Agilent E5022A/B and E5023A Hard Disk Read/Write Test System

UDS Programming Manual

5th Edition

Software Revision
This manual applies to the system which has
the software revision B.02.70 and above

Agilent Technologies

Part No. E5023-90012
January 2003

Printed in Japan

Notices

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

Agilent Technologies Japan, Ltd.

Component Test PGU Kobe

1-3-2, Murotani, Nishi-Ku, Kobe-shi, Hyogo, 651-2241 Japan

Manual Printing History

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

April 2000	1st Edition
June 2000	2nd Edition
October 2000	3rd Edition
June 2001	4th Edition (part number: E5023-90002)
January 2003	5th Edition (part number: E5023-90012)

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales and Service Office. Addresses are provided at the back of this manual.

Sample Program

A sample program is installed in the PC. The sample program list is described in the programming manual.

The customer shall have the personal, non-transferable rights to use, copy, or modify SAMPLE PROGRAMS in this manual for the Customer's internal operations. The customer shall use the SAMPLE PROGRAMS solely and exclusively for their own purpose and shall not license, lease, market, or distribute the SAMPLE PROGRAMS or modification of any part thereof.

Agilent Technologies shall not be liable for the quality, performance, or behavior of the SAMPLE PROGRAMS. Agilent Technologies especially disclaims that the operation of the SAMPLE PROGRAMS shall be uninterrupted or error free. The SAMPLE PROGRAMS are provided AS IS.

AGILENT TECHNOLOGIES DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Agilent Technologies shall not be liable for any infringement of any patent, trademark, copyright, or other proprietary rights by the SAMPLE PROGRAMS or their use. Agilent Technologies does not warrant that the SAMPLE PROGRAMS are free from infringements of such rights of third parties. However, Agilent Technologies will not knowingly infringe or deliver software that infringes the patent, trademark, copyright, or other proprietary right of a third party.

Contents

1. User-defined Sequence Programming

Objective . . . 10
Flowchart of UDS programming . . . 11
Structure of UDS. . . . 12
Base Segment . . . 12
Segment . . . 12
Sequence . . . 12
Basic Example . . . 12
. . . 13
. . . 14
Creating UDS . . . 16
Step 1. Create a base segment . . . 17
Step 2. Parameter setting to operate a Base Segment . . . 20
Step 3. Creating a segment . . . 22
Step 4. Registering Base Segment into a Segment . . . 22
Step 5. Parameter setting for segment operation . . . 24
Step 6. Creating a Sequence . . . 26
Step 7. Adding a segment into a sequence . . . 26
Step 8. Setting the Sequence . . . 27
Executing UDS . . . 28
Sequence Execution . . . 28
Getting Measurement Results . . . 30
Types of measurement results . . . 30
Order of Getting Measurement Result . . . 32
Sectoring a Base Segment . . . 33
Parameters that require pre-setting . . . 35
Setting the measurement frequency of Narrow Band TAA measurement . . . 35
Setting the measurement frequency of Spectrum Measurement . . . 35
Setting the wavelength and channel bit rate of an Oscilloscope . . . 35
Limitations of UDS Programming . . . 37
Number of Base Segment in a Sequence . . . 37
Spectrum Measurement Base Segment . . . 37
Sample Program for User Defined Sequence . . . 38
Steps to follow in making UDS are as follows: . . . 39
Process for creating UDS: . . . 39
Problem Solving and Programming . . . 43
Case Study : TAA and PW . . . 44
Case Study : Stability . . . 50
Case Study : Write Current Sweep . . . 59
Case Study : Overwrite . . . 66
Case Study : Signal To Noise Ratio . . . 75
Case Study : Resolution . . . 82

2. User-defined Sequence Function Reference

Sequence Functions . . . 92
hpe5022_createSeq . . . 92
hpe5022_setupSeq . . . 94
hpe5022_freeSeq . . . 96
hpe5022_deleteSeq . . . 97

Contents

hpe5022_executeSeq . . . 98
hpe5022_addSeqSeg . . . 99
hpe5022_seqConfiguration . . . 101
hpe5022_seqConfiguration_Q . . . 107
hpe5022_seqSaGateConfig . . . 108
hpe5022_seqSaGateConfig_Q . . . 111
Segment Functions . . . 113
hpe5022_createSeqSeg . . . 113
hpe5022_deleteSeqSeg . . . 114
hpe5022_addSeqBaseSeg . . . 115
hpe5022_seqSegParameter . . . 117
hpe5022_seqSegParameterSweep . . . 120
hpe5022_seqSegParameterList . . . 123
hpe5022_seqSegParameterPoint_Q . . . 126
hpe5022_seqSegParameter_Q . . . 127
Base Segment Functions . . . 130
hpe5022_createSeqBaseSegMove . . . 130
hpe5022_createSeqBaseSegMoveWriteOffset . . . 132
hpe5022_createSeqBaseSegMoveReadOffset . . . 133
hpe5022_createSeqBaseSegErase . . . 134
hpe5022_createSeqBaseSegWrite . . . 136
hpe5022_createSeqBaseSegRead . . . 139
hpe5022_createSeqBaseSegWriteRead . . . 143
hpe5022_deleteSeqBaseSeg . . . 147
hpe5022_seqBaseSegParameter . . . 149
hpe5022_seqBaseSegParameterPoint_Q . . . 153
hpe5022_seqBaseSegParameter_Q . . . 155
hpe5022_seqBaseSegGateConfig . . . 158
hpe5022_seqBaseSegGateConfig_Q . . . 162
hpe5022_seqBaseSegWriteReadGateConfig . . . 164
hpe5022_seqBaseSegWriteReadGateConfig_Q . . . 167
Query Functions . . . 170
hpe5022_testParameterPoint_Q . . . 170
hpe5022_testParameter_Q . . . 172
hpe5022_resultPoint_Q . . . 175
hpe5022_result_Q . . . 177
hpe5022_wavePoint_Q . . . 179
hpe5022_wave_Q . . . 181
hpe5022_measStatus_Q . . . 183
BER Functions . . . 185
hpe5022_BER_seqSegRegister . . . 185
hpe5022_BER_seqSegRegisterSweep . . . 188
hpe5022_BER_seqSegRegisterList . . . 191
hpe5022_BER_seqSegRegisterPoint_Q . . . 194
hpe5022_BER_seqSegRegister_Q . . . 195
hpe5022_BER_seqBaseSegRegister . . . 197
hpe5022_BER_seqBaseSegRegisterPoint_Q . . . 200
hpe5022_BER_seqBaseSegRegister_Q . . . 201
hpe5022_BER_testRegisterPoint_Q . . . 203

hpe5022_BER_testRegister_Q 204
hpe5022_BER_errorPoint_Q 206
hpe5022_BER_error_Q. 208
hpe5022_BER_sectorErrorPoint_Q 210
hpe5022_BER_sectorError_Q 212
hpe5022_BER_sectorDataPoint_Q. 214
hpe5022_BER_sectorData_Q 216
hpe5022_BER_errorHistogram_Q 218

Contents

1 User-defined Sequence Programming

Objective

Measuring the TAA, Track Profile and other test parameters can be done under a single function as described in Chapter 2 (i.e., at-once measurement) of the programming manual. Under the User Defined Sequence (UDS), individual test sequence such as write offset, erase, write etc. has been provided as a call function to allow the user to create his program and thus making it possible to create different measurement sequence.

As an example, when measuring TAA for each rotation using the function hpe5022_measureTaa or hpe5022_setupTaa, measurement is executed in the following steps.

1. Write Offset - position's the head to write the data.
2. Erase
3. Write Data
4. Read Offset - position's the head to read the data.
5. Measure TAA (Track Average Amplitude).

Similarly, when measuring PW using hpe5022_measurePw and hpe5022_ setupPw functions, measure PW becomes the last step. If TAA and PW are to be measured under similar measurement conditions using hpe5022_measureTaa and hpe5022_measurePw functions, then steps 1 to 4 will make the measurement longer. Such a case can be eliminated if the measurement steps listed below can be executed, this will make our measurement shorter and faster.

1. Write Offset - position's the head to write the data
2. Erase
3. Write Data
4. Read Offset- position's the head to read the data.
5. Measure TAA (Track Average Amplitude).
6. Measure PW (Pulse Width).

The limited ability to create a measurement sequence as provided by the functions hpe5022_ measure xxx and hpe5022_setup xxx has been made possible through UDS function.

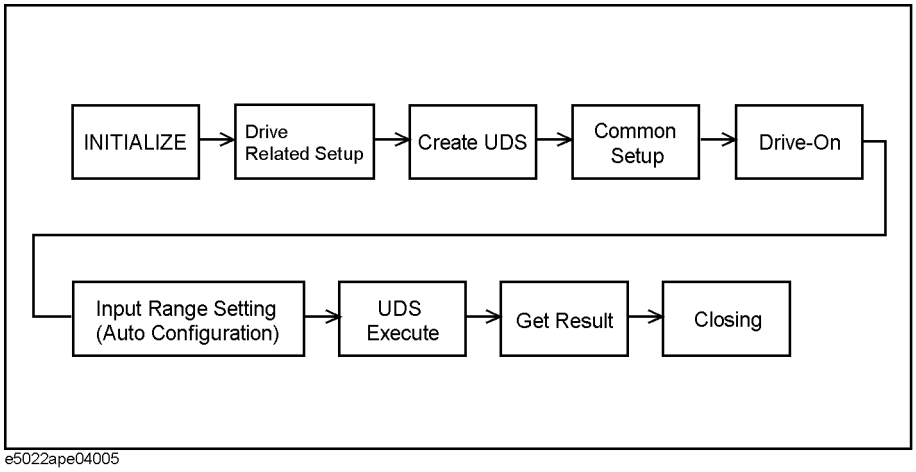
The initialization, drive related setup (spinstand), common parameter setup and close functions are used in a similar way when hpe5022_measure xxx and hpe5022_setupxxx functions are used.

