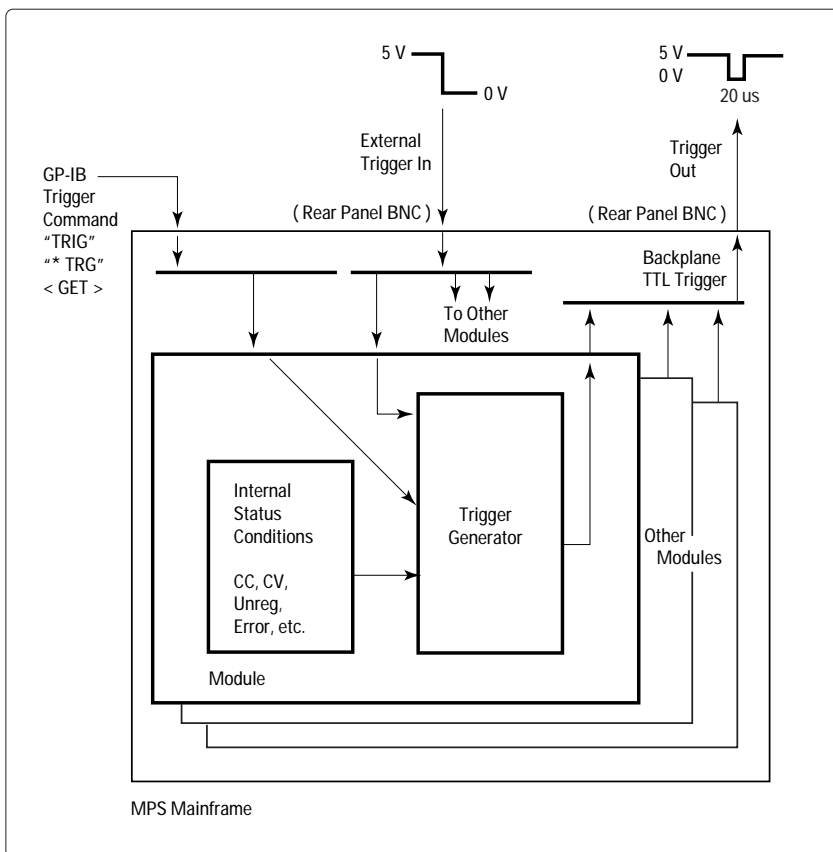


Figure 2. Generating TTL Triggers

## Agilent 66000 MPS Lists

### What is a List and How Can I Benefit from its Use?

Normally, an MPS dc power module operates in a mode called “Fixed”, where the output settings stay fixed at the programmed value until you send another command to change them. Another available mode is called “List” mode, where no computer commands are needed to alter the output settings.

Lists allow you to download a sequence of voltage and/or current values that will be generated at specific intervals or on the occurrence of a trigger. Up to 20 voltage and current values (points), with 20 associated dwell times, can be programmed. Once downloaded, the dc power module can execute the sequence of output settings with minimal interaction between the computer and the dc power module. By pacing a List using the programmable dwell times, you can time output changes more precisely. By pacing a List using triggers, you can better synchronize changes in the output with asynchronous events.

### Can a List be Repeated?

Although each List can contain up to 20 points, a List can be programmed to repeat. The number of repetitions, called the List count, is programmable from 1 to 65534 loops. It can also be set to repeat continuously by making the count infinite.

### How Do I Execute a List?

If you want the voltage to change per a downloaded List, set the voltage mode to List. Similarly, if you want the current to change per a downloaded List, set the current mode to List. If you want both to change, set both to List mode.

A module begins executing a List in response to a trigger. All of the trigger sources mentioned under “Agilent 66000 MPS Triggering” are valid. Once triggered, Lists can be paced by dwell times or paced by triggers.

### Dwell-Paced Lists

When the List is dwell paced, each List point is programmed for the dwell time associated with that point. When the dwell time expires, the output changes to the next point in the List.

Valid dwell times for the MPS dc power modules are from 10 milliseconds to 65 seconds per point. The resolution is in 2-millisecond increments. The command to select dwell-paced List mode is LIST:STEP AUTO. This commands the List to step automatically to the next point after expiration of the dwell time. If the List count is greater than 1, the List will automatically repeat until the count has been satisfied. With dwell pacing, the execution of the entire List is the triggered action, so only one trigger is required.

The diagram at right shows the timing for the start trigger, status conditions, and output voltage changes for a dwell paced List. The programmed parameters are:

---

List points:    5 V for 50 ms  
                   10 V for 150 ms  
                   0 V for 30 ms

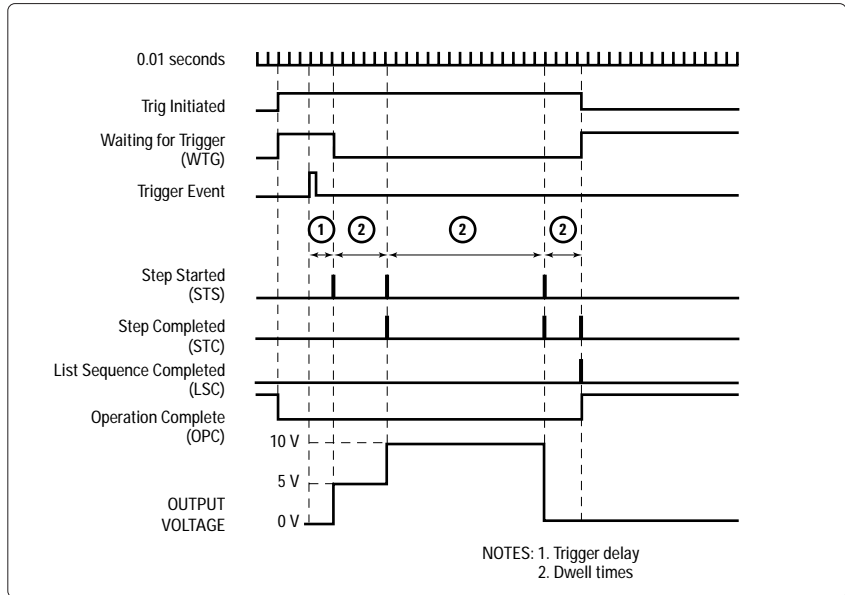
---

Trigger delay: 20 ms

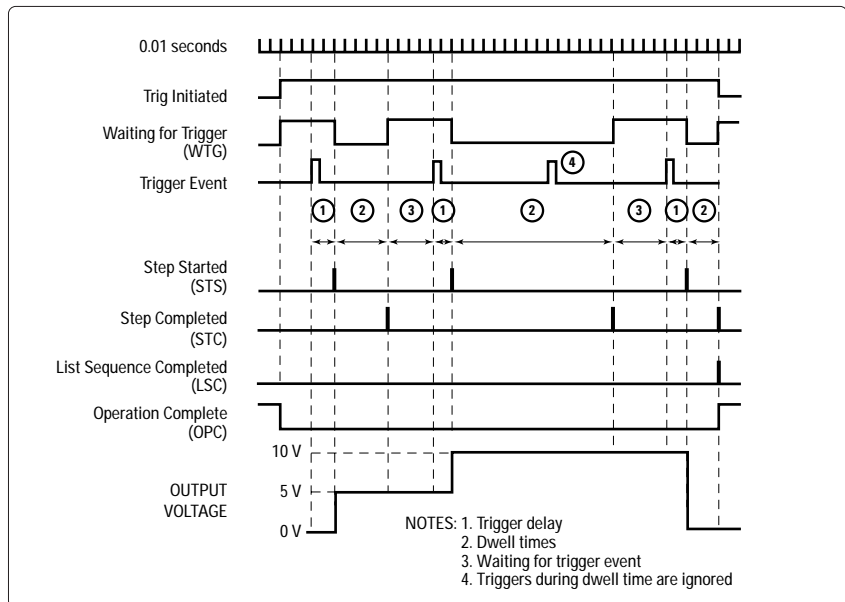
**Triggering Each Point in a List**

When the List is paced by triggers, each List point is generated when a trigger is detected. When the dwell time expires, the output stays at the same value until the next trigger is detected. Thus, one trigger is required to sequence the List from each point to the next. Any triggers received during a point's dwell time are ignored. The command to select trigger-paced List mode is LIST:STEP ONCE. This commands the List to execute only one List point (step) upon receipt of each trigger.

Note that since the module ignores triggers during the dwell time, you must be careful not to program a dwell time that is so long that you will miss the next trigger. If you are unsure of when triggers will be sent and you want to guarantee that you get all of them, set the dwell time to its minimum value.



**Figure 3.** Timing Diagram for Dwell-Paced Lists



**Figure 4.** Timing Diagram for Trigger-Paced Lists

The diagram above shows the timing for triggers, status conditions, and output voltage changes for a trigger paced List. The programmed parameters are:

---

List points:    5 V for 50 ms  
                   10 V for 150 ms  
                   0 V for 30 ms

---

Trigger delay: 20 ms



## Using Triggers with Lists to Implement Output Sequences

This section describes how to make several modules respond synchronously. The general concepts described here provide the basis for many synchronized multiple-output applications.

Each MPS dc power module functions independently. To get them to operate in unison, a common synchronizer must be established. This can be done by having a single trigger source or timebase for all modules.

For example, to sequence 3 modules to power up 50 ms apart, you could use the three simple Lists in the table below:

As you can see, although each module functions independently from the other, the result is a sequenced output. This simple method has two problems if tight synchronization is needed:

- How do you get three Lists to start simultaneously?
- How do you guarantee that the dwell time clocks in all modules stay synchronized?

The following sections describe how the MPS solves the problems of synchronization.

### GP-IB Triggering Between Modules

Sometimes a common trigger signal is not available. For example, when it sends a trigger command, the computer cannot talk to all devices on the GP-IB simultaneously. The time required to individually trigger each dc power module could cause an undesirable “skewing” of the trigger actions.

You can use the backplane TTL Trigger bus of the mainframe to solve the lack of a common trigger. The procedure is as follows: Program the first module to generate a backplane TTL Trigger when it receives a trigger command over GP-IB. Then, command all other modules to use the backplane TTL Trigger bus as a source. The computer then triggers only the first module, which “echos” that trigger over the backplane to all other modules simultaneously.

### Status Triggering Between Modules

Some applications require a trigger to occur in response to a change in status within a module. Since change-of-status detection is internal to each module, one module is unaware of, and therefore cannot react to, status changes within another module. Using the mainframe backplane TTL Trigger bus solves this. The module that detects a status change can be programmed to not only trigger itself to perform some trigger action, but also to send a TTL Trigger out on the backplane bus. This triggers all the other modules.

Step no.	Module #1		Module #2		Module #3	
	Voltage List	Dwell Time	Voltage List	Dwell Time	Voltage List	Dwell Time
1	5 V	50 ms	0 V	50 ms	0 V	50 ms
2	5 V	50 ms	5 V	50 ms	0 V	50 ms
3	5 V	50 ms	5 V	50 ms	5 V	50 ms

**A Generalized Synchronizing Scheme**

It is desirable to simultaneously start and then maintain synchronization between Lists running in multiple modules. You can use a combination of GP-IB and status triggering to synchronize several Lists. Use the following approach:

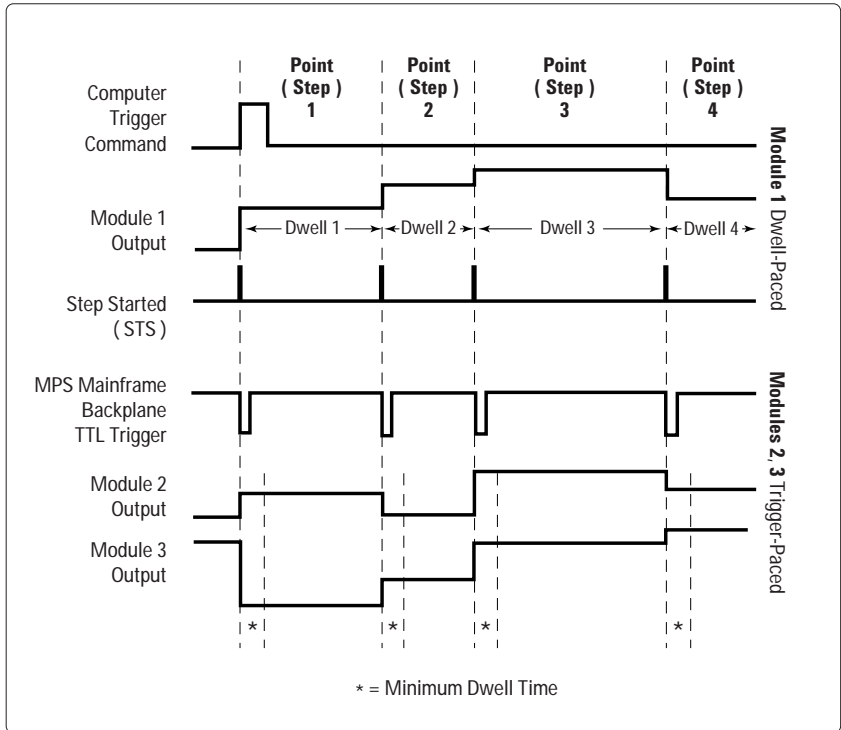
**For Module 1...**

- Trigger the module from a computer trigger command over GP-IB
- Generate the output voltage from a List
- Use a dwell-paced List. The computer trigger command causes the execution of the entire List.
- Have the module generate a backplane TTL Trigger at the start of each point (step) in the List. This can be done by triggering on the STS (Step Started) status condition.