NOTE The channel bit rate, erase current, write current and sense current parameters are integral parts of measurement and can be specified using the UDS function.

Flowchart of UDS programming

As mentioned in Chapter 2 of the E5022/E5023 programming manual, the UDS does not make up for the entire program. It simply allows the user to redefine the measurement sequence. The initialization, drive related setup, common parameter setup, drive on and auto configuration are executed under normal commands using the call function. Described in the figure below is a typical flowchart when UDS is used.

Figure 1-1 **Measurement Flowchart using UDS**



e5022ape04005

In this flowchart the Create UDS and Common Setup can also be set after the Drive-On operation. Also, the user can create multiple UDS, either in a single order or repeatable measurements

NOTE The BER measurement is not supported by UDS function.

Structure of UDS

The operation for each track of rotation are to be specified under UDS. These operations are registered to form a measurement sequence. The created measurement sequence is then executed for measurement.

UDS has three major elements, namely, the Base Segment, Segment and Sequence.

Base Segment

A base segment specifies the parameters needed to operate the head offset, erase, write and read (measurement) sequence. Parameters such as amount of head offset, erase current, write current and other parameters needed to execute the operation are also specified with this function.

Segment

A segment integrates the base segment. Base segments that are registered in this function are executed in accordance to the order of its registry.Thus, each base segment must be registered according to the order of measurement. Repeatable operation of base segments can also be done once they are registered in a segment by specifying the number of rotations. Also, the amount of movement such as head offset for Track Profile measurement can also be specified with this function.

Sequence

A sequence integrates the segment. Similar to a base segment, each segment must be registered into a sequence according to the order of measurement. After registering, the sequence must be setup and executed for measurement.

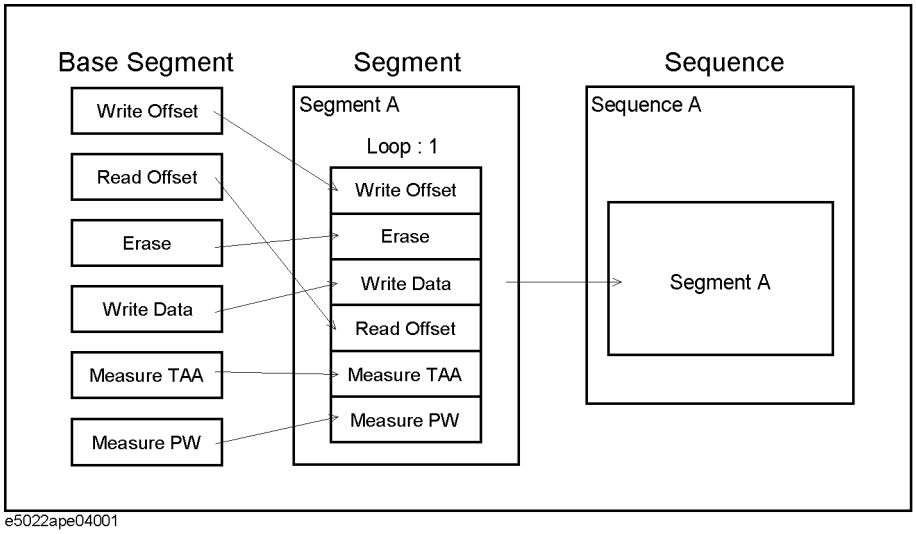
Basic Example

Listed below is a sample structure of the base segment, segment and sequence.

1. Write Offset - position’s the head to write the data
2. Erase
3. Write Data
4. Read Offset- position’s the head to read the data.
5. Measure the TAA (Track Average Amplitude).
6. Measure the PW (Pulse Width).

The first step is to create the needed base segments. The created base segments are then registered to form a segment. In this particular example only one segment is formed. The segment is then registered in a sequence. It is invalid to register a base segment directly into a sequence.

Figure 1-2 **Basic example**



Example of UDS with Repeated Segment

Example 1-1 **Example of UDS with Repeated Segment**

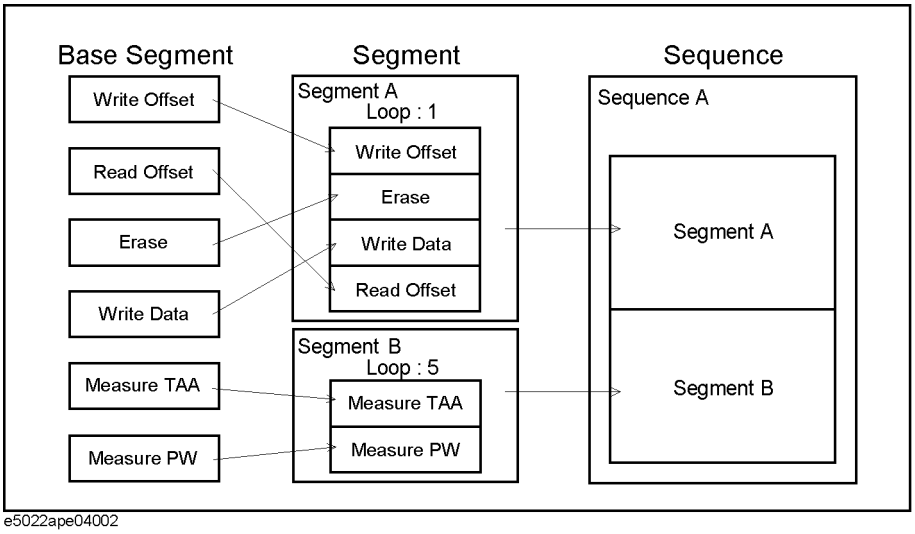
Repeated operation of UDS can be performed from the example given below.

1. Write Offset - position's the head to write the data
2. Erase
3. Write Data
4. Read Offset- position's the head to read the data.
5. Measure the TAA (Track Average Amplitude).
6. Measure the PW (Pulse Width).

The measurement of TAA and PW will be executed five times.

Similar to the construction of "Basic Example" on page 12, first create the base segment, then register it into a segment. But now two independent segments have to be created, since TAA and PW will be measured five times. We will divide the segment as segment 1 for non-measurement having one loop. And segment 2 as the measuring segment with 5 loops.

Figure 1-3 **Example of UDS with Repeated Segment**



Example of UDS with Sweep Operation

Example 1-2 **Example of UDS with sweep operation**

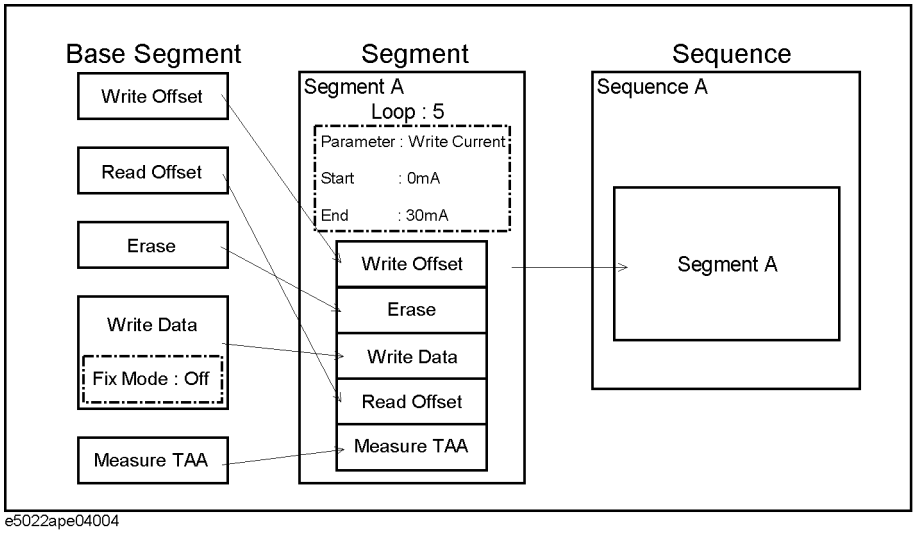
Sweep operation of UDS can also be performed in the given example below.

1. Write Offset - position's the head to write the data
2. Erase
3. Write Data
4. Read Offset- position's the head to read the data.
5. Measure the TAA (Track Average Amplitude).

The write current will be swept 7 times from 0mA to 30mA at an interval of 5mA.

The construction of base segment is somewhat similar to the previous example. However, in this case the parameters of write current base segment will have to be swept. To make this possible the parameter mode must be set to off (In actual setting it becomes VI_FALSE). Similarly, base segments are to be registered to form a segment. In this example only one segment exists, since the entire segment will be swept from a specified start and stop values.

Figure 1-4 **Example of UDS with sweep operation**



Remember that a segment can not be registered into another segment. If multiple measurements of TAA is required, then a separate segment will have to be created that carries the number of the desired measurement loops.

Creating UDS

This section explains the process on how to create the user define sequence. Create a sequence, execute the sequence and get the results are the first thing that we need to focus in order to create UDS.

The process for creating UDS are as follows:

Step 1. Create the base segment.

Step 2. Specify the parameters needed to move the base segment.

Step 3. Create the segment.

Step 4. Adding base segment to a segment.

Step 5. Specify the parameters needed to move the segment.

Step 6. Create the sequence.

Step 7. Adding segment to a sequence.