**For Modules 2 and 3...**

- Trigger the module from the backplane TTL Trigger bus
- Generate the output voltage from a List
- Use a trigger-paced List. Each backplane TTL Triggers causes the execution of one point of the List.
- Program the dwell times to minimum (so you don't miss any triggers)

When the computer sends the trigger command to module 1, it will begin executing its List. Module 1 will step through its List of points, generating a TTL Trigger each time the Step Started (STS) condition occurs at the start of each new point. Modules 2, 3... will wait for these triggers before proceeding to their next point. The result is an "unskewed" synchronized output, started by a single computer command.



**Figure 5.** Timing Diagram for the Generalized Synchronizing Scheme

## Applications

This section contains seven example applications. For each application, there is:

- An overview of the application
- Which MPS features are used to implement the application
- The advantages and benefits of the MPS solution
- The details of the implementation of the solution
- A block diagram of the setup
- A description of variations on the application
- A sample program listing in Microsoft Visual Basic 6.0

The sample programs were developed and tested using the following software/hardware:

- Microsoft Visual Basic 6.0
- Agilent I/O Library version M.01.00
- Agilent VISA COM (can be downloaded from the Agilent Developers Network website, see: <http://adn.tm.agilent.com/>)
- Microsoft Windows 2000
- Agilent 82350B PCI GPIB Interface

The following table lists what MPS features are used in each of the applications. It can be used as an index into this section.

Application 1. Sequencing Multiple Modules During Power Up

Application 2. Sequencing Multiple Modules to Power Down on Event

Application 3. Controlling Output Voltage Ramp Up at Turn On

Application 4. Providing Time-Varying Voltages

Application 5. Providing Time-Varying Current Limiting

Application 6. Output Sequencing Paced by the Computer

Application 7. Output Sequencing Without Computer Intervention

	Application						
	1	2	3	4	5	6	7
<b>Lists</b>							
20-point current List					•		
20-point voltage List			•	•		•	•
Repetitive Lists				•			
Dwell time			•	•	•		•
<b>List Pacing</b>							
Dwell-paced Lists			•	•	•		
Trigger-paced Lists						•	•
<b>Actions due to a change in status</b>							
Generate an SRQ				•			•
Generate a trigger		•					•
Disable the output				•	•		
Stop the List				•	•		
<b>Triggers</b>							
Change the voltage on trigger	•	•			•		
Trigger in/out from MPS backplane TTL Trigger	•	•				•	•
Trigger on an GP-IB trigger command	•			•	•	•	•
Trigger delay	•	•					
<b>Other Features</b>							
Active downprogramming		•		•	•		
Overcurrent protection				•	•		

## Application 1. Sequencing Multiple Modules during Power Up

### Overview of application

When testing mixed signal devices,  $\pm$  bias supply voltages are typically applied before logic bias supply voltages. For a device that is sensitive to when bias voltages are applied, the order of power up of multiple power modules can be controlled.

For this example, the device requires three bias supplies, +5 V for the logic circuits and  $\pm 15$  V for amplifier circuits. To properly power up the device, the supplies must be sequenced so that the  $\pm 15$  V are applied first and the +5 V is applied 50 ms later.

The MPS can easily address this application through the use of triggers. The trigger will cause the modules to change from 0 V, where they are not powering the DUT, to their final voltage. By delaying the response to the trigger, you can control when the module's output voltage changes. This means you can control the sequence of the modules during power up.

### MPS features used

- Change the voltage on trigger
- Trigger in/out from MPS mainframe backplane TTL Trigger
- Trigger on a GP-IB trigger command Trigger delay
- Trigger delay

### Advantages/benefits of the MPS solution

- By using trigger delay, the timing is accurate and repeatable.
- The sequence is simpler to program (no timing loops).
- The computer is not devoted to sequencing power modules.
- The computer does not provide timing for the sequence.
- One command initiates the sequence.

### Implementation details

#### How the MPS implements the sequence

- The computer sends a trigger command to the first module.
- The first module simultaneously sends a backplane trigger to other two modules and goes to +15 V.

- The second module receives the backplane TTL Trigger and immediately goes to  $-15$  V.
- The third module receives the backplane TTL Trigger, delays 50 ms, and then goes to +5 V.

### MPS set up

#### Module in slot 0:

- The module is connected to +15 V on the DUT.
- The initial voltage setting is 0 V.
- The module listens for the computer to send a trigger command.
- Upon receipt of the trigger command, the module goes to 15 V.
- Also upon receipt of the trigger command, the module generates a backplane TTL Trigger.

*Module in slot 1:*

- The module is connected to -15 V on the DUT.
- The initial voltage setting is 0 V.
- The module listens for a backplane TTL Trigger.
- Upon receipt of the trigger, the module goes to 15 V.

*Module in slot 2:*

- The module is connected to +5 V on the DUT.
- The initial voltage setting is 0 V.
- The module listens for a backplane TTL Trigger.
- The trigger delay is programmed to 50 ms.
- Upon receipt of the trigger, the module waits the trigger delay time and then goes to 5 V.

**Variations on this implementation**

1. The modules could be set to generate SRQ when the last module (+5 V) reaches its final output value. This would notify the computer that power has been applied to the DUT and the testing can begin.
2. To provide a delay between the application of the +15 V and the -15 V bias, you can program different trigger delays into modules 2 and 3. The delay time will be relative to module in slot 0.
3. To get all three modules to apply power to the DUT at the same time, simply eliminate the trigger delay on the +5 V module.
4. When modules need to be connected in parallel to increase current, they will also need to be synchronized so that they all apply power simultaneously. To get modules in parallel to apply power at the same time, use the approach described in this example, but eliminate any trigger delays.

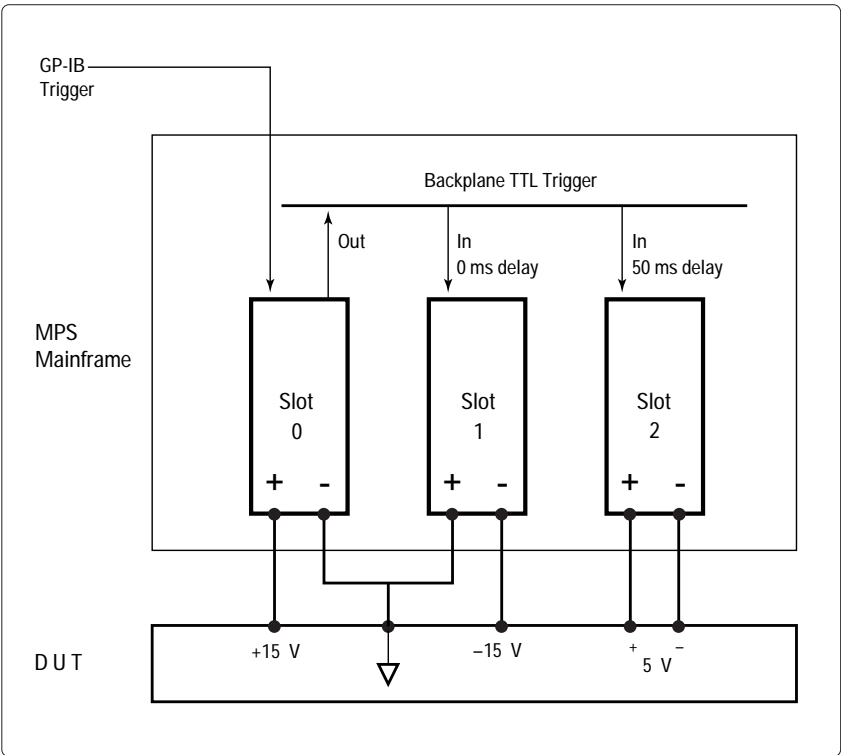


Figure 1-1. Block Diagram of Application #1

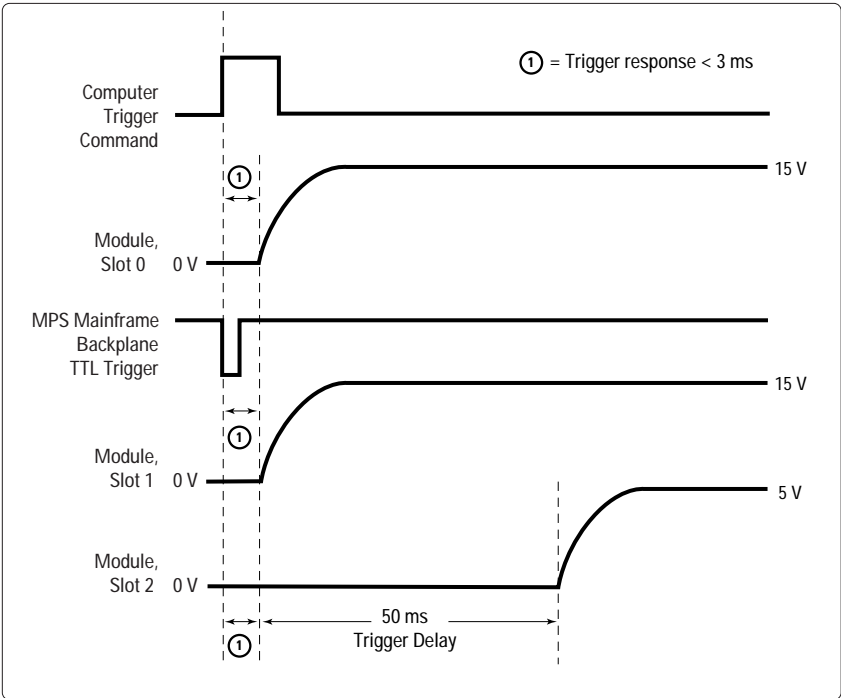


Figure 1-2. Timing Diagram of Application #1

### 'Application #1: Sequencing Multiple Modules During Power Up

```
Private Sub CommandApp1_Click()
' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT
' VISA COM OBJECTS.
'
' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES
' IN THE PROJECT/REFERENCES MENU:
' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtIRM.DLL
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL
'
' USE A STATEMENT SUCH AS "Dim MPS_Slot As VisaComLib.FormattedIO488" TO CREATE THE
' FORMATTED I/O REFERENCE AND USE "Set MPS_Slot = New VisaComLib.FormattedIO488" TO CREATE
' THE ACTUAL OBJECT

Dim io_mgr As VisaComLib.ResourceManager
Dim MPS_Slot0 As VisaComLib.FormattedIO488
Dim MPS_Slot1 As VisaComLib.FormattedIO488
Dim MPS_Slot2 As VisaComLib.FormattedIO488

Set io_mgr = New AgilentRMLib.SRMCIs
Set MPS_Slot0 = New VisaComLib.FormattedIO488
Set MPS_Slot1 = New VisaComLib.FormattedIO488
Set MPS_Slot2 = New VisaComLib.FormattedIO488

Set MPS_Slot0.IO = io_mgr.Open("GPIB0::5::0::INSTR")
Set MPS_Slot1.IO = io_mgr.Open("GPIB0::5::1::INSTR")
Set MPS_Slot2.IO = io_mgr.Open("GPIB0::5::2::INSTR")

Dim st_value As Integer          ' HOLD STATUS VALUE

On Error GoTo MyError

' SET UP MODULE IN SLOT 0 AS +15V BIAS SUPPLY -----
With MPS_Slot0
    .IO.Timeout = 30000

    .WriteString "*RST;*CLS;Status:Preset"      ' RESET AND CLEAR MODULE
    .WriteString "VOLT 0"                        ' START AT 0V
    .WriteString "VOLT:TRIGGERED 15"            ' GO TO 15V ON TRIGGER
    .WriteString "TRIGGER:SOURCE BUS"           ' TRIGGER SOURCE IS GPIB 'BUS'
    .WriteString "OUTPUT:TTLTRG:SOURCE BUS"     ' GENERATE BACKPLANE TTL TRIGGER WHEN GPIB 'BUS'
                                                ' TRIGGER IS RECEIVED
    .WriteString "OUTPUT:TTLTRG:STATE ON"       ' ENABLE BACKPLANE TTL TRIGGER DRIVE
    .WriteString "OUTPUT ON"                   ' ENABLE OUTPUT
    .WriteString "INITIATE"                    ' ENABLE RESPONSE TO TRIGGER
End With

' SET UP MODULE IN SLOT 1 AS -15V BIAS SUPPLY -----
With MPS_Slot1
    .WriteString "*RST;*CLS;Status:Preset"      ' RESET AND CLEAR MODULE
    .WriteString "VOLT 0"                        ' START AT 0V
    .WriteString "VOLT:TRIGGERED 15"            ' GO TO 15V ON TRIGGER
    .WriteString "TRIGGER:SOURCE TTLTRG"       ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
    .WriteString "OUTPUT ON"                   ' ENABLE OUTPUT
    .WriteString "INITIATE"                    ' ENABLE RESPONSE TO TRIGGER
End With
```

End With

' SET UP MODULE IN SLOT 2 AS +5V BIAS SUPPLY -----

With MPS\_Slot2

```
.WriteString "**RST;*CLS;Status:Preset"      ' RESET AND CLEAR MODULE
.WriteString "VOLT 0"                          ' START AT 0V
.WriteString "VOLT:TRIGGERED 5"                ' GO TO 5V ON TRIGGER
.WriteString "TRIGGER:SOURCE TTLTRG"          ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
.WriteString "TRIGGER:DELAY 0.050"            ' 50 ms TRIGGER DELAY
.WriteString "OUTPUT ON"                      ' ENABLE OUTPUT
.WriteString "INITIATE"                       ' ENABLE RESPONSE TO TRIGGER
```

End With

'

' BEFORE TRIGGERING THE MODULES, DETERMINE IF THE MODULES ARE READY BY CHECKING FOR

' 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER). IF THE LAST MODULE

' PROGRAMMED IS READY THEN SO ARE THE OTHERS, SO JUST CHECK SLOT 2.

'

' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY

' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS

' THAT TAKE TIME WILL GIVE THE MODULE A CHANCE TO COMPLETE PROCESSING.