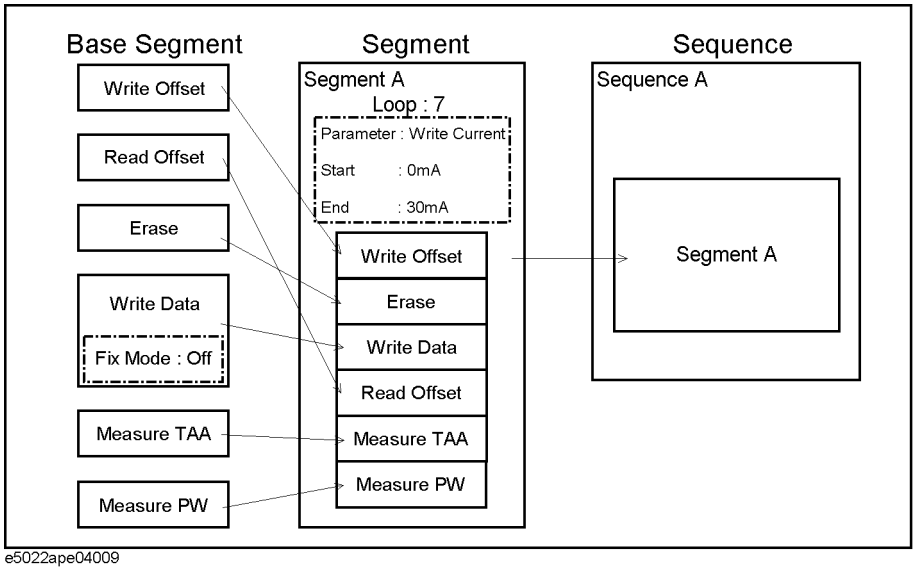
Step 8. Setting the sequence.

Listed below is an example on how a sequence operates.

1. Write Offset - move the head's position to write the data.
2. Erase
3. Write Data
4. Read Offset - move the head's position to read the data.
5. Measure TAA
6. Measure PW

(The write current will be swept 7 times from 0mA to 30mA at an interval of 5mA. The sequence structure is shown in Figure 1-5 on page 17.)

Figure 1-5 **Example of UDS**



Step 1. Create a base segment

The first step is to create a base segment for operation of the head offset, erase, write, read etc.

Types of Base Segment

There are various types of base segments, the following base segments will be defined for each operation.

Table 1-1 **Types of Base Segment**

Definition of Motion	Base Segment Function
Head Offset	hpe5022_createSeqBaseSegMove
Move the head for write offset	hpe5022_createSeqBaseSegMoveWriteOffset
Move the head for read offset	hpe5022_createSeqBaseSegMoveReadOffset
Erase	hpe5022_createSeqBaseSegErase
Write Data	hpe5022_createSeqBaseSegWrite
Read Data (Measurement)	hpe5022_createSeqBaseSegRead
Stability Measurement	hpe5022_createSeqBaseSegWriteRead

The head offset is used to move the head from the track center. Since you can not move the track, if you want to change tracks use the hpe5022_track function.

Except for Stability Measurement and Head Movement all other base segments will be used for this example.

Using the Base Segment Handler and base segment function

A handle number is used to distinguish each base segment in the system. The user must assign a name to each of the handler. From the given example below, the base segment handler has been assigned with the following names.

For each of these types the corresponding base segment function should be selected to execute operation. For example if the user wants to move the head to write offset, use the hpe5022_createSeqBaseSegMoveWriteOffset function.

Variable Name for Base Segment Handle (Name given by the user)	Base Segment Function
IdWriteOffset	hpe5022_createSeqBaseSegMoveWriteOffset
IdErase	hpe5022_createSeqBaseSegErase
IdWrite	hpe5022_createSeqBaseSegWrite
IdReadOffset	hpe5022_createSeqBaseSegMoveReadOffset
IdMeasTaa	hpe5022_createSeqBaseSegRead
IdMeasPw	hpe5022_createSeqBaseSegRead

Parameters for Base Segment Function

For a base segment function, the system ID of **Agilent E5022/E5023** will be assigned as the first parameter. While the last parameter of the base segment will be designated as the base segment handler.

Other parameters will vary according to its function. Listed below are the parameters for base segment.

Table 1-2 Parameters for Base Segment Function

Base Segment Function	Parameter	Designated Parameter
hpe5022_createSeqBaseSegMoveWriteOffset	Module Parameter	hpe5022_MODULE_PIEZO
hpe5022_createSeqBaseSegMoveReadOffset	Module Parameter	hpe5022_MODULE_PIEZO
hpe5022_createSeqBaseSegErase	Module Parameter	hpe5022_MODULE_DATA_GEN
	Sectoring	VI_FALSE

Table 1-2 Parameters for Base Segment Function

Base Segment Function	Parameter	Designated Parameter
hpe5022_createSeqBaseSegWrite	Data Pattern	hpe5022_PAT_DEFAULT
	Sectoring	VI_FALSE
	Module Parameter	hpe5022_MODULE_DATA_GEN
hpe5022_createSeqBaseSegRead	Data Pattern	hpe5022_PAT_DEFAULT
	Sectoring	VI_FALSE
	Module Parameter	hpe5022_MODULE_DATA_PARAM
	Measurement Parameter	hpe5022_MEAS_TAA and hpe5022_MEAS_PW

Module Parameter

Specifies the required module to execute the measurement.

Piezo device is used to move the head with respect to its base segment function. To move the head use the hpe5022_MODULE_PIEZO as the parameter. To operate the data generator for erase and write current base function use the hpe5022_MODULE_DATA_ GEN as the designated parameter. The parameter to be used for measurement depends on the type of module being used. Thus, the measurement parameter must be selected. From the given example the measurement for TAA and PW uses the parametric module for measurement, in this case select hpe5022_MODULE_ DATA_PARAM.

Sectoring

Specifies whether the track is to be divided into sectors or not.

Under normal operation the designated parameter is set to VI_FALSE. If sectoring is required, set it to VI_TRUE. The number of sector is specified by the function hpe5022_createSeq.

Data Pattern

Specifies the type of data pattern to be written on the media.

If the selected type is hpe5022_PAT_DEFAULT, then the data pattern specified by hpe5022_selectPattern function will be used. Other data patterns can also be selected from the hpe5022_selectPattern function.

Measurement Parameter

Specifies the measurement parameter to be used to read the data.

The parameter to be specified depends on what type of measuring instrument is to be used.

If the Parametric Module has been designated as the instrument as defined by 'hpe5022_MODULE_PARAM', then the functions hpe5022_MEAS_TAA, hpe5022_MEAS_ PW or hpe5022_MEAS_BASELINE should be selected.

Refer to the Reference Function to check what type of module is to be used for a particular measurement.

Example of Create Base Segment

Listed below are the actual base segments of the given example provided earlier.

Example 1-3 Create Base Segment

```
Call hpe5022_createSeqBaseSegMoveWriteOffset(id, hpe5022_MODULE_PIEZO, IdWriteOffset)

Call hpe5022_createSeqBaseSegMoveReadOffset(id, hpe5022_MODULE_PIEZO, IdReadOffset)

Call hpe5022_createSeqBaseSegErase(id, hpe5022_MODULE_DATA_GEN,VI_FALSE,IdErase)

Call hpe5022_createSeqBaseSegWrite(id, hpe5022_PAT_DEFAULT, VI_FALSE,
hpe5022_MODULE_DATA_GEN, IdWrite)

Call hpe5022_createSeqBaseSegRead(id, hpe5022_PAT_DEFAULT, VI_FALSE, hpe5022_MODULE_PARAM,
hpe5022_MEAS_TAA, IdMeasTaa)

Call hpe5022_createSeqBaseSegRead(id, hpe5022_PAT_DEFAULT, VI_FALSE, hpe5022_MODULE_PARAM,
hpe5022_MEAS_PW, IdMeasPw)
```

Step 2. Parameter setting to operate a Base Segment

This section explains the parameter setting necessary to operate the base segment. For example the operation to move the head, this will require us to specify the parameter of hpe5022_PARAMETER_HEAD_POS. This parameter is called the Designated Operating Parameter. To set this parameter, use the hpe5022_seqBaseSegParameter function and assign an ID for its base segment.

Parameter setting

Listed in the table below are valid parameters for each base segment.

Table 1-3 Parameter setting to operate Base Segment

Base Segment Function	Parameter Setting (Operating Parameter)	hpe5022_seqBaseSegParameter (Designated Operating Parameter)
hpe5022_createSeqBaseSegMove	Move Head	hpe5022_PARAMETER_HEAD_POS
hpe5022_createSeqBaseSegErase	Erase Current	hpe5022_PARAMETER_ERAS_CURR

Table 1-3 Parameter setting to operate Base Segment

Base Segment Function	Parameter Setting (Operating Parameter)	hpe5022_seqBaseSegParameter (Designated Operating Parameter)
hpe5022_createSeqBaseSegWrite	Write Current	hpe5022_PARAMETER_WRIT_CURR
	Channel Bit Rate	hpe5022_PARAMETER_CHAN_BIT_RATE
	Precompensation Delay 3 Types	hpe5022_PARAMETER_PREC_VAL1 hpe5022_PARAMETER_PREC_VAL2 hpe5022_PARAMETER_PREC_VAL3
hpe5022_createSeqBaseSegRead	Read Current	hpe5022_PARAMETER_SENS_CURR
	Channel Bit Rate	hpe5022_PARAMETER_CHAN_BIT_RATE
hpe5022_createSeqBaseSegWriteRead	Write Current	hpe5022_PARAMETER_WRIT_CURR
	Read Current	hpe5022_PARAMETER_SENS_CURR
	Channel Bit Rate	hpe5022_PARAMETER_CHAN_BIT_RATE
	Precompensation Delay 3 Types	hpe5022_PARAMETER_PREC_VAL1 hpe5022_PARAMETER_PREC_VAL2 hpe5022_PARAMETER_PREC_VAL3

Setting the parameters

The constant parameters defined in UDS sequence are always set by an exclusive function that sets each of these parameters. In this example the write offset, read offset, write current, channel bit rate and pre compensation delay are constant within the sequence. The functions hpe5022_writeOffset, hpe5022_readOffset, hpe5022_senseCurrent and other functions are used to set the parameters and are specified outside (Drive On, Set Up etc.) of UDS. The sequence of the UDS will execute the parameter values specified by these functions.

- Sweeping the parameter by segment.

  Use the hpe5022_seqBaseSegParameter function, when you sweep the parameters in a segment. Set the ‘fix mode’ to off (VI_FALSE) relative to its parameter. If ‘fix mode’ is not set to off (i.e. VI_TRUE), then its parameter value becomes constant, unaffected by the segment parameter value.

- Changing the parameter values within the UDS sequence.

  The specified write current value of hpe5022_writeCurrent function can be changed by specifying a new write current value in the hpe5022_seqBaseSegParameter function.

Example for setting the parameter of a base segment

Since the write current will be swept, the write current parameter mode must be set to off (VI_FALSE). And use ‘IdWrite’ as the designated base segment handler.

The parameter value will be set to 0. The sweep start and stop value will be used as

write current in the hpe5022_seqSegParameterSweep function.

Example 1-4 **Example for setting the parameters of a base segment**

```
Call hpe5022_seqBaseSegParameter(id, IdWrite, hpe5022_PARAMETER_WRITE_
CURR, 0, VI_FALSE)
```

Step 3. Creating a segment

The collection of base segment is called a segment.The number of segments to be created depends if a separate erase>write>read repetition and/or sweeping is required. There are no types of segment compared to base segment. A segment can not be integrated into another segment (i.e., nested segment is invalid).

Designating the Segment Handler

A handle number system must be designated to distinct one segment from another. This handle system can be any given variable name.

Parameters of segment function

For a segment function the system ID of **Agilent E5022/E5023** will be assigned as the first parameter. Set the parameters needed to execute sweeping for the registered base segments.The segment handler must be designated as the last parameter of the function.

Example of Create Segment

From the given example earlier, the write current will be swept 7 times at a step of 5mA from 0mA to 30mA. The designated segment handler will be IdSegment Example.

Example 1-5 **Create segment**

```
Call hpe5022_createSeqSeg(id, 7, IdSegmentExample)
```

Step 4. Registering Base Segment into a Segment

Register the base segment into the created segment. The registered base segment will be executed according to the order by which it is registered.

Designating the Data ID

In order to get the results of measurement, an ID number must be designated for each registered base segment. This will enable us to identify the source of the measurement results, (i.e, from what segment is a base segment being executed).

However, only the hpe5022_createSeqBaseSegRead and hpe5022_createSeqBase SegReadWrite base segment function will require a data ID, others can be considered as dummy since it has no purpose of use.

In this particular example the measured data will be returned by the base segment handler of ‘IdMeasTaa’ and ‘IdMeasPw’. And the assigned IdDataTaa and

IdDataPw will pick up the returned data.

Parameters for adding a base segment

In order to register a base segment with its parameter functions. Assign the **Agilent E5022/E5023** of the system as the first parameter ID. Then set the segment handler to be registered corresponding to its parameter. The next step is to set the base segment handler to be registered relative to its parameter. The last parameter will be a base segment ID, to be used in order to pick up the data.

Except for the ID of hpe5022_createSeqBaseSegRead and hpe5022_createSeq BaseSegReadWrite base segment function. All other base segment IDs will be considered as dummy since it will not be used, even though each ID returns a corresponding value.

Closing the registry

Closing the segment registry can be done after adding the base segments using the 'hpe5022_SEQ_END' function. You can not register a segment into a sequence without closing the registry. Furthermore, once the segment registry had been closed you can no longer add any base segment to it.

Example of adding Base Segment to a Segment

As shown in the example below, the IdMeasTaa and IdMeasPw have been set to get the data relative to 'hpe5022_createSeqBasesegRead' function.

Example 1-6 Example of adding Base Segment to a Segment

```
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdWriteOffset, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdErase, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdWrite, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdReadOffset, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdMeasTaa, IdDataTaa)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, IdMeasPw, IdDataPw)
Call hpe5022_addSeqBaseSeg(id, IdSegmentExample, hpe5022_SEQ_END, Dummy)
```

Step 5. Parameter setting for segment operation

By setting the fix mode parameter of the base segment “hpe5022_seqBaseSegParameter” to VI_FALSE. The user will be able to set the operating parameter of the segment which integrates the base segment.

Types of Parameter Settings

There are 3 types of setting the operating parameter of a segment.

- Sweep

  Repeatable operation of the base segment can be made possible by setting the base segment’s operating parameters as start value and stop value. And set the number of increments within this range.

  For example, given that the segment loop is 6 with a start value of 0 and a stop value of 10. The value will be incremented by 2 from 0, 2, 4, 6, 8, and 10 for each rotation.

  If the number of loops is set to 3 then its value will change from 0, 5, and 10 for each rotation. With the number of increment set to 5 (i.e, step).

  This type of setting can be executed by the **hpe5022_seqSegParameterSweep** function.

- List Sweep

  List sweep is a type of sweep parameter setting. The only difference is that values are listed in array form. The intervals between each set value may not be equal thus giving the user the flexibility in setting the parameter values. For example, with six sweep counts, the array elements can be 0, 2, 5, 7, 8.5 and 10.

  If the number of sweep count is greater than the number of array elements then the last element will be repeated for continuity.

  If the number of sweep count is less than the number of array elements then the number of corresponding sweep counts will be used for each element.

  By using the **hpe5022_seqSegParameterList** function you will able to operate the list sweep.

- Constant Parameter

  By using the **hpe5022_seqSegParameter** function the operating parameter of the segment can be set to a constant value.

Parameter Setting

Shown in Table 1-4 on page 25 is the parameter setting of the base segment whose operation can be changed by the parameters described inTable 1-3 on page 20.

.However the parameters listed below can only be executed if the base segment’s fix mode is set to ‘VI_FALSE’.

Table 1-4 Parameter setting to operate Base Segment

Operating Parameter	Base segment functions that are affected by the Fix Mode, when it is set to off (VI_FALSE)
Head Position hpe5022_PARAMETER_HEAD_POS	hpe5022_createSeqBaseSegMove hpe5022_createSeqBaseSegMoveWriteOffset hpe5022_createSeqBaseSegMoveReadOffset
Erase Current hpe5022_PARAMETER_ERAS_CURR	hpe5022_createSeqBaseSegErase
Write Current hpe5022_PARAMETER_WRIT_CURR	hpe5022_createSeqBaseSegWrite hpe5022_createSeqBaseSegReadWrite
Read Current hpe5022_PARAMETER_SENS_CURR	hpe5022_createSeqBaseSegRead hpe5022_createSeqBaseSegReadWrite
Channel Bit Rate hpe5022_PARAMETER_CHAN_BIT_RATE	hpe5022_createSeqBaseSegWrite hpe5022_createSeqBaseSegRead hpe5022_createSeqBaseSegReadWrite
Precompensation Delay (3 Types) hpe5022_PARAMETER_PREC_VAL1 hpe5022_PARAMETER_PREC_VAL2 hpe5022_PARAMETER_PREC_VAL3	hpe5022_createSeqBaseSegWrite hpe5022_createSeqBaseSegReadWrite

As shown in the example below, the IdMove of the base segment is not set to VI_FALSE. Thus even if the head position is swept as defined by hpe5022_seqSegParameterSweep it will not affect the head offset value of the hpe5022seqBaseSegparameter function.

```
Call hpe5022_createSeqBaseSegMove(id, hpe5022_MODULE_PIEZO, IdMove)
Call hpe5022_createSeqBaseSegMoveWriteOffset(id, hpe5022_MODULE_PIEZO, IdWriteOffset)
Call hpe5022_seqBaseSegParameter(id, IdMove, hpe5022_PARAMETER_HEAD_POS, 0, VI_TRUE)
Call hpe5022_createSeqSeg(id, 7, IdExample)
Call hpe5022_addSeqBaseSeg(id, IdExample, IdMove, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdExample, IdWriteOffset, Dummy)
Call hpe5022_addSeqBaseSeg(id, IdExample, hpe5022_SEQ_END, Dummy)
Call hpe5022_seqSegParameterSweep(id, IdExample, hpe5022_PARAMETER_HEAD_POS, -0.001,
+0.001)
```

Getting the specified value

The user can get the results when hpe5022_testParameter_Q is used. With this function the user can actually query the specified value.

As an example, having a sweep count of 6 from start = 0 to stop=10. The returned value will change as it loops around from 0,2,4,6,8 and 10. The data will be returned in array form. The “hpe5022_testParameter_Q” function can be executed after setting up the the measurement sequence.

Example of parameter setting for base segment operation in a segment

As an example, let the write current be swept from 0mA to 30mA. having equal intervals. With respect to the segment the write current, start (0mA) and stop (30mA) value are to be specified.

Example 1-7 **Example of parameter setting for base segment operation**

```
Call hpe5022_seqSegSweep(id, IdSegmentExample,
hpe5022_PARAMETER_WRIT_CURR, 0, 0.03)
```

Step 6. Creating a Sequence

The sequence integrates the segments needed to execute measurement. The sequence will execute each segment in accordance to the order by which it is registered. As mentioned in the previous section, base segment is registered to form a segment and segment is registered to form a sequence. A base segment can not be registered directly into a sequence.

Designating a Sequence Handler

A handle number system will be used to distinct the created sequence. The sequence handler to be designated can be any variable name.

Parameters of Sequence Function

The system ID of **Agilent E5022 A** will be assigned as the first parameter of the sequence function. The second parameter will be the number of sectors, used when base segment sectoring is required. The third parameter will be the sequence handler itself.

Sequence example

The IdSequence Example will be used as the sequence handler.