'

Do

With MPS\_Slot2

```
.WriteString "STATUS:OPERATION:CONDITION?"
```

```
st_value = .ReadNumber
```

End With

Loop Until (st\_value & 32) ' TEST FOR BIT 5 = TRUE

' TRIGGER MODULE IN SLOT 0 TO BEGIN SEQUENCING THE 3 MODULES AT POWER UP

'

```
MPS_Slot0.WriteString "**TRG"          ' SEND GPIB 'BUS' TRIGGER
```

' CLOSE THE I/O SESSIONS.

```
MPS_Slot0.FlushRead
```

```
MPS_Slot0.FlushWrite False
```

```
MPS_Slot0.IO.Close
```

```
Set MPS_Slot0 = Nothing
```

```
MPS_Slot1.FlushRead
```

```
MPS_Slot1.FlushWrite False
```

```
MPS_Slot1.IO.Close
```

```
Set MPS_Slot1 = Nothing
```

```
MPS_Slot2.FlushRead
```

```
MPS_Slot2.FlushWrite False
```

```
MPS_Slot2.IO.Close
```

```
Set MPS_Slot2 = Nothing
```

Exit Sub

MyError:

```
txtError = Err.Description & vbCrLf
```

```
Resume Next
```

End Sub

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)



## Application 2. Sequencing Multiple Modules to Power Down on Event

### Overview of application

When testing devices, such as some GaAs and ECL devices that are sensitive to when bias voltages are removed, the order of power-down of multiple power modules can be controlled. The power-down sequence can be initiated by an event, such as a change in power module status, fault condition, detection of a TTL signal, etc.

For this example, there are three supplies +5 V and  $\pm 15$  V. (See previous application for how to generate a power up sequence.) Once the power has been applied to the DUT, the modules can be reprogrammed to perform the power down sequence. The power down sequence is initiated when a fault in the DUT draws excessive current from the power module, causing the module to change from CV to CC. To prevent damage to the DUT, it is necessary to remove the +5 V first, then the  $\pm 15$  V modules 15 ms later.

Once again, MPS triggering can solve the application. In this scenario, the CV-to-CC crossover event will be used as the trigger source. The trigger will cause the modules, in the correct order, to change from their programmed voltages down to 0 V.

### MPS features used

- Generate a trigger on a change in internal status
- Change the voltage on trigger
- Trigger in/out from MPS mainframe backplane TTL Trigger
- Trigger delay
- Active downprogramming

### Advantages/benefits of the MPS solution

- By using the modules' change in status to automatically generate a trigger, the computer is not devoted to polling the modules to detect a change in state.
- By letting each module monitor its status, the CC condition will generate a response faster than if the computer was polling the module to detect a change in state.
- The sequence is simpler to program (no timing loops).
- By using trigger delay, the timing is accurate and repeatable because the computer does not provide timing for the sequence.
- The active downprogrammers in the module output can quickly discharge the module's output capacitors and any capacitance in the DUT.

### Implementation details

#### How the MPS implements the solution

- All modules are set to listen for a backplane TTL Trigger.
- When any module detects a change in status from CV to CC, it sends out a backplane TTL Trigger.
- When the +5 V module receives the trigger, it immediately goes to 0 V.
- When the +15 V and -15 V modules receive the trigger, they wait the trigger delay time and then go to 0 V.

(Note that any module can both generate the backplane TTL Trigger signal and be triggered by that same signal.)

### **MPS set up**

#### *Module in slot 0:*

- The module is connected to +15 V on the DUT.
- The initial voltage setting is 15 V.
- The module monitors its status.
- The module will generate a backplane TTL Trigger on CV-to-CC crossover.
- The module listens for a backplane TTL Trigger.
- The trigger delay is programmed to 15 ms.
- Upon receipt of the trigger, the module waits the trigger delay time and then goes to 0 V.

#### *Module in slot 1:*

- The module is connected to supply -15 V to the DUT.
- The initial voltage setting is 15 V.
- The module monitors its status.
- The module will generate a backplane TTL Trigger on CV-to-CC crossover.

- The module listens for backplane TTL Trigger.
- The trigger delay is programmed to 15 ms.
- Upon receipt of the trigger, the module waits the trigger delay time and then goes to 0 V.

#### *Module in slot 2:*

- The module is connected to supply +5 V to the DUT.
- The initial voltage setting is 5 V.
- The module monitors its status.
- The module will generate a backplane TTL Trigger on CV-to-CC crossover.
- The module listens for backplane TTL Trigger.
- Upon receipt of the trigger, the module immediately goes to 0 V.

### **Variations on this implementation**

1. The modules could be set to generate SRQ when the last module reaches 0 V. This could notify the computer that power has been removed from the DUT.
2. The modules could be set to generate a DFI (Discrete Fault Indicator) signal on the MPS rear panel on a change in status. This signal could be used to shut down other power modules, to flash an alarm light, or to sound a buzzer. This could also be routed to other instruments to signal them to stop making measurements.
3. To get all three modules to remove power from the DUT at the same time, simply eliminate the trigger delay on the  $\pm 15$  V modules.
4. To provide a delay between the removal of the three bias voltages, you can program a different trigger delay into each module.

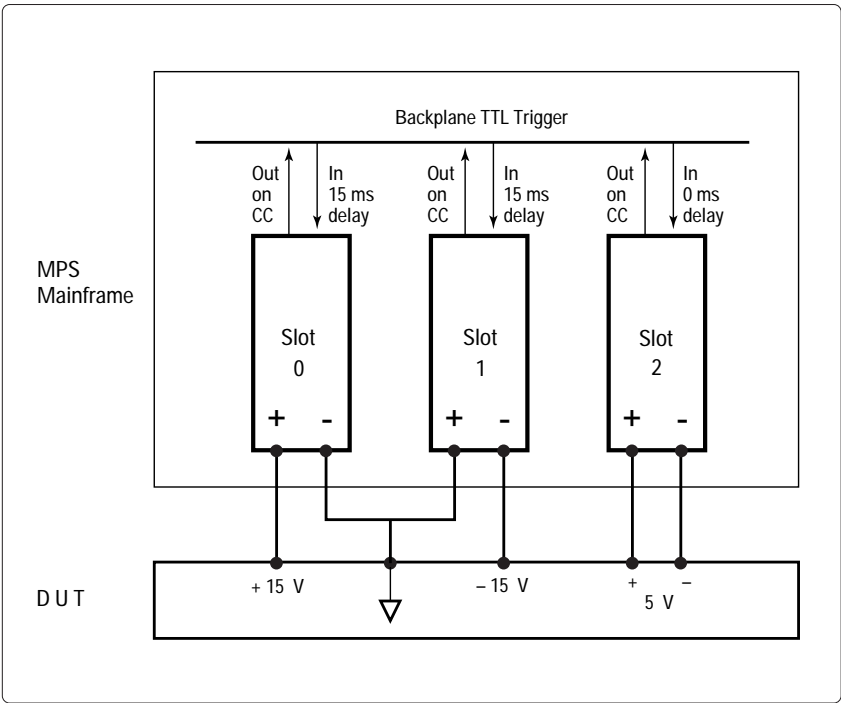


Figure 2-1. Block Diagram of Application #2

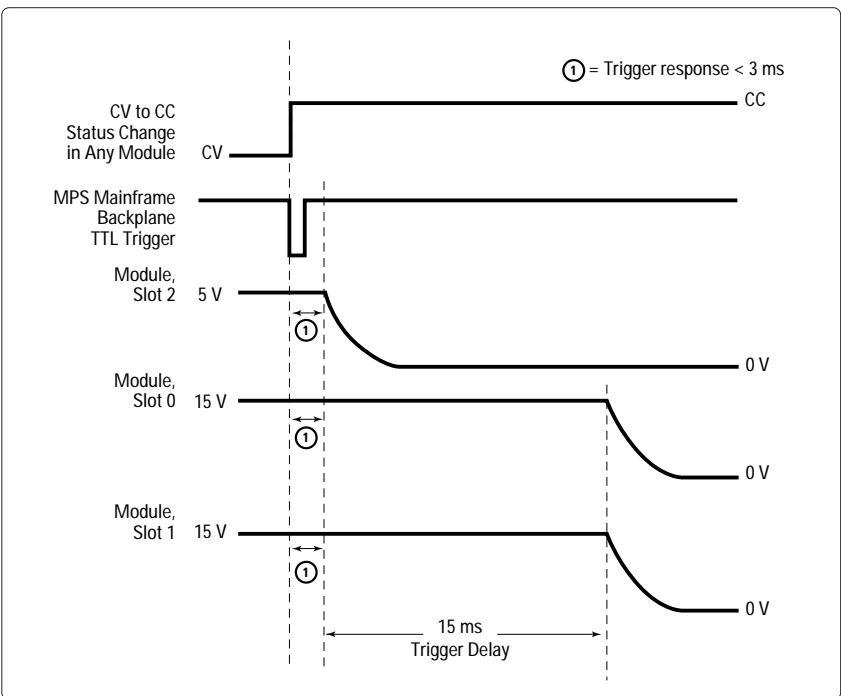


Figure 2-2. Timing Diagram of Application #2

**'Application #2: Sequencing Multiple Modules to Power Down on Event**

Private Sub CommandApp2\_Click()

' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.

' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:

' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL

' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb

' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL

' USE A STATEMENT SUCH AS "Dim MPS\_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS\_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT

Dim io\_mgr As VisaComLib.ResourceManager  
Dim MPS\_Slot0 As VisaComLib.FormattedIO488  
Dim MPS\_Slot1 As VisaComLib.FormattedIO488  
Dim MPS\_Slot2 As VisaComLib.FormattedIO488

Set io\_mgr = New AgilentRMLib.SRMClis  
Set MPS\_Slot0 = New VisaComLib.FormattedIO488  
Set MPS\_Slot1 = New VisaComLib.FormattedIO488  
Set MPS\_Slot2 = New VisaComLib.FormattedIO488

Set MPS\_Slot0.IO = io\_mgr.Open("GPIB0::5::0::INSTR")  
Set MPS\_Slot1.IO = io\_mgr.Open("GPIB0::5::1::INSTR")  
Set MPS\_Slot2.IO = io\_mgr.Open("GPIB0::5::2::INSTR")

On Error GoTo MyError

' SET UP MODULE IN SLOT 0 AS +15V BIAS SUPPLY -----

With MPS\_Slot0

.IO.Timeout = 30000  
.WriteString "\*\*RST;\*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE  
.WriteString "CURR .5" ' START AT 15V  
.WriteString "VOLT 15" ' GO TO 0V ON TRIGGER  
.WriteString "VOLT:TRIGGERED 0" ' TRIGGER SOURCE IS TTL TRIGGER  
.WriteString "TRIGGER:SOURCE TTLTRG" ' 15 ms TRIGGER DELAY  
.WriteString "TRIGGER:DELAY .015" ' ENABLE RESPONSE TO TRIGGER  
.WriteString "INITIATE" ' GENERATE A BACKPLANE TTL TRIGGER  
.WriteString "OUTPUT:TTLTRG:SOURCE LINK" ' WHEN A CV-TO-CC TRANSITION OCCURS  
.WriteString "OUTPUT:TTLTRG:LINK 'CC'" ' ENABLE TTL TRIGGER DRIVE  
.WriteString "OUTPUT:TTLTRG:STATE ON" ' ENABLE OUTPUT  
.WriteString "OUTPUT ON"

End With

' SET UP MODULE IN SLOT 1 AS -15V BIAS SUPPLY -----

With MPS\_Slot1

```

.WriteString "**RST;*CLS;STATUS:PRESET"      ' RESET AND CLEAR MODULE
.WriteString "CURR .5"
.WriteString "VOLT 15"                      ' START AT 15V
.WriteString "VOLT:TRIGGERED 0"            ' GO TO 0V ON TRIGGER
.WriteString "TRIGGER:SOURCE TTLTRG"       ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
.WriteString "TRIGGER:DELAY .015"          ' 15 ms TRIGGER DELAY
.WriteString "INITIATE"                    ' ENABLE RESPONSE TO TRIGGER
.WriteString "OUTPUT:TTLTRG:SOURCE LINK"   ' GENERATE A BACKPLANE TTL TRIGGER
.WriteString "OUTPUT:TTLTRG:LINK 'CC'"    ' WHEN A CV-TO-CC TRANSITION OCCURS
.WriteString "OUTPUT:TTLTRG:STATE ON"      ' ENABLE TTL TRIGGER DRIVE
.WriteString "OUTPUT ON"                  ' ENABLE OUTPUT

```

End With

' SET UP MODULE IN SLOT 2 AS +5V BIAS SUPPLY -----

With MPS\_Slot2

```

.WriteString "**RST;*CLS;STATUS:PRESET"      ' RESET AND CLEAR MODULE
.WriteString "CURR .5"
.WriteString "VOLT 5"                      ' START AT 5V
.WriteString "VOLT:TRIGGERED 0"            ' GO TO 0V ON TRIGGER
.WriteString "TRIGGER:SOURCE TTLTRG"       ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
.WriteString "TRIGGER:DELAY .015"          ' 15 ms TRIGGER DELAY
.WriteString "INITIATE"                    ' ENABLE RESPONSE TO TRIGGER
.WriteString "OUTPUT:TTLTRG:SOURCE LINK"   ' GENERATE A BACKPLANE TTL TRIGGER
.WriteString "OUTPUT:TTLTRG:LINK 'CC'"    ' WHEN A CV-TO-CC TRANSITION OCCURS
.WriteString "OUTPUT:TTLTRG:STATE ON"      ' ENABLE TTL TRIGGER DRIVE
.WriteString "OUTPUT ON"                  ' ENABLE OUTPUT

```

End With

' THE POWER MODULES ARE NOW SET UP TO IMPLEMENT THE POWER DOWN ON EVENT.