Example 1-8 **Create a Sequence**

```
Call hpe5022_createSeq(id,numOfSec,IdSequenceExample)
```

Step 7. Adding a segment into a sequence

The created segment must be registered into a sequence. The sequence will execute each segment in accordance to the order by which it is registered.

Parameters for adding Segment

In order to register a segment to a sequence relative to its parameter functions. Assign the **Agilent E5022/E5023** of the system as the first parameter ID. Then set the sequence handler to be registered corresponding to its parameter. Then set the segment handler as its third parameter. The last parameter will be used for segment ID.

Closing the Registry

Closing the sequence registry can be done after adding the segments using the “hpe5022_SEQ_END” function. Registering a sequence into another sequence is not allowed. Also, once the sequence registry is closed you can not add any segments to it.

Example of adding segment into a sequence

From the given example below only one segment is registered in the sequence.

Example 1-9 **Example of adding segment to a sequence**

```
Call hpe5022_addSeqSeg(id, IdSequenceExample, IdSegmentExample,
IdDataSegment)

Call hpe5022_addSeqSeg(id, IdSequenceExample, hpe5022_SEQ_END,
Dummy)
```

Step 8. Setting the Sequence

When measurement requires the use of spectrum analyzer or digital oscilloscope The user must set the parameters to execute sequence measurement.

For example when spectrum analyzer is used for measurement, its center frequency must be specified. To set the parameter forr center frequency use the “hpe5022_seqConfiguration” function and set the sequence handle.

Executing UDS

This section explains the execution of the sequence and how to get the measurement results.

Sequence Execution

This covers the necessary preparations after creating the sequence.

Before executing the sequence

Before executing the sequence make sure that the drive is on, the spindle is running and the head is loaded on the media. Also, the common setup parameter and auto configuration (input range setting) must be performed.

Setting the sequence

Set the “hpe5022_setupSeq” function before you execute the sequence. Without this function the sequence can not be executed.

Once the sequence had been set, the “hpe5022_setupSeq” function will set the test sequence in each module and measuring instrument. If there are multiple sequences to be executed, executing them at the same time may result to memory error. To avoid this, set and execute the sequence one at a time and delete the previously set up sequence. When you delete, use the “hpe5022_freeSeq” and “hpe5022_deleteSeq” functions. However, if there are only a few test sequences in UDS then the user may set up and execute the sequence at the same time.

Example 1-10 **Setting the sequence**

Set the sequence created by “Sequence example” on page 26

```
Call hpe5022_setupSeq(id, IdSequenceExample)
```

Executing the sequence

To execute the sequence use the function hpe5022_executeSeq and set the sequence handle.

Example 1-11 **Executing sequence**

```
Call hpe5022_executeSeq(id, IdSequenceExample)
```

Removing sequence

As mentioned above if there are too many sequences it could result to memory error. To avoid this the user must remove some of the sequences. Using the “hpe5022_freeSeq” function, and specify the handle of the sequence you want to remove.

This function (hpe5022_freeSeq) removes the set up test sequence from the module, but still remains in the PC memory. The sequence can still be executed

when you reset the sequence. When “hpe5022_deleteSeq” is used, this function will remove the sequence completely from all modules and you will not be able to recover it again. When you use “hpe5022_deleteSeq”, you should also use “hpe5022_deleteSeqSeg” and “hpe5022_deleteSeqBaseSeg” function to avoid memory errors.

Example 1-12 Example execute sequence

```
Call hpe5022_executeSeq(id, IdSequenceExample)
```

Getting Measurement Results

This section explains how to get the data after sequence execution.

Types of measurement results

The measured results from “hpe5022_createSeqBaseSegRead” and “hpe5022_create SeqBaseSegWriteRead” can be acquired by the hpe5022_result_Q and hpe5022_ wave_Q functions. The measurement results carried by hpe5022_result_Q and hpe5022_wave_Q functions are dependent on the type of measurement specified by hpe5022_createSeqBaseSegRead and hpe5022_createSeqBaseSegWriteRead functions.

Table 1-5 Measurement results from hpe5022_result_Q and hpe5022_wave_Q function

Measurement	Results from hpe5022_result_Q function	Results from hpe5022_wave_Q function
hpe5022_MEAS_TAA	TAA Positive (TAA+) TAA Negative (TAA-)	TAA (raw data) Positive value Negative value
hpe5022_MEAS_PW	PW Positive (PW+) PW Negative (PW-)	PW (raw data) Positive value Negative value
hpe5022_MEAS_BASELINE	Baseline Positive Baseline Negative	Baseline Histogram Data
hpe5022_MEAS_LEVEL	Signal Level	None
hpe5022_MEAS_WAVE	None	Waveform Data
hpe5022_MEAS_SPEC	None	Spectrum Data

The hpe5022_result_Q is generally used for hpe5022_MEAS_TAA, hpe5022_MEAS_PW and hpe5022_MEAS_BASELINE measurements.

- hpe5022_result_Q function

hpe5022_MEAS_TAA, hpe5022_MEAS_PW and hpe5022_MEAS_ BASELINE measurement.

Each of these functions will return a positive and negative value. The user will have to compute other measurements based on these results. For example if the user wants to know other statistics such as standard deviation and average. the user will must to other programming techniques for computation.

hpe5022_MEAS_LEVEL measurement

Returns the Narrow Band TAA value only.

- hpe5022_wave_Q function

hpe5022_MEAS_TAA, hpe5022_MEAS_PW and hpe5022_MEAS_ BASELINE measurement.

The results returned by “hpe5022_result_Q” function gives the average results for one revolution of TAA(+,-), PW(+,-), and Baseline(+,-) measurement. The TAA and PW raw data serves as element to get the average results. Hence, the average results of the elements from hpe5022_wave_Q function will be returned by the hpe5022_result_Q function.The parametric module monitors the output voltage of the PW detector and peak level detector of the internal module in a fix time interval. And transforms the result to TAA(+,-), PW(+,-) and Baseline(+,-).

If loop and sweep measurements are to be executed in a segment, only the results of the last track rotation will be returned.

hpe5022_MEAS_WAVE measurement

The hpe5022_wave_Q returns the data of the waveform. However, it only returns one cycle of the waveform data. The user must set the sampling frequency when this function is used, sampling frequency is the product of channel bit rate and over sample rate as defined by hpe5022_waveOverSampleRate function.

hpe5022_MEAS_SPEC measurement

The hpe5022_wave_Q returns the data of the spectrum measurement.

Elements of returned data

By using the hpe5022_resultPoint_Q and hpe5022_wavePoint_Q function, this will allow the user to evaluate the returned data from the elements. It also returns the number of sectors and number of sweep specified by the base segment. Refer to “Sectoring a Base Segment” on page 33.

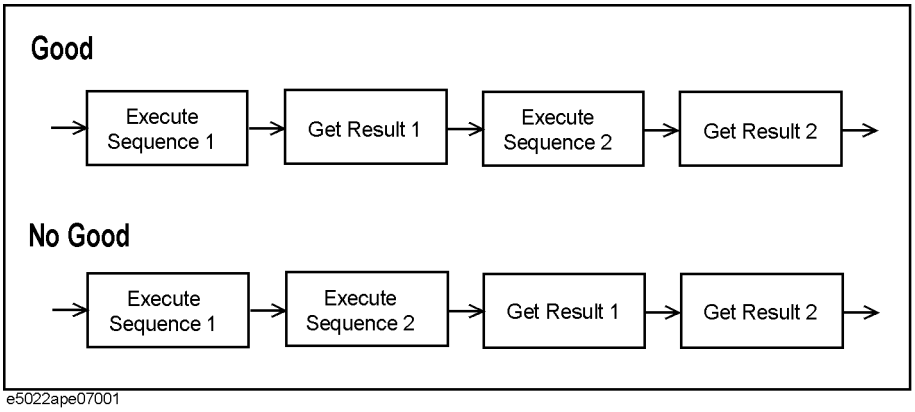
Example 1-13 Example of measurement results (visual basic, option base 0)

```
Call hpe5022_resultPoint_Q(id, IdSequenceExample, IdDataSegment, IdDataTaa, NoOfLoopTaa, NoOfSectorTaa, NoOfElementTaa)

Call hpe5022_resultPoint_Q(id, IdSequenceExample, IdDataSegment, IdDataPw, NoOfLoopPw, NoOfSectorPw, NoOfElementPw)

ReDim ResultTaa(NoOfLoopTaa-1, NoOfElementTaa-1)

ReDim ResultPw(NoOfLoopPw-1, NoOfElementPw-1)

Call hpe5022_result_Q(id, IdSequenceExample, IdDataSegment, IdDataTaa, ResultTaa(0,0))

Call hpe5022_result_Q(id, IdSequenceExample, IdDataSegment, IdDataPw, ResultPw(0,0))
```

Order of Getting Measurement Result

After the executing the sequence, the user must query the results of each corresponding measurement right after executing each sequence as shown in Figure 1-6.