' ANY TIME ANY MODULE GOES INTO CC, THE SEQUENCE WILL OCCUR.

,

' CLOSE THE I/O SESSIONS.

```

MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

```

```

MPS_Slot1.FlushRead
MPS_Slot1.FlushWrite False
MPS_Slot1.IO.Close
Set MPS_Slot1 = Nothing

```

```

MPS_Slot2.FlushRead
MPS_Slot2.FlushWrite False
MPS_Slot2.IO.Close
Set MPS_Slot2 = Nothing

```

Exit Sub

MyError:

```

txtError = Err.Description & vbCrLf
Resume Next

```

End Sub

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

### **Application 3. Controlling Output Voltage Ramp Up at Turn On**

#### **Overview of application**

When control over the rate of voltage ramp up at turn-on of the power module output is required, the desired shape can be approximated by downloading and executing a series of voltage and dwell time points.

For this example, you need to program the power module to change its output from 2 volts to 10 volts, slewing through the 8 volt transition in 0.5 seconds. This results in a turn-on ramp-up of 16 V per second.

The MPS can create this voltage versus time characteristic using Lists. The desired characteristic (in this case, linear) is simulated using the 20 available voltage points. To determine the value of each point in the transition, simply divide the change in voltage by 20. To determine the dwell time of each voltage point, divide the total transition time by 19. After the List has been executed, the module will continue to output the final value (in this case, 10 volts) until the output has been reprogrammed to another value. Note that the dwell-time of the last point is not part of the transition time.

To determine the slowest ramp up (longest transition time) that can be generated, you must consider how smooth you need the voltage versus time characteristic to be. As the dwell time associated with each point gets longer, the output voltage will become more like a “stair step” and less like a linear transition. (See Figure 3-1)

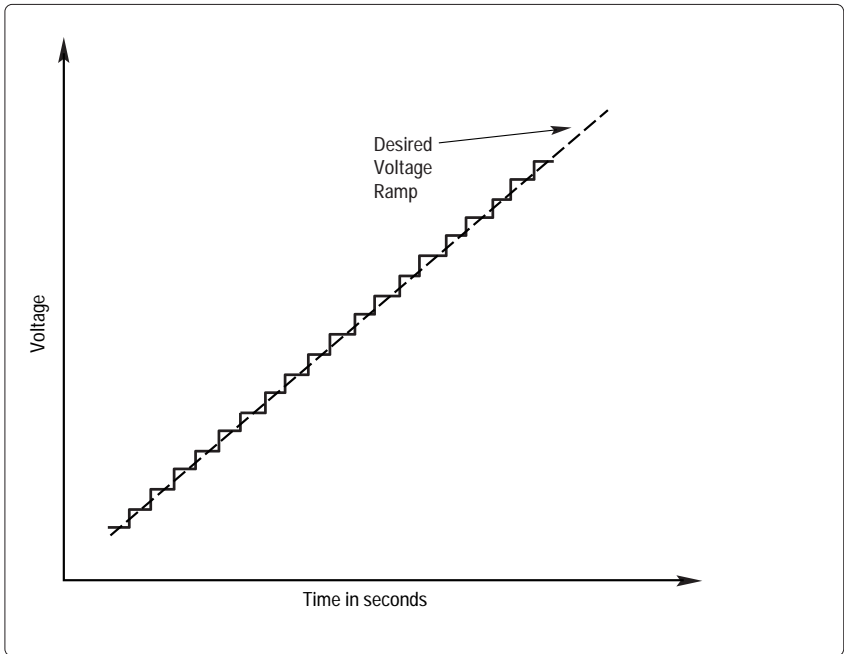
To determine the fastest ramp up (shortest transition time) that can be generated, you must consider the minimum dwell time specification (10 ms) and the maximum rise-time of specification the power module (20 ms). If you program 10 ms dwell times, the power module will not be able to reach its output voltage before the next voltage point is output. (See Figure 3-2)

#### **MPS features used**

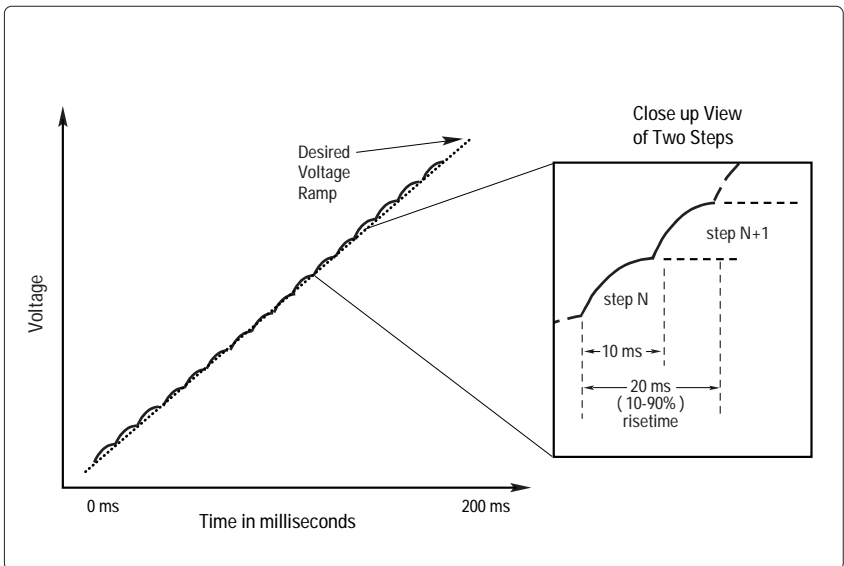
- 20-point voltage List
- Dwell time
- Dwell-paced Lists

#### **Advantages/benefits of the MPS solution**

- By using Lists, the module changes its output voltage automatically, so that the computer is not devoted to reprogramming the output voltage.
- The outputs can change faster when dwell paced than when the computer must explicitly reprogram each change.
- The sequence is simpler to program (no timing loops).
- By using dwell times, the timing of each point is accurate and repeatable.
- The computer does not provide timing for the sequence.
- For negative-going ramps, the active downprogrammers in the module output can quickly discharge the module's output capacitors and any capacitance in the DUT when negative going ramps are required.



**Figure 3-1.** Simulating a Slow Voltage Ramp



**Figure 3-2.** Simulating a Fast Voltage Ramp

### Implementation details

#### How the MPS implements the sequence

- The module is programmed to List mode.
- The module will execute a dwell-paced List.
- The 20 voltage points are downloaded to the module.
- The 20 dwell times are downloaded to the module.
- When the transition must occur, the module is triggered by the computer.
- The module output ramps under its own control.

#### Variations on this implementation

1. The module could be set to begin ramping in response to an external or backplane TTL Trigger.
2. The module could be set to generate SRQ when it has finished its transition. This would notify the computer that the voltage is at the proper level.
3. The module could be set to generate an external trigger when it has finished its transition. This trigger could be routed to other instruments as a signal to start making measurements.

4. Multiple modules could be programmed to slew together in response to the computer trigger command.
5. The module could be set to generate an external trigger for each point in the transition. This trigger could be routed to other instruments as a signal to take a measurement at various supply voltages. (See application #7)
6. Many voltage versus time characteristics can be generated by varying the voltage values and the dwell times in the List.

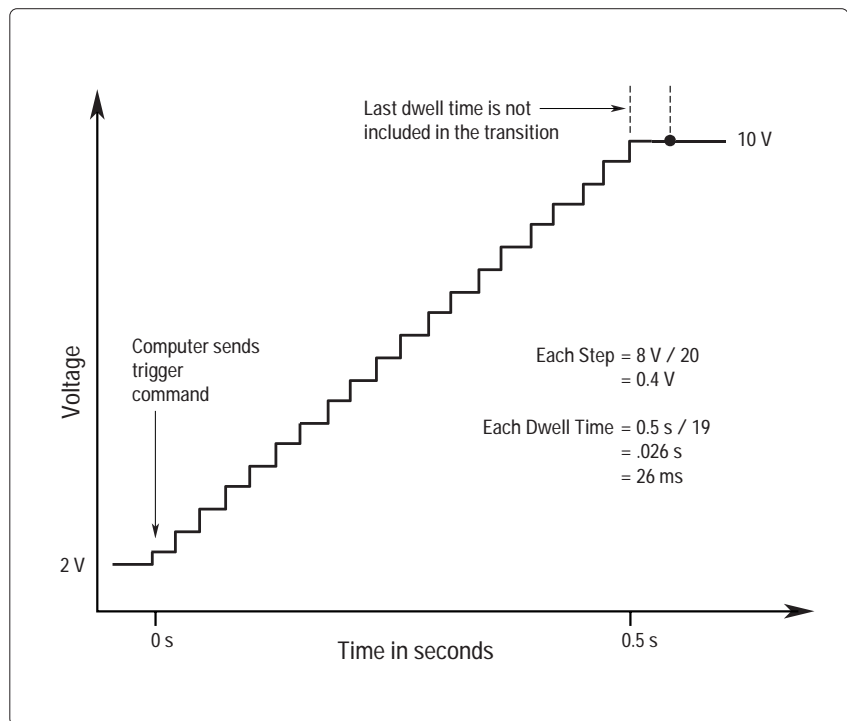


Figure 3-3. Generating the Desired Voltage Ramp for Application #3



### 'Application #3: Controlling Voltage Ramp Up at Turn On

Private Sub CommandApp3\_Click()

' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.

' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:

' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL  
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb  
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL

' USE A STATEMENT SUCH AS "Dim MPS\_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS\_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT

Dim io\_mgr As VisaComLib.ResourceManager  
Dim MPS\_Slot0 As VisaComLib.FormattedIO488

Set io\_mgr = New AgilentRMLib.SRMClS  
Set MPS\_Slot0 = New VisaComLib.FormattedIO488  
Set MPS\_Slot0.IO = io\_mgr.Open("GPIB0::5::0::INSTR")

On Error GoTo MyError

Dim V\_Step(20) As String       ' ARRAY TO HOLD THE VOLTAGE RAMP STEPS  
Dim V\_StepList As String

Dim i As Integer  
Dim Vstart, Vstop, Ramp\_time, Dwell As Double

Dim st\_value As Integer       ' HOLD STATUS VALUE

Vstart = 2                    ' START VOLTAGE FOR RAMP  
Vstop = 10                    ' STOP VOLTAGE FOR RAMP  
Ramp\_time = 0.5                ' SECONDS TO CHANGE FROM Vstart TO Vstop  
Dwell = Ramp\_time / 19        ' IN SECONDS

' SINCE THE OUTPUT STAYS AT THE LAST VOLTAGE POINT AFTER ITS DWELL TIME EXPIRES,  
' THE DWELL TIME OF THE LAST POINT IS NOT PART OF THE TRANSITION TIME. THEREFORE,  
' DIVIDE THE TOTAL TIME BY 19 POINTS, NOT 20. ALSO, YOU ONLY NEED TO DOWNLOAD 1  
' DWELL TIME. IF THE MODULE RECEIVES ONLY 1 DWELL TIME, IT ASSUMES YOU WANT THE  
' SAME DWELL TIME FOR EVERY POINT IN THE LIST.

For i = 0 To 19  
    V\_Step(i) = Vstart + (((Vstop - Vstart) / 20) \* (i + 1))   ' CALCULATES VOLTAGE POINTS  
Next i

V\_StepList = Join(V\_Step, ",")                               ' FORMAT THE VOLTAGE STEP LIST INTO A COMMA SEPARATED STRING  
  ' "2.4,2.8,3.2,3.6,4.0,4.4,4.8,5.2,.....,9.6,10.0"

```

With MPS_Slot0
    .WriteString "*RST;*CLS;STATUS:PRESET"      ' RESET AND CLEAR MODULE
    .WriteString "VOLT" & " " & Vstart        ' START RAMP AT Vstart
    .WriteString "CURR .1"                    '
    .WriteString "OUTPUT ON"                  ' ENABLE OUTPUT
    .WriteString "VOLT:MODE LIST"              ' SET TO GET VOLTAGE FROM LIST
    .WriteString "LIST:VOLT" & " " & V_StepList ' DOWNLOAD VOLTAGE POINTS
    .WriteString "LIST:DWELL" & " " & Dwell    ' DOWNLOAD 1 DWELL TIME
    .WriteString "LIST:STEP AUTO"              ' DWELL-PACED LIST
    .WriteString "INITIATE"                    ' ENABLE TRIGGER TO START LIST

End With
'
' BEFORE TRIGGERING THE MODULE, DETERMINE IF THE MODULE IS READY BY CHECKING FOR
' 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER).
'
' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
' THAT TAKE TIME WILL GIVE THE MOCULE A CHANCE TO COMPLETE PROCESSING.
'
Do
    With MPS_Slot0
        .WriteString "STATUS:OPERATION:CONDITION?"
        Value = .ReadNumber
    End With
Loop Until (Value & 32)          ' TEST FOR BIT 5 = TRUE

'
' SEND TRIGGER COMMAND TO START LIST AND GENERATE THE VOLTAGE RAMP
'
' SEND AN IMMEDIATE TRIGGER, WHICH IS ALWAYS ACTIVE. THEREFORE, IT DOES NOT NEED TO BE
' SELECTED AS A TRIGGER SOURCE.

MPS_Slot0.WriteString "TRIGGER:IMMEDIATE"

' CLOSE THE I/O SESSIONS.
MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

Exit Sub

MyError:
    txtError = Err.Description & vbCrLf
    Resume Next

End Sub

```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

#### Application 4. Providing Time-Varying Voltages

##### Overview of application

To burn-in devices using thermal or mechanical cycling/stress, cyclical time-varying voltage is provided by programming a set of voltage and dwell time points that repetitively sequence over time.