Figure 1-6 Order of Getting Measurement Result



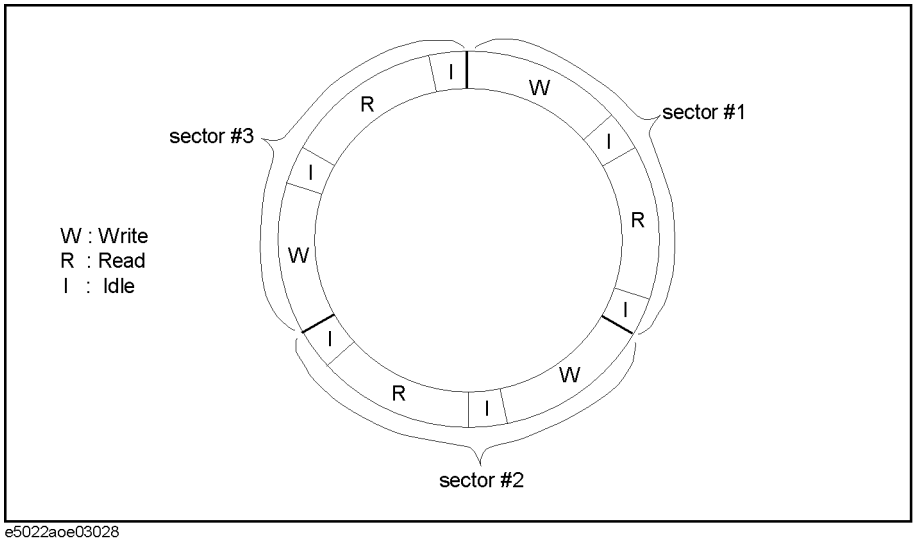
e5022ape07001

Sectoring a Base Segment

As mentioned in the previous example a base segment is usually operated in an unsectored track. However, it can also be operated for sectored measurement, commonly used in Stability measurement.

Sectored measurement can be made possible only for “hpe5022_measureStability” and “hpe5022_setupStability” function prior to UDS.

Figure 1-7 **Sectored Track**



e5022aoe03028

Set the parameter of the “hpe5022_createSeqBaseSegReadWrite” function to VI_TRUE, to be able to divide the track into sectors. The number of sector is specified by hpe5022_createSeq(xxx, **numOfSec**, xxx) function. Sectoring is used to speed up measurement and to evaluate TAA stability at various sectors of the track. For example if the user wants to read the data for 10 loops and get the average, you can reduce it to five and still get the same number of data provided that there are two sectors, thereby shortening the time to measure.

Track sectoring can only be executed by the following functions:

- hpe5022_createSeqBaseSegErase
- hpe5022_createSeqBaseSegWrite
- hpe5022_createSeqBaseSegRead
- hpe5022_createSeqBaseSegReadWrite

Sectoring is not allowed for ‘Sweep Measurement’ since the specified parameter value of the track is fixed.

The “hpe5022_createSeqBaseSegRead” and “hpe5022_createSeqBaseSegWrite” executes sectoring for one revolution. As a result, the measured results will increase. The element results are then returned by hpe5022_result Point_Q and

hpe5022_wavePoint_Q. The signal between sector intervals will be disrupted by the hpe5022_createSeq BaseSegErase and hpe5022_createSeqBaseSegWrite function during execution.

Parameters that require pre-setting

As mentioned previously, under normal operation the functions (hpe5022_measurexxxx) and (hpe5022_setupxxxx) automatically sets the parameter values for measurement. However, under UDS there are parameters that the user must set up in order to execute measurement.

Setting the measurement frequency of Narrow Band TAA measurement

When measuring with a spectrum analyzer at a specific level of frequency and resolution band width, use the function below.

- hpe5022_seqConfiguration

When measurement is performed with a spectrum analyzer, The RBW (resolution band width) must be set relative to the sequence handle. However, only one value can be set for each sequence. See “hpe5022_seqConfiguration”.

The values set by functions ‘hpe5022_narrowBandTaaFrequencyMode’, ‘hpe5022_narrowBandTaaFrequency’ and ‘hpe5022_narrowBandTaaBandWidth’ will not be reflected using a spectrum analyzer under UDS.

Setting the measurement frequency of Spectrum Measurement

When measuring with a spectrum analyzer at some frequency level and resolution band width, use the function below .

- hpe5022_seqConfiguration

When spectrum measurement is performed by a spectrum analyzer, The start and stop frequencies and RBW (resolution band width) must be set relative to the sequence. However, only one value can be set for each sequence.

The values set by functions ‘hpe5022_spectrumFrequency’ and ‘hpe5022_spectrumBandWidth’ under standard measurement will not work under UDS.

Setting the wavelength and channel bit rate of an Oscilloscope

When measurement is performed by an oscilloscope, the channel bit rate and waveform period must be specified.

- hpe5022_seqConfiguration

The waveform period must be set relative to its sequence. However, only one value can be set for each sequence. See “hpe5022_seqConfiguration” .

Values specified by functions ‘hpe5022_waveOverSampleRate’, ‘hpe5022_waveAverage’ and ‘hpe5022_waveDelayTime’ will be reflected in measurement of the oscilloscope under a UDS. By setting the parameters channel

bit rate and the wavelength for one cycle (period), the user will be able to determine the sampling frequency and length of data.

Limitations of UDS Programming

User Defined Sequence is a primitive programming language related with the system hardware such that there are some limitations to programming.

Number of Base Segment in a Sequence

The total number of base segments in a sequence that you can register is limited to 1024. When segment is repeated, the total number of base segments are also counted according to the number of repeated segments.

Spectrum Measurement Base Segment

A base segment that uses spectrum analyzer must be included in the sequence. this base segment must be located at the last line of the sequence. In other words, When you register the base segment into a segment, the base segment must be at the last line. Also, when you register the segment which contains the base segment, it must be located at the last line of the sequence.

Sample Program for User Defined Sequence

As previously discussed in Chapter 2 of Agilent E5022/E5023 programming manual, measurements such as TAA and Track Profile can be done under a single function. However, in User Define Sequence a step by step process is required in order to execute measurement, such as functions for head offset, erase, write and measurement function. These individual functions are integrated to form a sequence. As a user you have the flexibility to create your own sequence for any measurements.

In User Defined Sequence, each step must be defined for every movement of the head.These steps are then registered to form a sequence. The created sequence is then executed for measurement. For every rotation, steps such as move head, erase, write and read are called base segments.

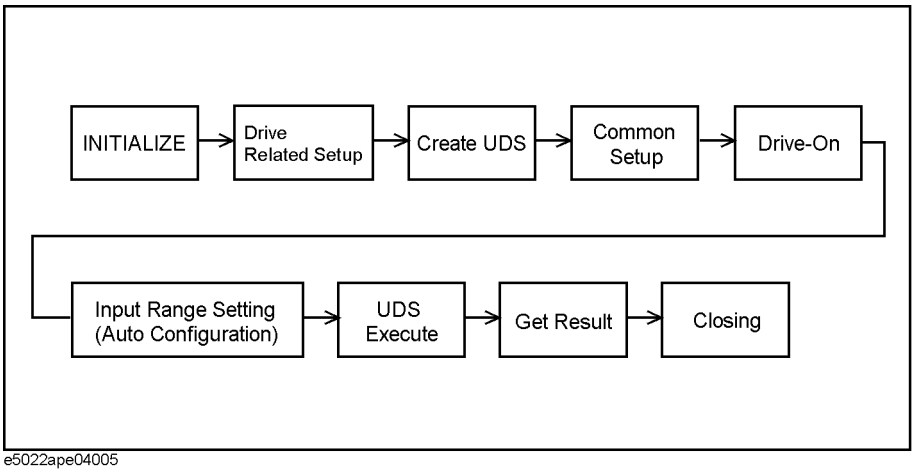
These base segments are then integrated to form a segment. Every base segment that is registered into a segment must be written in proper order to execute measurement. In addition, repeatable operation of segments can be done by sweeping a specified segment parameter value.

Segments are then integrated and registered to form a sequence. Segments that need to be registered to a sequence must also be in proper order. After registering, measurement can be performed using the execute function.

NOTE

Sample program for UDS is installed together with the system software “Agilent Technologies E5022/E5023 Hard Disk Read/Write Test System Operation/ Programming Manual”. This sample program is developed in Visual Basic, the file directory is “c:\Program Files\Agilent\E5022\doc\sample3”.

Figure 1-8 Measurement Flowchart using User Defined Sequence



e5022ape04005

Before performing any measurement with UDS there are various parameter settings and procedures that must be given due attention. The illustration above shows how the system works using UDS. There are nine steps in making the UDS

these are: initialize, drive related setup, create the UDS program, Common parameter setup, the Drive On, Auto Configuration, execute UDS, get results and closing. After program execution, get the results and close the system. The Common parameter setup and Drive On can be set right after Create UDS but not after the UDS execute. It is also possible to create multiple and repeatable UDS for measurement.

Steps to follow in making UDS are as follows:

Step 1. Initialize: This step sets the address for each module, initializes all modules and get the handle number. The handle number is a number given to a system to control it.

Step 2. Drive Related Set Up: This step specifies the spinstand configuration such as spinstand rpm, data area, and the head position.

Step 3. Create UDS : This step is where the User Defined Sequence is created, where the base segment, segment, and sequence are defined for a specific measurement.

Step 4. Common Setup: This setup configures the data pattern and bit error rate.

Step 5. Drive On: This step specifies the spinstand configuration such as spinstand rpm, data area, and the head position. After configuring the spinstand , turn on the drive to rotate the spindle and to load the head.

Step 6. Input Range Setting (Auto Configuration): This step optimizes the input level control.

Step 7. Execute UDS: This step processes the parameters for measurement and executes the UDS.

Step 8. Get the results: This step gets the measurement results.

Step 9. Closing: This step closes the system and turns off the drive.

Process for creating UDS:

Step 1. Create the base segment.

Step 2. Specify the parameters needed to operate the base segment.

Step 3. Create the segment.

Step 4. Adding base segment to a segment.

Step 5. Specify the parameters needed to operate the segment.

Step 6. Create the Sequence.

Step 7. Adding segment to a Sequence.

Step 8. Setting the Sequence.

Refer to Chapter 4 of the programming manual for more detailed description on how to create UDS.

The table below shows the valid parameters to be used in order to create a base segment function. This table serves as a quick reference.