For this example, the power module must provide the repetitive waveform shown in Figure 4-1. This time-varying voltage will be applied to a hybrid IC. By continually cycling the voltage from 0 to 7 volts over a 33 second interval, the hybrid is given time to heat up and undergo thermal and mechanical stress as the welds inside the hybrid expand and contract, and then subsequently cool down.

In addition to generating the cyclical voltage, it is desirable to have the power module notify the computer should the device fail and stop the cycling. Since the module is monitoring test status, the computer is free to perform other tests.

The MPS can address this application using dwell-paced repetitive Lists. This application could be thought of as a simple power arbitrary waveform generator. To get the desired time-varying voltage, you must be able to describe the waveform in 20 discrete voltage points, with each point ranging from 10 ms to 65 seconds. This range of dwell times determines the range of frequencies (or time rate of change) of the voltage waveform to be generated.

Once the waveform has been described, it is downloaded to the module. Upon being triggered, it will repetitively generate the waveform without computer intervention.

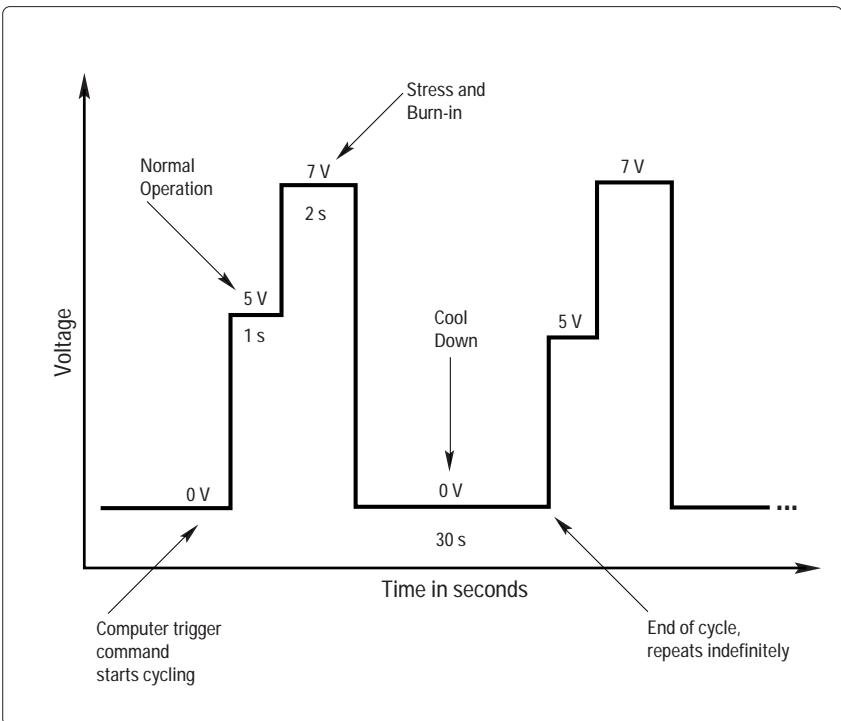


Figure 4-1. Voltage Waveform for Application #4

The module will also be set up to generate an SRQ and stop the voltage cycling of the hybrid should fail. If the hybrid fails by shorting, the module will go into CC. This change in status will cause the module to protect the DUT by disabling the output, which will stop the test and generate an SRQ. (Open circuit failures will not be detected. Since failures of this type are less likely to have destructive consequences, detection is not required.)

#### **MPS features used**

- 20-point voltage List
- Repetitive Lists
- Dwell time
- Dwell-paced Lists
- Generate an SRQ on a change in internal status
- Disable the output on a change in internal status
- Stop the List on a change in internal status
- Trigger on a GP-IB trigger command
- Overcurrent protection
- Active downprogramming

#### **Advantages/benefits of the MPS solution**

- By using Lists, the module changes its output voltage automatically, so that the computer is not devoted to reprogramming the output voltage.
- The output can change faster when dwell paced than when the computer must explicitly reprogram each change.
- Overcurrent protection can disable the output before the DUT is damaged.
- By letting each module monitor its status, the CC condition will be responded to faster than if the computer was responsible for stopping the test.
- The sequence is simpler to program (no timing loops).
- By using dwell times, the timing of each point is accurate and repeatable because the computer does not provide timing for the sequence.
- When the output is disabled, the active downprogrammers in the module output can quickly discharge the module's output capacitors and any capacitance in the DUT.

#### **Implementation details**

##### **How the MPS implements the sequence**

- The module is programmed to List mode.
- The module will execute a dwell-paced List.
- The 3 voltage points are downloaded to the module.
- The 3 dwell times are downloaded to the module.
- To begin the cycling, the module is triggered by the computer.
- The module continuously generates the voltage waveform.
- The module continuously monitors its status.
- If the module goes into CC, the overcurrent protection disables the output.
- The module generates an SRQ when the overcurrent protection occurs.

**Module set up**

- Set voltage mode to List.
- Download voltage List.
- Download dwell times.
- Set Lists to dwell paced.
- Set Lists to infinitely repeat.
- Enable status monitoring of overcurrent condition.
- Enable overcurrent protection.
- Enable SRQ generation on overcurrent protection occurrence.

**Variations on this implementation**

1. The module could be set to begin generating the waveform in response to an external or backplane TTL Trigger.
2. The module could be set to generate external triggers for each point in the List. This trigger could be routed to other instruments to synchronize external measurements to the change in voltage. (See application #7) Using this technique, parametric measurements could be made on the device during the thermal cycling.
3. Multiple modules could be programmed to cycle together in response to the computer trigger command.
4. To determine how many times the hybrid was cycled before it failed, you can use the SRQ (that was generated when the hybrid failed and the module went into CC) to timestamp the failure. The elapsed time will give the number of cycles executed.

#### 'Application #4: Providing Time-Varying Voltages

```
Private Sub CommandApp4_Click()
```

```
' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.
```

```
' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:
```

```
' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL  
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb  
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL
```

```
' USE A STATEMENT SUCH AS "Dim MPS_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT
```

```
Dim io_mgr As VisaComLib.ResourceManager  
Dim MPS_Slot0 As VisaComLib.FormattedIO488
```

```
Set io_mgr = New AgilentRMLib.SRMClS  
Set MPS_Slot0 = New VisaComLib.FormattedIO488  
Set MPS_Slot0.IO = io_mgr.Open("GPIB0::5::0::INSTR")
```

```
On Error GoTo MyError
```

```
Dim st_value As Integer ' HOLD STATUS VALUE
```

```
With MPS_Slot0
```

```
 ' INITIALIZE THE MODULE
```

```
 .WriteString "**RST;*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE
```

```
 .WriteString "VOLT 0" ' START TEST AT 0 V
```

```
 .WriteString "CURR .1" ' SET CURRENT LIMIT
```

```
 .WriteString "OUTPUT ON" ' ENABLE OUTPUT
```

```
 ' SET UP OVERCURRENT PROTECTION (OCP) AND GENERATE SRQ ON OCP TRIP
```

```
 .WriteString "CURRENT:PROTECTION:STATE ON" ' ENABLE OCP
```

```
 .WriteString "OUTPUT:PROTECTION:DELAY 0" ' NO DELAY BEFORE PROTECTION OCCURS
```

```
 .WriteString "STATUS:QUESTIONABLE:ENABLE 2" ' ENABLE DETECTION OF OC CONDITION IN THE  
' QUESTIONABLE REGISTER, WHERE OC = BIT 1 = VALUE 2.
```

```
 .WriteString "STATUS:QUESTIONABLE:PTRANSITION 2" ' ENABLES DETECTION ON POSITIVE TRANSITION, I.E., GOING INTO OC.
```

```
 .WriteString "**SRE 8" ' ENABLES THE SERVICE REQUEST REGISTER TO GENERATE
```

```
 ' AN SRQ WHEN ANY EVENT IN THE QUESTIONABLE REGISTER  
' IS ASSERTED. THE QUESTIONABLE REGISTER = BIT 3= VALUE 8.
```

```

' SET UP THE VOLTAGE LIST
'
.WriteString "VOLT:MODE LIST"
.WriteString "LIST:VOLT 5,7,0"
.WriteString "LIST:DWELL 1,2,30"
.WriteString "LIST:STEP AUTO"
.WriteString "LIST:COUNT INF"
.WriteString "INITIATE"
End With
'
' BEFORE TRIGGERING THE MODULE, DETERMINE IF IT IS READY BY CHECKING FOR
' WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER).
'
' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
' THAT TAKE TIME WILL GIVE THE MODULE A CHANCE TO COMPLETE PROCESSING.
'
Do
  With MPS_Slot0
    .WriteString "STATUS:OPERATION:CONDITION?"
    st_value = .ReadNumber
  End With
Loop Until (st_value & 32)
' TEST FOR BIT 5 = TRUE
'
' SEND TRIGGER COMMAND TO START LIST
'
' SEND IMMEDIATE TRIGGER, WHICH IS ALWAYS ACTIVE. THEREFORE, IT DOES NOT NEED TO BE SELECTED
' AS A TRIGGER SOURCE.

MPS_Slot0.WriteString "TRIGGER:IMMEDIATE"

' CLOSE THE I/O SESSIONS.
MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

Exit Sub

MyError:
  txtError = Err.Description & vbCrLf
  Resume Next

End Sub

```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

## Application 5. Providing Time-Varying Current Limiting

### Overview of application

To provide current limit protection which varies as a function of time, multiple thresholds on current limit are required. Having multiple thresholds can provide a high limit to protect the DUT during its power-up in-rush with automatic switchover to a lower limit to protect the DUT during its steady state operation.

For this example, the DUT is a printed circuit assembly. This assembly is being tested prior to installation in the end product. The module provides power to the assembly, which will undergo a functional test. The assembly has capacitors on-board, and when power is applied, the in-rush current approaches 4 A. After the capacitors charge, which takes about 500 milliseconds, the steady state current settles to 600 mA. See Figure 5-1.

The MPS can address this application using dwell-paced Lists. In this case, the List will consist of a set of current limits and dwell times, because the voltage will remain constant throughout the test.

Once power has been applied, the first current limit, which provides protection to a shorted DUT while still allowing high current in-rush to occur, will remain in effect for the dwell time. Then the current limit will switch to its next setting in the List. The result is a current limit which changes with time and provides protection as the DUT current requirements drop off to their steady state value. When the dwell time expires for the last current limit in the List, the current limit stays at this value until reprogrammed. Thus, the actual value of the last dwell time is not important. The last current List point would be the current limit for the steady state operation during the test of the DUT. See Figure 5-2 for how the MPS implements this protection.

Throughout List execution, overcurrent protection will be enabled. If at any time the module goes into CC, the output will be disabled, the test stopped, and the DUT protected.

### MPS features used

- 20-point current List
- Dwell time
- Dwell-paced Lists
- Disable the output on a change in internal status
- Stop the List on a change in internal status
- Change the voltage on trigger
- Trigger on a GP-IB trigger command
- Overcurrent protection
- Active downprogramming



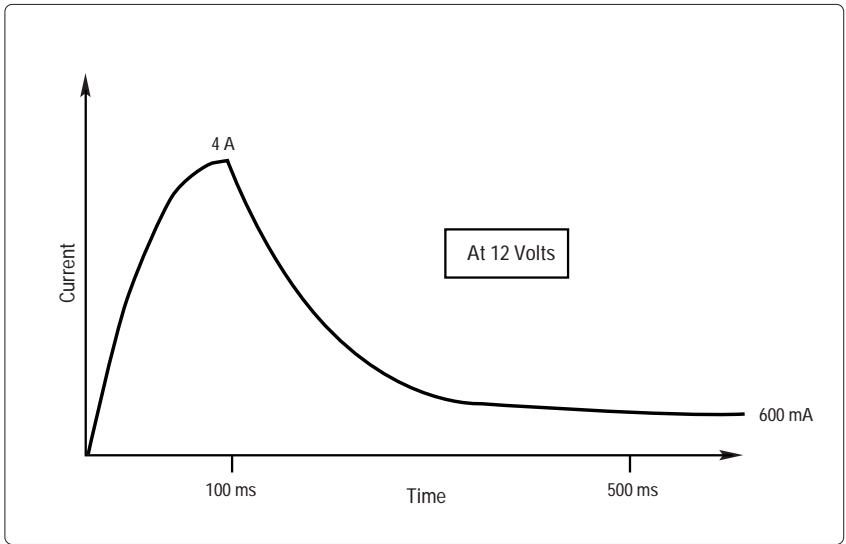


Figure 5-1. Typical DUT Current vs Time

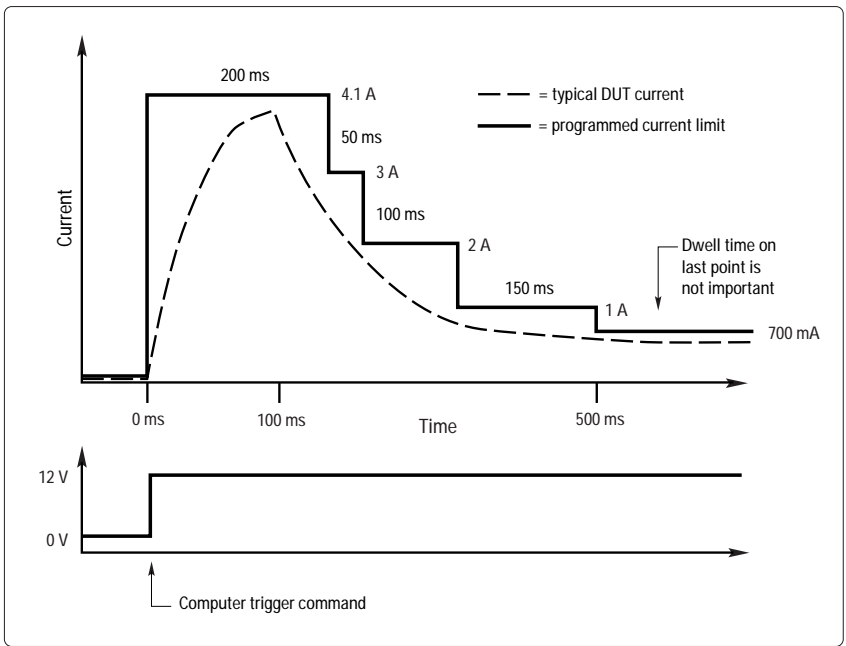


Figure 5-2. Desired Current Limit vs Time

### **Advantages/benefits of the MPS solution**

- By using Lists, the module changes its current limit automatically, so that the computer is not devoted to reprogramming the current limit.
- The output can change faster when dwell paced than when the computer must explicitly reprogram each change.
- Overcurrent protection can disable the output before the DUT is damaged.
- By letting the modules monitor status, the CC condition will be responded to faster than if the computer was responsible for stopping the test.
- The sequence is simpler to program (no timing loops).
- By using dwell times, the timing of each point is accurate and repeatable because the computer does not provide timing for the sequence.
- When the output is disabled, the active downprogrammers in the module output can quickly discharge the module's output capacitors and any capacitance in the DUT.