Table 1-6 Valid parameters to create a base segment function

Base Segment Function		
Move	Module	hpe5022_MODULE_PIEZO
Read Offset	Module	hpe5022_MODULE_PIEZO
Write Offset	Module	hpe5022_MODULE_PIEZO
Erase	Module	hpe5022_MODULE_DATA_GEN
Write	Data Pattern	hpe5022_PAT_DEFAULT
		hpe5022_PAT_LF
		hpe5022_PAT_HF
		hpe5022_PAT_ISO
		hpe5022_PAT_PRBS
		hpe5022_PAT_REP
		hpe5022_PAT_NLTS_5TH
		hpe5022_PAT_OWHF
		hpe5022_PAT_OWLF
		hpe5022_PAT_USER
	Sectoring	VI_TRUE
		VI_FALSE
	Module	hpe5022_MODULE_DATA_GEN
Read	Data Pattern	hpe5022_PAT_DEFAULT
		hpe5022_PAT_LF
		hpe5022_PAT_HF
		hpe5022_PAT_ISO
		hpe5022_PAT_PRBS
		hpe5022_PAT_REP
		hpe5022_PAT_NLTS_5TH
		hpe5022_PAT_OWLF

Table 1-6 **Valid parameters to create a base segment function**

Base Segment Function		
		hpe5022_PAT_OWHF
		hpe5022_PAT_USER
	Sectoring	VI_TRUE
		VI_FALSE
	Module	hpe5022_MODULE_PARAM
		hpe5022_MODULE_DSO
		hpe5022_MODULE_SA
	Read Option	hpe5022_MEAS_TAA (for MODULE_PARAM)
		hpe5022_MEAS_PW (for MODULE_PARAM)
		hpe5022_MEAS_BASE (for MODULE_PARAM)
		hpe5022_MEAS_LEVEL (for MODULE_SA)
		hpe5022_MEAS_WAVE (for MODULE_DSO)
Read&Write	Data Pattern	hpe5022_PAT_DEFAULT
		hpe5022_PAT_LF
		hpe5022_PAT_HF
		hpe_5022_PAT_ISO
		hpe_5022_PAT_PRBS
		hpe_5022_PAT_REP
		hpe_5022_PAT_NLTS_5TH
		hpe_5022_PAT_OWLF
		hpe_5022_PAT_OWHF
		hpe_5022_PAT_USER
	Sectoring	VI_TRUE
		VI_FALSE
	Write Module	hpe5022_MODULE_DATA_GEN
	Write Ratio	0.2 ~ 0.8 (range)
	Read Module	hpe5022_MODULE_PARAM
		hpe5022_MODULE_DSO
		hpe5022_MODULE_SA

Table 1-6 Valid parameters to create a base segment function

Base Segment Function		
	Read Ratio	0.2 ~ 0.8 (range)
	Read Option	hpe5022_MEAS_TAA (for MODULE_PARAM)
		hpe5022_MEAS_PW (for MODULE_PARAM)
		hpe5022_MEAS_BASE (for MODULE_PARAM)
		hpe5022_MEAS_LEVEL (for MODULE_SA)
		hpe5022_MEAS_WAVE (for MODULE_DSO)

Problem Solving and Programming

Programmers use a widely practiced technique for solving programming problems called the software engineering method.

The software engineering method of problem solving uses the following five steps:

1. Specify the problem requirements.
2. Analyze the problem.
3. Design the algorithm to solve the problem.
4. Implement the algorithm.
5. Test and verify the program.

Specify the problem requirements the user must state the problem completely and unambiguously in order to gain a clear understanding of what solution is required. You must be able to recognize and define the problem precisely, to eliminate unimportant aspects and to identify the root problem.

Analyze the problem identify the problem (a) inputs, i.e. the data you have to work with; (b) outputs, i.e., the desired results and (c) any additional requirements for or constraints on the solution. At this stage, you need to determine the required format in which the results should be displayed.

These first two steps are the most critical; if they are not done properly, you will be trying to solve the wrong problem. Read the problem statement carefully, first to obtain a clear idea of the problem and second, to determine the inputs and outputs.

Design the algorithm to solve the problem this requires you to write step-by-step procedures - the algorithm - solve the problem as intended. Writing the algorithm is often the most difficult part of the problem-solving process. Don't attempt to solve every last detail of the problem at the beginning. You first list the major steps (sub- problems) that need to be solved. Most computer algorithm consists of at least the following sub-problems.

1. Read the data.
2. Perform the computation/calculation.
3. Display the results.

Implement the algorithm this involves writing the program. This requires you to know a particular programming language. You must convert each algorithm step into a statement in that language.

Test and verify the program this calls for the user to test the completed program to verify if it works as desired. Don't rely on just one test case, run the program several times using different sets of data.

Case Study : TAA and PW

Determine the Track Average Amplitude and Pulse Width of the of the read back signal.

Analysis

In order to ensure accurate detection of the read back pulses its read channel amplitude characteristics are to be identified. One characteristic, is to measure the average amplitude of evenly spaced pulses read from the disk, this average amplitude is called track average amplitude. PW is a parameter which indicates the upper limitation of flux change per inch from the head/disk combination under test. An increased density of FCI in the media will depend on the ability of the head to write and read narrow pulses having minimum interaction with each other.

Data Requirements

Problem Input : Number of revolutions to be averaged.

Write and read data patterns

Problem Output : Measure TAA+ and TAA-.

Measure PW+ and PW- .

Algorithm

1. Set up the parameters for measurement.
2. Compute and display the measured TAA and PW.

Implementation

To create the UDS follow the steps and procedures listed on “Process for creating UDS:” on page 39. First, create the write offset, erase, write, read offset and read base segments. Set the necessary parameters to operate each base segment function. Then register them to form a segment, and specify the number of loops or as to how many times you want to execute the base segments registered in it. Then we register the segment to a sequence for program execution.

Description

The TAA and PW are both parametric tests. TAA provides an average peak to peak amplitude measurement of equally spaced transitions of the read channel. It measures the efficiency of the read channel, determines the proper signal required for drive design and provides reference for other test methods. The PW on the other hand provides information about the quality of the head and media under test. It also provides information about the shape of individual pulses.

Table 1-7 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Write	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 4	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 5	Read (Measure TAA)	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_PARAM
		Read Option	hpe5022_MEAS_TAA
Step 6	Read (Measure PW)	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_PARAM
		Read Option	hpe5022_MEAS_PW

Step 1. Move the head to the write track offset.

Step 2. Erase the track without sectoring.

Step 3. Write the LF data pattern without sectoring. The data signal is supplied by the hpe5022_MODULE_DATA_GEN.

Step 4. Move the head to the read track offset.

Step 5. Read the TAA of the LF data without sectoring. The read data is measured by the parametric module.

Step 6. Read the PW of the LF data without sectoring. The read data is measured by the parametric module.

Described in the table above are six steps to create the parameters of each base segment function. The sectored flag operation of the Write base segment (Step 3) is defined as VI_FALSE because the sequence will not be sectored. Also, the sectoring of read base segments `hpe5022_createSeqBaseSegRead (HpE5022, hpe5022_PAT_ISO, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, SeqBaseSegMeasTaa)` (Step 5) and `hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_ISO, VI_FALSE,hpe5022_MODULE_PARAM, hpe5022_MEAS_PW, SeqBaseSegMeasPw)` (Step6) are defined as VI_FALSE. Since the read option is defined as hpe5022_MEAS_TAA and hpe5022_MEAS_PW, the module must be hpe5022_MODULE_PARAM.

Refer to Chapter 3 of the programming manual for more detailed descriptions of the function of each base segment and its parameters.

Example 1-14 Sample Program (TAA and PW Measurement)

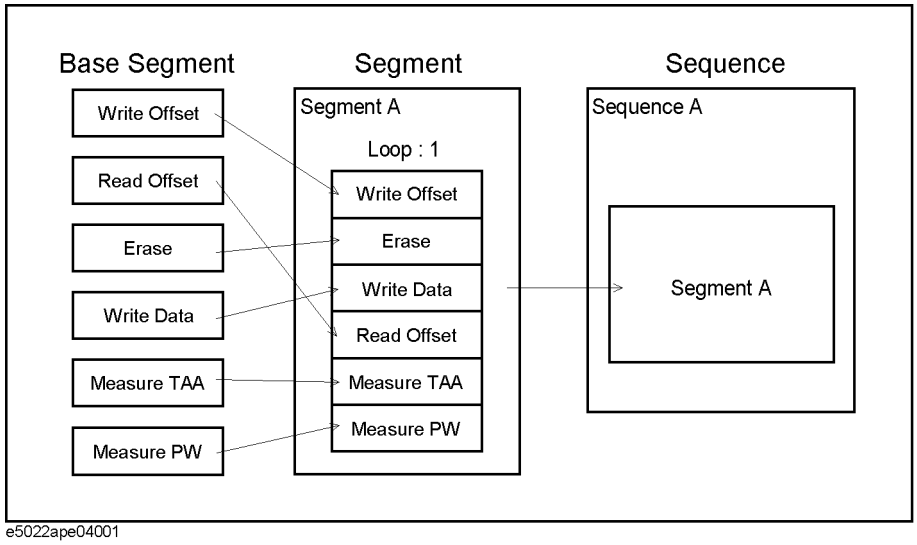
```
Private Sub CmdTaaPw_Click()

Const numOfSec As Integer = 1

Dim SeqBaseSegErase, SeqBaseSegWriteOffset As Long
Dim SeqBaseSegReadOffset, SeqBaseSegWrite As Long
Dim SeqBaseSegMeasTaa, SeqBaseSegMeasPw As Long
Dim SeqBaseSegMeasTaaId, SeqBaseSegMeasPwId As Long
Dim SeqSegWrite, dummy As Long
Dim SeqTest, SeqSegWriteId As Long
Dim loopCoun, sectCoun, elemCoun As Integer
Dim ResultTAA(1) As Double
Dim ResultPW(1) As Double


'Create Base Segment
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase)
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_ISO, VI_FALSE, hpe5022_MODULE_DATA_GEN,
SeqBaseSegWrite)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_ISO, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, _
                                        SeqBaseSegMeasTaa)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_ISO, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_PW, _
                                        SeqBaseSegMeasPw)

'SEGMENT 1
ErrorCheck hpe5022_createSeqSeg(HpE5022, 1, SeqSegWrite)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegWrite, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegMeasTaa, SeqBaseSegMeasTaaId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegMeasPw, SeqBaseSegMeasPwId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, hpe5022_SEQ_END, dummy)

'SEQUENCE 1
ErrorCheck hpe5022_createSeq(HpE5022, numOfSec, SeqTest)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegWrite, SeqSegWriteId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, hpe5022_SEQ_END, dummy)

ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest)
'PRINT QUERY DATA
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegWriteId, SeqBaseSegMeasTaaId, loopCoun, sectCoun, elemCoun)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegWriteId, SeqBaseSegMeasPwId, loopCoun, sectCoun, elemCoun)

ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegWriteId, SeqBaseSegMeasTaaId, ResultTAA(0))
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegWriteId, SeqBaseSegMeasPwId, ResultPW(0))

lbltaa.Caption = "TAA(+,-) " + Format(ResultTAA(0), "0.00E-##") + "[V]" + vbCrLf _
                             + Format(ResultTAA(1), "0.00E-##") + "[V]" + vbCrLf
lblaveTaa.Caption = "TAA " + Format((ResultTAA(0) + ResultTAA(1)), "0.00E-##") + "[V]"
lblpw.Caption = "PW  " + Format((ResultPW(0) + ResultPW(1)), "0.00E-##") + "[sec]"

ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegWrite)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWrite)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasTaa)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasPw)

End Sub
```