### **Implementation details**

#### **How the MPS implements the sequence.**

- The module is programmed to current List mode.
- The module will execute a dwell-paced current List.
- The current limit List points are downloaded to the module.
- The dwell times are downloaded to the module.
- To begin powering the DUT, the module is triggered by the computer. This one trigger causes the current List to begin executing and the voltage to go to its programmed value.
- The module steps through the current limit List.
- The module continuously monitors its status.
- If the modules goes into CC, the overcurrent protection disables the output.

#### **Module set up**

- Set the current mode to List.
- Download current List.
- Download the dwell times.
- Set the List to be dwell paced.
- Enable overcurrent protection.
- The initial voltage setting is 0 V.
- The module listens for the computer to send a trigger command.
- Upon receipt of the trigger command, the module goes to 12 V.
- Also upon receipt of the trigger command, the module begins executing its current limit List.

### **Variations on this implementation**

1. The module could be set to begin applying power in response to an external or backplane TTL Trigger.
2. Multiple modules could be programmed to cycle together in response to the computer trigger command. Each module could have unique current limits, voltage settings, and dwell times.
3. The module could be set to generate an SRQ if the overcurrent protection disables the output.
4. The module could be set to generate an external or backplane TTL Trigger if the overcurrent protection disables the output.
5. The in-rush current can be controlled using the current limit settings of the current List. Instead of setting the current limit slightly above 4 A, it could be set at a much lower value. This would limit the in-rush current to the value in the List. It would take longer to charge the capacitors on the assembly, but the in-rush condition would be controlled. In this variation, the overcurrent protection could not be used, because you want the module to be in CC.

### 'Application #5: Providing Time-Varying Current Limiting

```
Private Sub CommandApp5_Click()
```

```
' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.
```

```
' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:
```

```
' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL  
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb  
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL
```

```
' USE A STATEMENT SUCH AS "Dim MPS_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT
```

```
Dim io_mgr As VisaComLib.ResourceManager  
Dim MPS_Slot0 As VisaComLib.FormattedIO488
```

```
Set io_mgr = New AgilentRMLib.SRMCIs  
Set MPS_Slot0 = New VisaComLib.FormattedIO488  
Set MPS_Slot0.IO = io_mgr.Open("GPIB0::5::0::INSTR")
```

```
On Error GoTo MyError
```

```
Dim C_limit, Dwell As String
```

```
C_limit = "4.1, 3.0, 2.0, 1.0, 0.7"           ' CURRENT LIMIT DATA  
Dwell = "0.2, 0.05, 0.1, 0.15, 0.1"         ' DWELL TIME DATA
```

```
Dim st_value As Integer                       ' HOLD STATUS VALUE
```

```
With MPS_Slot0  
    .WriteString "**RST;*CLS;STATUS:PRESET"    ' RESET AND CLEAR MODULE  
    .WriteString "VOLT 0"                     ' START TEST AT 0 V  
    .WriteString "OUTPUT ON"                  ' ENABLE OUTPUT  
    .WriteString "CURRENT:PROTECTION:STATE ON" ' ENABLE OCP  
    .WriteString "OUTPUT:PROTECTION:DELAY 0"  ' NO DELAY BEFORE PROTECTION OCCURS  
    .WriteString "CURRENT:MODE LIST"          ' SET TO GET CURRENT FROM LIST  
    .WriteString "LIST:CURRENT " & C_limit    ' DOWNLOAD CURRENT POINTS  
    .WriteString "LIST:DWELL " & Dwell        ' DOWNLOAD DWELL TIMES  
    .WriteString "LIST:STEP AUTO"            ' DWELL-PACED LIST  
    .WriteString "VOLT:TRIGGERED 12"         ' GO TO 12 V WHEN TRIGGERED  
    .WriteString "INITIATE"                  ' ENABLE TRIGGER TO START LIST AND APPLY 12 V
```

```

End With
'
' BEFORE TRIGGERING THE MODULE, DETERMINE IF IT IS READY BY CHECKING FOR
' 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER).
'
' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
' THAT TAKE TIME WILL GIVE THE MODULE A CHANCE TO COMPLETE PROCESSING.
'
Do
  With MPS_Slot0
    .WriteString "STATUS:OPERATION:CONDITION?"
    st_value = .ReadNumber
  End With
Loop Until (st_value & 32)          ' TEST FOR BIT 5 = TRUE
'
' SEND TRIGGER COMMAND TO START LIST AND APPLY 12V
' SEND IMMEDIATE TRIGGER, WHICH IS ALWAYS ACTIVE. THEREFORE, IT DOES NOT NEED TO BE
' SELECTED AS A TRIGGER SOURCE.

MPS_Slot0.WriteString "TRIGGER:IMMEDIATE"

' CLOSE THE I/O SESSIONS.
MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

Exit Sub

MyError:
  txtError = Err.Description & vbCrLf
  Resume Next

End Sub

```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

**Application 6.**  
**Output Sequencing Paced by the Computer**

**Overview of application**

When performing bias supply margin testing, throughput can be maximized by eliminating the command processing time associated with reprogramming all outputs for each set of limit conditions. Instead, multiple sets of bias limit conditions can be downloaded to the power modules during test system initialization. During the testing, the computer can use a single command to simultaneously signal all power modules to step through each test condition.

In this example, the DUT requires +5 V and ±12 V. The DUT is tested to ensure proper operation at marginal supply voltages. The margin specified is ±5% of nominal voltage. At each of the combinations given below, the computer first sets up the three modules and makes a measurement on the DUT. The combinations to be tested are:

When conducting this test, the modules will need to be reprogrammed 21 times and seven measurements made. The command processing time could slow down this test.

The MPS can be used to increase throughput. By downloading all of the combinations into the three modules, each setting can be quickly stepped through by triggering all modules to change to their next voltage setting and then taking a measurement from the DUT. This permits testing without command processing overhead.

**MPS features used**

- 20-point voltage List
- Trigger-paced Lists
- Trigger in/out from MPS mainframe backplane TTL Trigger
- Trigger on a GP-IB trigger command

**Advantages/benefits of the MPS solution**

- By using Lists, the module changes its voltage without delays due to processing the command to change the output voltage.
- By using triggers, all three outputs can be changed with one command.
- The computer loop to change the settings and take a measurement is simplified, because you do not have to explicitly reprogram each module output. Instead, the loop becomes “Trigger” and “Measure”.

Nominal 5 V	Nominal +12 V	Nominal -12 V
4.75 V	12 V	- 12 V
5 V	12 V	- 12 V
5.25 V	12 V	- 12 V
5 V	11.4 V	- 12 V
5 V	12.6 V	- 12 V
5 V	12 V	- 11.4 V
5 V	12 V	- 12.6 V

## Implementation details

### How the MPS implements the sequence

The following steps are performed for each point in the List:

- The computer sends a trigger command to the first module.
- The first module simultaneously sends a backplane TTL Trigger to the other two modules and goes to its next List point.
- The second module receives the backplane TTL Trigger and immediately goes to its next List point.
- The third module receives the backplane TTL Trigger, immediately goes to its next List point.
- The computer gets a measurement from the measurement instrument.

### MPS set up

#### *Module in slot 0:*

- The module is connected to +5 V on the DUT.
- The initial voltage setting is 0 V.
- Set the voltage mode to List.
- Download the voltage List.
- Set the List to be trigger paced.

- The module listens for the computer to send a trigger command.
- Upon receipt of the trigger command, the module outputs its next List point.
- Also upon receipt of the trigger command, the module generates a backplane TTL Trigger.

#### *Module in slot 1:*

- The module is connected to +12 V on the DUT.
- The initial voltage setting is 0 V.
- Set the voltage mode to List.
- Download the voltage List.
- Set the List to be trigger paced.
- The module listens for a backplane TTL Trigger.
- Upon receipt of a trigger, the module goes to its next List point.

#### *Module in slot 2:*

- The module is connected to -12 V on the DUT.
- The initial voltage setting is 0 V.
- Set the voltage mode to List.
- Download the voltage List.
- Set the List to be trigger paced.
- The module listens for a backplane TTL Trigger.
- Upon receipt of a trigger, the module goes to its next List point.

### Variations on this implementation

1. A current List could also have been executed by the module so that for each voltage point, a corresponding current limit could be programmed.
2. Overcurrent protection could be enabled to protect a faulty DUT.
3. The module could generate an SRQ when it finishes changing voltage for each point in the List based on the STC (Step Completed) status bit, which indicates when the module has completed executing the next point in the List. The SRQ could tell the computer to get a measurement from the measurement instrument.
4. The module could be told to output its next List point in response to an external or backplane TTL Trigger. (See next application.)

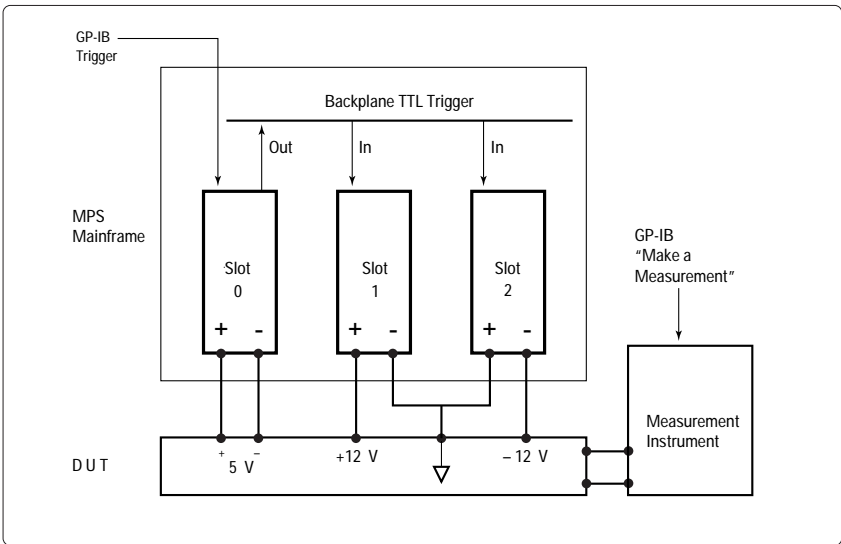


Figure 6-1. Block Diagram of Application #6

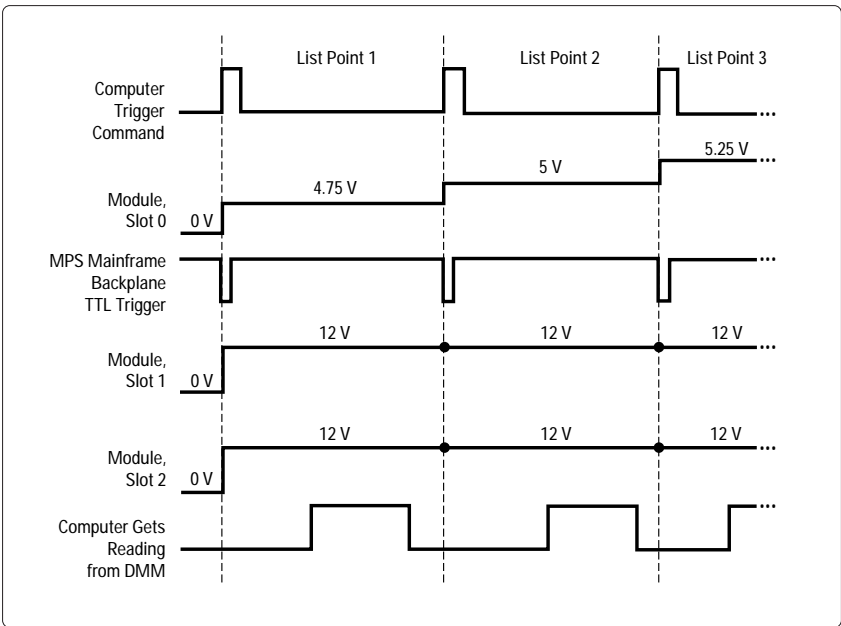


Figure 6-2. Timing Diagram of Application #6

**'Application #6: Output Sequencing Paced by the Computer**

Private Sub CommandApp6\_Click()

' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.

' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:

' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL  
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb  
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL

' USE A STATEMENT SUCH AS "Dim MPS\_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS\_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT

Dim io\_mgr As VisaComLib.ResourceManager  
Dim MPS\_Slot0 As VisaComLib.FormattedIO488  
Dim MPS\_Slot1 As VisaComLib.FormattedIO488  
Dim MPS\_Slot2 As VisaComLib.FormattedIO488

Set io\_mgr = New AgilentRMLib.SRMClis  
Set MPS\_Slot0 = New VisaComLib.FormattedIO488  
Set MPS\_Slot1 = New VisaComLib.FormattedIO488  
Set MPS\_Slot2 = New VisaComLib.FormattedIO488

Set MPS\_Slot0.IO = io\_mgr.Open("GPIB0::5::0::INSTR")  
Set MPS\_Slot1.IO = io\_mgr.Open("GPIB0::5::1::INSTR")  
Set MPS\_Slot2.IO = io\_mgr.Open("GPIB0::5::2::INSTR")

On Error GoTo MyError

Dim Plus\_5v, Plus\_12v, Minus\_12v As String

Plus\_5v = "4.75, 5, 5.25, 5, 5, 5, 5"                   ' THESE ARE THE BIAS  
Plus\_12v = "12, 12, 12, 11.4, 12.6, 12, 12"       ' SUPPLY LIMIT CONDITIONS  
Minus\_12v = "12, 12, 12, 12, 12, 11.4, 12.6"     ' TO BE TESTED

Num\_test\_steps = 7                                   ' NUMBER OF BIAS SUPPLY LIMIT COMBINATIONS  
Dwell = 0.01   ' SECONDS OF DWELL TIME

Dim st\_value As Integer                             ' HOLD STATUS VALUE



```

' SET UP MODULE IN SLOT 0 AS +5 V BIAS SUPPLY -----
'
With MPS_Slot0
  .WriteString "**RST;*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE
  .WriteString "VOLT 0" ' START AT 0 V
  .WriteString "OUTPUT ON" ' ENABLE OUTPUT
  .WriteString "VOLTAGE:MODE LIST" ' SET TO GET VOLTAGE FROM LIST
  .WriteString "LIST:VOLTAGE " & Plus_5v ' DOWNLOAD VOLTAGE LIST POINTS
  .WriteString "LIST:DWELL " & Dwell ' DOWNLOAD 1 DWELL TIME (ASSUMES SAME FOR ALL POINTS)
  .WriteString "LIST:STEP ONCE" ' EXECUTE 1 LIST POINT PER TRIGGER
  .WriteString "TRIGGER:SOURCE BUS" ' TRIGGER SOURCE IS GPIB 'BUS'
  .WriteString "OUTPUT:TTLTRG:SOURCE BUS" ' GENERATE BACKPLANE TTL TRIGGER WHEN GPIB 'BUS' TRIGGER IS RECEIVED
  .WriteString "OUTPUT:TTLTRG:STATE ON" ' ENABLE TTL TRIGGER DRIVE
  .WriteString "INITIATE" ' ENABLE RESPONSE TO TRIGGER
End With
'
' SET UP MODULE IN SLOT 1 AS +12 V BIAS SUPPLY -----
'
With MPS_Slot1
  .WriteString "**RST;*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE
  .WriteString "VOLT 0" ' START AT 0 V
  .WriteString "OUTPUT ON" ' ENABLE OUTPUT
  .WriteString "VOLT:MODE LIST" ' SET TO GET VOLTAGE FROM LIST
  .WriteString "LIST:VOLTAGE " & Plus_12v ' DOWNLOAD VOLTAGE LIST POINTS
  .WriteString "LIST:DWELL " & Dwell ' DOWNLOAD 1 DWELL TIME (ASSUMES SAME FOR ALL POINTS)
  .WriteString "LIST:STEP ONCE" ' EXECUTE 1 LIST POINT PER TRIGGER
  .WriteString "TRIGGER:SOURCE TTLTRG" ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
  .WriteString "INITIATE" ' ENABLE RESPONSE TO TRIGGER
End With
'
' SET UP MODULE IN SLOT 2 AS -12 V BIAS SUPPLY -----
'
With MPS_Slot2
  .WriteString "**RST;*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE
  .WriteString "VOLT 0" ' START AT 0 V"
  .WriteString "OUTPUT ON" ' ENABLE OUTPUT
  .WriteString "VOLT:MODE LIST" ' SET TO GET VOLTAGE FROM LIST
  .WriteString "LIST:VOLTAGE " & Minus_12v ' DOWNLOAD VOLTAGE LIST POINTS
  .WriteString "LIST:DWELL " & Dwell ' DOWNLOAD 1 DWELL TIME (ASSUMES SAME FOR ALL POINTS)
  .WriteString "LIST:STEP ONCE" ' EXECUTE 1 LIST POINT PER TRIGGER
  .WriteString "TRIGGER:SOURCE TTLTRG" ' TRIGGER SOURCE IS BACKPLANE TTL TRIGGER
  .WriteString "INITIATE" ' ENABLE RESPONSE TO TTL TRIGGER
End With
'
' BEFORE TRIGGERING THE MODULES, DETERMINE IF THE MODULES ARE READY BY CHECKING FOR
' 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER). IF THE LAST MODULE PROGRAMMED
' IS READY THEN SO ARE THE OTHERS, SO JUST CHECK SLOT 2.
'
' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
' THAT TAKE TIME WILL GIVE THE MODULES A CHANCE TO COMPLETE PROCESSING.
'

```

```

Do

    MPS_Slot2.WriteString "STATUS:OPERATION:CONDITION?"
    st_value = MPS_Slot2.ReadNumber

Loop Until (st_value & 32)      ' TEST FOR BIT 5 = TRUE
'
' GENERATE A TRIGGER AND MAKE A MEASUREMENT FOR EACH TEST CONDITION
'
For Loop_count = 1 To Num_test_steps

    MPS_Slot0.WriteString "*TRG"      ' SEND GPIB BUS TRIGGER
    Get_Measurement

Next Loop_count

' CLOSE THE I/O SESSIONS.
MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

MPS_Slot1.FlushRead
MPS_Slot1.FlushWrite False
MPS_Slot1.IO.Close
Set MPS_Slot1 = Nothing

MPS_Slot2.FlushRead
MPS_Slot2.FlushWrite False
MPS_Slot2.IO.Close
Set MPS_Slot2 = Nothing

Exit Sub

MyError:

    txtError = Err.Description & vbCrLf
    Resume Next

End Sub

Private Sub Get_Measurement()
'
' THIS IS JUST TO SHOW YOU WHERE YOU WOULD ADD CODE TO GET DATA FROM THE MEASUREMENT INSTRUMENT.
' THE MEASUREMENT MUST TAKE LONGER THAN THE PROGRAMMED DWELL TIME OR YOU WILL MISS TRIGGERS.
'
End Sub

```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

## Application 7. Output Sequencing without Computer Intervention

### Overview of application

When characterizing devices, the DUT's performance is measured over a range of power supply voltages. This test can be performed without computer intervention by using hardware signals from the measurement instrument to cause the power module to sequence to the next voltage in a preprogrammed List. By buffering these readings in the measurement instrument, the entire test can be executed without computer involvement. For characterizations that require long measurement times, the computer is free to do other tasks. For characterizations that must execute at hardware speeds, the computer is not involved and will not slow down the test loop.

In this example, the power module must apply 8 to 14 volts (in 13 0.5-volt increments) to an automotive engine sensor. The module varies the bias voltage to the engine sensor and the sensor's output is measured to characterize its performance over the range of possible "battery voltages". The sensor output is measured by a DMM that has an internal buffer and stores each reading.

By combining Lists and trigger capabilities, the MPS can be used to address this application. The module can be programmed to use its triggering capabilities to the fullest extent. Each time the module executes the next step in its List and changes voltage, the module will generate an external trigger. The external trigger will cause the DMM, equipped with an external trigger input, to take and store a reading. The DMM, also equipped with a "Measurement Complete" output, sends its output trigger signal to the module to tell the module to go to its next List point. Effectively, the module and the DMM "handshake", so that the two function at hardware speeds without computer intervention.

When the test is complete, either device can signal the computer to get the data from the DMM. For the purpose of this example, the module will generate an SRQ when the last List point has been executed. This is indicated by the OPC (Operation Complete) bit in the status register.

Another detail that needs attention is timing. The DUT may require some settling time before the DMM is told to take a reading. The module's dwell time can be used to do this. The STC (Step Complete) status signal indicates when the point has been executed and its dwell time has expired. The dwell time is programmed to be the engine sensor's settling time. The external trigger is generated when STC is asserted. Thus, the DMM will not be triggered until the dwell time has expired and the sensor's output has settled.

This type of self-paced test execution is useful in two situations. When the test must execute very fast, there is no time for the computer to be involved in each iteration of the test loop. Therefore, the test must execute without computer intervention. The second situation is when the test is very long. For example, if the measurement instrument took 1 minute to make each measurement, the test would take 13 minutes to execute. The computer is not used efficiently if it is idle while waiting for each measurement loop, so it would be best to have the computer executing another task. Without self-pacing, you would need to develop interrupt driven software that stops every 1 minute to take a reading. By letting the module and the DMM run on their own, code development is much simpler and computer resources are used more efficiently.

#### **MPS features used**

- 20-point voltage List
- Dwell time
- Trigger-paced Lists
- Generate an SRQ on a change in internal status
- Generate a trigger on a change in internal status
- Trigger in/out from MPS mainframe backplane TTL Trigger
- Trigger on a GP-IB trigger command

#### **Advantages/benefits of the MPS solution**

- The entire test executes without computer involvement, the command processing time is eliminated from the test loop.
- The entire test executes without computer involvement, so the computer can perform other tasks while the test executes.
- Software development is simplified; you do not need to write a test loop because the module and the DMM are running on their own.
- By using dwell times, the trigger out signal can be sent at the correct time, which permits the DUT to settle before a reading is taken.

#### **Implementation details**

##### **How the MPS implements the sequence**

- The module listens for the computer to send a trigger command.
- Upon receipt of the trigger command, the module outputs its first List point.
- After the dwell time expires, the STC is asserted and the module generates an external trigger.
- The DMM receives the external trigger, takes and stores a reading.
- The DMM generates a "Measurement Complete" output signal when it's done.
- The module receives the DMM output signal as an external trigger in.
- Also upon receipt of the trigger in, the module outputs its next List point.
- The process repeats for each List point.
- After the last List point has executed, the module generates SRQ, telling the computer the test has completed.

**MPS set up**

- Set the voltage mode to List.
- Download the voltage List.
- Download the dwell time List.
- Set the List to be trigger paced.
- Set the trigger source to external trigger. (Note that the computer trigger command “TRIGGER: IMMEDIATE” is always active, even if the external trigger is the selected source.)
- Set the module to generate a backplane TTL Trigger on STC. This backplane TTL Trigger drives external trigger out. Set the module to generate SRQ on OPC.

**Variations on this implementation**

1. A current List could also have been executed by the module so that for each voltage point, a corresponding current limit could be programmed.
2. Overcurrent protection could be enabled to protect a faulty engine sensor.
3. If the DMM does not have an internal buffer, the computer could take a reading on each iteration of the test loop (see previous application).

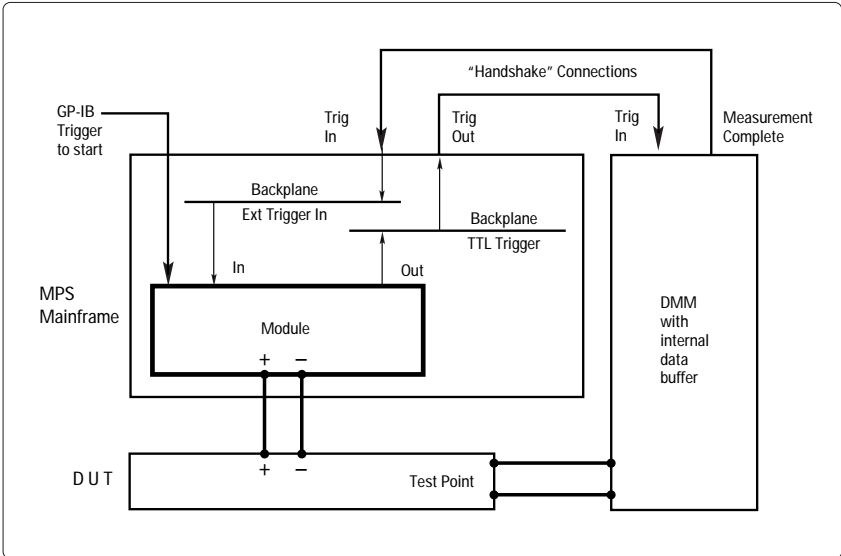


Figure 7-1. Block Diagram of Application #7

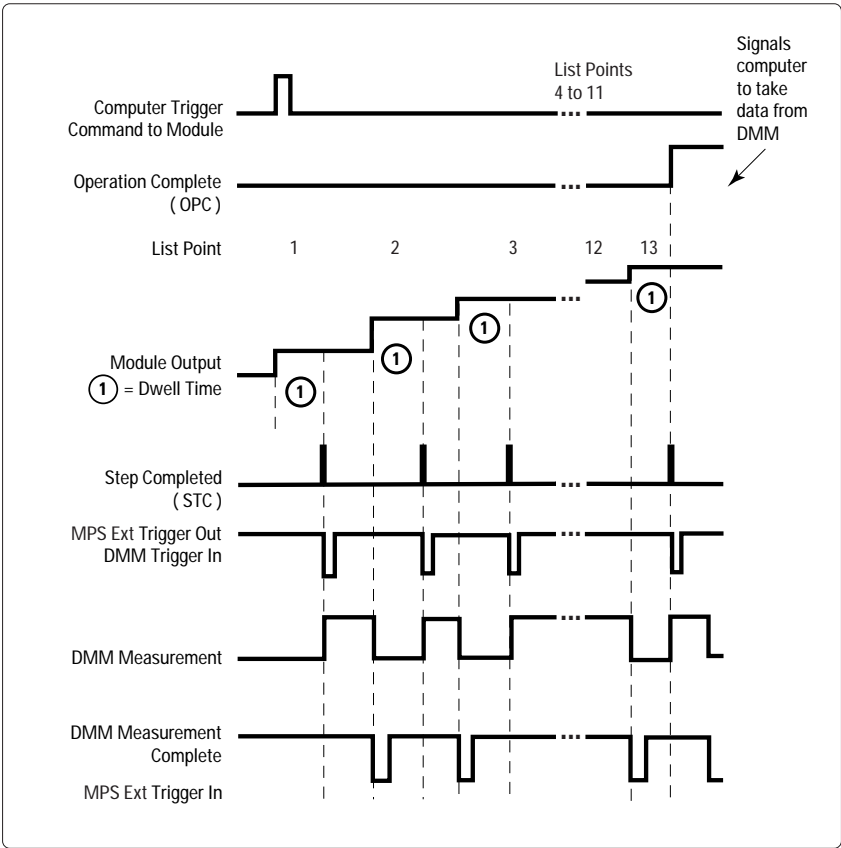


Figure 7-2. Timing Diagram of Application #7

### 'Application #7: Output Sequencing Without Computer Intervention

```
'  
Implements VisaComLib.IEventHandler  
Dim MPS_Slot0 As VisaComLib.FormattedIO488 'THIS REPRESENTS THE MPS MODULE  
Dim SRQ As VisaComLib.IEventManager      'USE THIS TO ENABLE THE EVENT HANDLER  
Private Sub CommandApp7_Click()  
  
' THIS IS AN EXAMPLE PROGRAM INTENDED FOR USE WITH MICROSOFT VISUAL BASIC 6.0 AND THE AGILENT  
' VISA COM OBJECTS.  
'  
' TO USE THE IO OBJECT IN A VISUAL BASIC PROJECT, SET THE REFERENCE TO INCLUDE THE LIBRARIES  
' IN THE PROJECT/REFERENCES MENU:  
' "AGILENT VISA COM RESOURCE MANAGER 1.0", CORRESPONDS TO AgtRM.DLL  
' "VISA COM 1.0 TYPE LIBRARY", CORRESPONDS TO SISACOM.tlb  
' "VISA COM 488.2 FORMATTED I/O 1.0", CORRESPONDS TO BasicFormattedIO.DLL  
'  
' USE A STATEMENT SUCH AS "Dim MPS_Slot As VisaComLib.FormattedIO488" TO CREATE THE  
' FORMATTED I/O REFERENCE AND USE "Set MPS_Slot = New VisaComLib.FormattedIO488" TO CREATE  
' THE ACTUAL OBJECT  
  
Dim io_mgr As VisaComLib.ResourceManager  
  
Set io_mgr = New AgilentRMLib.SRMCIs  
Set MPS_Slot0 = New VisaComLib.FormattedIO488  
Set MPS_Slot0.IO = io_mgr.Open("GPIB0::5::0::INSTR") 'SET IO ADDRESS  
  
On Error GoTo MyError  
  
Dim Vlist As String  
Dim st_value As Integer          ' HOLD STATUS VALUE  
  
Vlist = "8, 8.5, 9, 9.5, 10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14" ' VOLTAGE LIST POINTS  
  
With MPS_Slot0  
    .WriteString "**RST;*CLS;STATUS:PRESET" ' RESET AND CLEAR MODULE  
    .WriteString "VOLT 0" ' START AT 0 V  
    .WriteString "CURR 1" ' SET CURRENT LIMIT  
    .WriteString "OUTPUT ON" ' ENABLE OUTPUT  
    .WriteString "VOLT:MODE LIST" ' SET TO GET VOLTAGE FROM LIST  
    .WriteString "LIST:VOLT" & Vlist ' DOWNLOAD VOLTAGE LIST POINTS  
    .WriteString "LIST:DWELL 0.05" ' DOWNLOAD 1 DWELL POINT (ASSUMES SAME FOR ALL POINTS)  
    ' USE A 50 ms SETTling TIME AS THE DWELL TIME  
    .WriteString "LIST:STEP ONCE" ' EXECUTE 1 POINT PER TRIGGER  
  
    .WriteString "**ESE 1" ' ENABLES DETECTION OF OPC IN THE STANDARD EVENT REGISTER.  
    ' OPC = BIT 0 = VALUE 1 OF THE STANDARD EVENT REGISTER.  
    .WriteString "**SRE 32" ' ENABLES THE SERVICE REQUEST REGISTER TO GENERATE AN SRQ WHEN  
    ' ANY EVENT IN THE STANDARD EVENT REGISTER IS ASSERTED.  
    ' THE STANDARD EVENT REGISTER = BIT 5 = VALUE 32.  
  
    .WriteString "OUTPUT:TTLTRG:STATE ON" ' ENABLE BACKPLANE TTL TRIGGER DRIVE  
    .WriteString "OUTPUT:TTLTRG:SOURCE LINK" ' WHEN THE MODULE INDICATES SIC (STEP COMPLETED),  
    .WriteString "OUTPUT:TTLTRG:LINK 'STC'" ' GENERATE A BACKPLANE TTL TRIGGER  
    .WriteString "TRIGGER:SOURCE EXTERNAL" ' USE EXTERNAL TRIGGER IN BNC AS TRIGGER SOURCE  
    .WriteString "INITIATE" ' ENABLE RESPONSE TO TRIGGER  
    .WriteString "**OPC" ' TELLS MODULE TO ASSERT OPC (OPERATION COMPLETE)  
    ' WHEN IT COMPLETES THE LIST. OPC GENERATES SRQ.  
  
End With
```

```

Set SRQ = MPS_Slot0.IO          ' SET SRQ TO THE SESSION AND
SRQ.InstallHandler EVENT_SERVICE_REQ, Me
SRQ.EnableEvent EVENT_SERVICE_REQ, EVENT_HNDLR ' ENABLE THE SRQ
'
' BEFORE TRIGGERING THE MODULE, DETERMINE IF IT IS READY BY CHECKING FOR
' 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER).
'
' YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
' CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
' THAT TAKE TIME WILL GIVE THE MODULE A CHANCE TO COMPLETE PROCESSING.
'
Do
  With MPS_Slot0
    .WriteString "STATUS:OPERATION:CONDITION?"
    st_value = .ReadNumber
  End With
Loop Until (st_value & 32)
'
' SEND TRIGGER COMMAND TO START LIST AND APPLY 12V
' SEND IMMEDIATE TRIGGER, WHICH IS ALWAYS ACTIVE. THEREFORE, IT DOES NOT NEED TO BE
' SELECTED AS A TRIGGER SOURCE.

MPS_Slot0.WriteString "TRIGGER:IMMEDIATE"

' CLOSE THE I/O SESSIONS.
MPS_Slot0.FlushRead
MPS_Slot0.FlushWrite False
MPS_Slot0.IO.Close
Set MPS_Slot0 = Nothing

MyError:

  txtError = Err.Description & vbCrLf
  Resume Next

End Sub
Private Sub mySrTask()

  MPS_Slot0.WriteString "**ESR?"          ' THIS CLEARS INTERRUPT FROM OPC EVENT.

  ' ADD LINES HERE TO READ THE DATA BUFFER FROM THE DMM.

  ' INTERACTIVE MESSAGE FOR DEBUG.
  txtData.Select = vbCrLf & "mySrTask fired, getting data " & Time() & vbCrLf

End Sub
Private Sub IEHandler_HandleEvent(ByVal vi As VisaComLib.IEventManager, ByVal SRQevent As VisaComLib.IEvent, ByVal userHandle As Long)

  mySrTask          ' PASS CONTROL TO YOUR INTERRUPT ROUTINE.

End Sub

```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)



## Appendix

This appendix contains a sample program listing in HP BASIC for application #3. This example program was chosen as representative of all application programs because it shows how to:

- Address the power module.
- Write strings to the power module.
- Write real arrays to the power module.
- Receive real numbers from the power module.

The six other application programs all use a subset of these functions.

```
10 ! APPLICATION #3: CONTROLLING VOLTAGE RAMP UP AT TURN ON
20 ! PROGRAM LISTING IN HP BASIC FOR APPLICATION #3
30 ! PROGRAM: APP_3
40 ASSIGN @Slot0 TO 70500 ! SELECT CODE 7, MAINFRAME ADDRESS 05, SLOT 00
50 !
60 OPTION BASE 1
70 DIM V-step(20) ! ARRAY TO HOLD THE VOLTAGE RAMP STEPS
80 Vstart=2 ! START VOLTAGE FOR RAMP
90 Vstop=10 ! STOP VOLTAGE FOR RAMP
100 Ramp_time=.5 ! SECONDS TO CHANGE FROM Vstart TO Vstop
110 Dwell=Ramp_time/19 ! IN SECONDS
120 !
130 ! SINCE THE OUTPUT STAYS AT THE LAST VOLTAGE POINT AFTER ITS DWELL TIME EXPIRES, THE DWELL TIME OF THE
140 ! LAST POINT IS NOT PART OF THE TRANSITION TIME. THEREFORE, DIVIDE THE TOTAL TIME BY 19 POINTS, NOT 20.
150 ! ALSO, YOU ONLY NEED TO DOWNLOAD 1 DWELL TIME. IF THE MODULE RECEIVES ONLY 1 DWELL TIME, IT ASSUMES
160 ! YOU WANT THE SAME DWELL TIME FOR EVERY POINT IN THE LIST.
170 !
180 FOR I=1 To 20
190 V-Step(I)=Vstart+(((Vstop-Vstart)/20)*I) ! CALCULATES VOLTAGE LIST POINTS
200 NEXT I
210 !
220 OUTPUT @Slot0;"*RST;*CLS;STATUS:PRESET" ! RESET AND CLEAR MODULE
230 OUTPUT @Slot0;"VOLT";Vstart ! START RAMP AT Vstart
240 OUTPUT @Slot0;"CURR .1"
250 OUTPUT @Slot0;"OUTPUT ON" ! ENABLE OUTPUT
260 OUTPUT @Slot0;"VOLT:MODE LIST" ! SET TO GET VOLTAGE FROM LIST
270 OUTPUT @Slot0;"LIST:VOLT";V_step(*) ! DOWNLOAD VOLTAGE POINTS
280 OUTPUT @Slot0;"LIST:DWELL";Dwell ! DOWNLOAD 1 DWELL TIME
290 OUTPUT @Slot0;"LIST:STEP AUTO" ! DWELL-PACED LIST
300 OUTPUT @Slot0;"INITIATE" ! ENABLE TRIGGER TO START LIST
310 !
320 ! BEFORE TRIGGERING THE MODULE, DETERMINE IF IT IS READY BY CHECKING FOR
330 ! 'WAITING FOR TRIGGER' (BIT 5 OF THE OPERATION STATUS REGISTER).
340 !
350 ! YOU COULD ELIMINATE THIS STEP BY SIMPLY INSERTING A PAUSE IN THE PROGRAM. HOWEVER, BY
360 ! CHECKING THE INSTRUMENT STATUS, YOU CAN AVOID TIMING PROBLEMS. ALSO, ANY OTHER OPERATIONS
370 ! THAT TAKE TIME WILL GIVE THE MODULE A CHANCE TO COMPLETE PROCESSING.
380 !
390 REPEAT
400 OUTPUT @Slot0;"STATUS:OPERATION:CONDITION?"
410 ENTER @Slot0;Condition-data
420 UNTIL BIT(Condition-data,5) ! TEST FOR BIT 5 = TRUE
430 !
440 ! SEND TRIGGER COMMAND TO START LIST AND GENERATE THE VOLTAGE RAMP
450 !
460 OUTPUT @Slot0;"TRIGGER:IMMEDIATE" ! THIS IS AN IMMEDIATE TRIGGER, WHICH IS ALWAYS ACTIVE.
470 ! THEREFORE, IT DOES NOT NEED TO BE SELECTED AS A TRIGGER SOURCE
480 !
490 END
```

To download this example program, see: [www.agilent.com/find/66000PN](http://www.agilent.com/find/66000PN)

**Agilent Technologies' Test and Measurement Support, Services, and Assistance**

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

**Our Promise**

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

**Your Advantage**

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.



**Agilent Email Updates**

[www.agilent.com/find/emailupdates](http://www.agilent.com/find/emailupdates)

Get the latest information on the products and applications you select.

**Agilent T&M Software and Connectivity**

Agilent's Test and Measurement software and connectivity products, solutions and developer network allows you to take time out of connecting your instruments to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit [www.agilent.com/find/connectivity](http://www.agilent.com/find/connectivity) for more information.

**By internet, phone, or fax, get assistance with all your test & measurement needs**

**Online assistance:**  
[www.agilent.com/find/assist](http://www.agilent.com/find/assist)

**Phone or Fax**

**United States:**  
(tel) 800 452 4844

**Canada:**  
(tel) 877 894 4414  
(fax) 905 282 6495

**China:**  
(tel) 800 810 0189  
(fax) 800 820 2816

**Europe:**  
(tel) (31 20) 547 2323  
(fax) (31 20) 547 2390

**Japan:**  
(tel) (81) 426 56 7832  
(fax) (81) 426 56 7840

**Korea:**  
(tel) (82 2) 2004 5004  
(fax) (82 2) 2004 5115

**Latin America:**  
(tel) 305 269 7500  
(fax) 305 269 7599

**Taiwan:**  
(tel) 0800 047 866  
(fax) 0800 286 331

**Other Asia Pacific Countries:**  
(tel) (65) 6375 8100  
(fax) (65) 6836 0252  
Email: [tm\\_asia@agilent.com](mailto:tm_asia@agilent.com)

Product specifications and descriptions in this document subject to change without notice.  
© Agilent Technologies, Inc. 2003  
Printed in the USA April 28, 2003  
5988-2800EN



**Agilent Technologies**