Table 1-8 **Segment 1 : SeqSegWrite**

Segment 1: (Loop Count =1)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegWrite	dummy	Write Offset	moves the head to write position
SeqSegWrite	dummy	Erase	erases the track
SeqSegWrite	dummy	Write	writes LF
SeqSegWrite	dummy	Read Offset	moves the head to read position
SeqSegWrite	SeqBaseSegMeasTaaId	Read	measures the TAA
SeqSegWrite	SeqBaseSegMeasPwId	Read	measures the PW
SeqSegWrite	dummy		closes the registry

Segment 1 integrates the base segment and handles the WriteOffset, Erase, Write, Read Offset and Read base segment functions for both TAA and PW. The function `hpe5022_createSeqSeg(HpE5022, 1, SeqSegWrite)` executes all base segments registered under it for one loop. All base segment IDs can be considered as dummy except for SeqBaseSegMeasTaaId of `hpe5022_addSeqBaseSeg (HpE5022, SeqSegWrite, SeqBaseSegMeasTaa, SeqBaseSegMeasTaaId)` and SeqBaseSeg MeasPwId of the `hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSegMeasPw, SeqBaseSegMeasPwId)` functions which contain the measured data. The dummy IDs are named as such because its data have no purpose of use (i.e, it does not contain the measurement results).

Figure 1-9 **Sequence 1**



e5022ape04001

As shown in Figure 1-9 the sequence has one segment registered under it. The `hpe5022_createSequence (Hpe5022, numOfSec, SeqTest)` function integrates the segment and all base segments registered with this segment. When you add a segment to a sequence using hpe5022_addSeqSeg function, make sure that its third parameter agrees with the created sequence handle. The function `hpe5022_executeSeq(HpE5022, SeqTest)` executes the program when the sequence is completed.

During its reading operation the head will read the TAA and PW for one revolution. Then it will return the values of TAA+, TAA- for Track Average Amplitude and PW+, PW- for Pulse Width measurements.

NOTE Base segments that are registered in a segment are executed in accordance to the order by which it is registered. The same rule applies for segment registered in a sequence.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sector count and elemCoun. If the selected result for elemCoun is TAA and PW, the returned value for elemCoun becomes two since two resulting values will be returned TAA+ and TAA-. If the base segment `hpe5022_createSeqBaseSeg(xxx)` is sectored it will return a value equal to sectCoun, otherwise it returns a value of one. The same for PW. LoopCoun is defined as the number of segment loops.

- hpe5022_result_Q

  This function returns the data array as the measurement results of the specified base segment. The specified base segment SeqBaseSegMeasTaaId and SeqBaseSegMeasPwId must be the same as the base segment ID returned by `hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite, SeqBaseSeg MeasTaa, SeqBaseSegMeasTaaId)` and `hpe5022_addSeqBaseSeg (HpE5022, SeqSegWrite, SeqBaseSegMeasPw, SeqBaseSegMeasPwId)` function because it contains the measured data. Refer to “User-defined Sequence Functions” of the programming manual for its detailed description.

After executing the program it is advisable that you delete all sequence, segments and base segments in order to clean up the memory from previously instored data.

NOTE The results are also dependent on the parameters set by the cassette, the drive on and set up parameters. Be sure that these parameters are set according to the type of cassette being used.

Case Study : Stability

The Stability of the head is affected by many of the head's write-read iterations. One way of checking this is to find out the degree of change of the measured Taa. In addition, we want to speed up its measurement capability, let's say instead of executing measurement for 4 loops we can cut it to 2 and still get the same number of data.

Analysis

One way to speed up measurement is to divide the track into some sectors and allow the data to be written and read within this sector. In this case we will divide the track to 2 sectors which means that data will be read twice with respect to its sector. Allow the data to be written for one revolution and be read as specified by the read ratio.

Data Requirements

Problem Input : Write ratio.

Read ratio.

Number of loops for write-read iteration.

Number of sectors.

Problem Output : Resulting stability for Taa.

Algorithm

1. Input the write ratio, read ratio and number of sectors.
2. Compute and display the measured stability Taa.

Measuring the stability can be done easily under a single function `hpe5022_measure Stability (hpe5022,hpe5022_MEAS_TAA, numOfSeg,numOfRev)`. As a user you can modify this example and create a new sequence of your own.

Implementation

To create the UDS follow the steps and procedures are listed on "Process for creating UDS:" on page 39. First create the move, erase, write offset, write, read offset and write-read base segments. Set the necessary parameters to operate each base segment function. Then integrate them to form a segment, and specify the number of write-read loop, or the number of times you want to segment function to repeat itself. In this case we have divided it into two segments, the write and read segment having one loop each. Then we integrate the two segments into a single sequence for program execution.

Description

The stability measurement allows you to observe the Taa and Pulse Width variations during many write-read iterations. For this program we will divide the track to 2 sectors and allow the data to be written and read in this sector as defined by the write ratio and read ratio.

By sectoring the track, you only read a portion of the sector as defined by read ratio. The remaining portion of the sector as defined by the write ratio will be bypassed by the read head. Note that the head takes an idle break during read and write iteration. Refer to Chapter 3 of the programming manual “Stability” for its detailed description.

Table 1-9 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Write	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 4	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 5	WriteRead	Sectoring	VI_TRUE
		Data Pattern	hpe_5022_PAT_HF (write)
		Module	hpe5022_MODULE_DATA_GEN
		Write ratio	0.4
		Data Pattern	hpe_5022_PAT_HF (read)
		Module	hpe5022_PARAM
		Read Option	hpe5022_MEAS_TAA
		Read ratio	0.4

Step 1. Move the write head to its writing position.

Step 2. Erase the track without sectoring.

Step 3. Write an HF data pattern without sectoring. The data signal is supplied by the hpe5022_MODULE_DATA_GEN.

Step 4. Move the read head to its reading position.

Step 5. Read the TAA of the HF data with sectoring. The read data is to be measured by the parametric module.

Described in the table above are five steps to form the parameters and functions of each base segment. Steps 1 to 4 constitute Segment 1 while Step 5 constitutes Segment 2. The sectored flag operation of the Write base segment (Step 3) is defined as VI_FALSE since the sequence will not be sectored during writing. The boolean of function `hpe5022_ seqBaseSegParameter (HpE5022,SeqBaseSegWrite1, hpe5022_PARAMETER_WRIT_ CURR, WriteCurrent, VI_TRUE)` is defined as VI_TRUE because the write current will only be swept once and therefore it must be fixed. You will also notice that the Write Current is defined to have a value of 0.035 [A], this is true because we want a constant write current. For stability measurement, (Step 5) base segment hpe5022_createSeqBaseSegWriteRead function will be used for reading data. And the write ratio and read ratio must be specified within the allowable limits, that is 0.2-0.8 for both write and read ratio. As defined by `hpe5022_createSeqBaseSegWriteRead(HpE5022, hpe5022_PAT_HF, VI_TRUE, hpe5022_MODULE_DATA_GEN, writeRatio, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, readRatio, SeqBaseSegstab )` function, the sector flag operation **VI_TRUE** means that the stability process will be sectored.

Refer to Chapter 4 of the programming manual for the detailed description of the various parameters and functions of each base segment.

NOTE Base segments that are registered in a segment are executed in accordance to its order of registry. The same rule applies for segment registered in a sequence.

NOTE The write ratio and read ratio must be specified within their respective limits.

Example 1-15 Sample Program (Stability)

```
Private Sub cmdStability_Click()

Const writeRatio As Double = 0.4 'Write Ratio
Const readRatio As Double = 0.4 'Read Ratio
Const WriteCurrent As Double = 0.035  'Ampere
Const numOfsec As Integer = 2

Dim SeqBaseSegErase1, SeqBaseSegWriteOffset1 As Long
Dim SeqBaseSegWrite1, SeqBaseSegReadOffset1  As Long
Dim SeqSegWrite1, dummy, SeqSegRead1, SeqBaseSegstab, stabId As Long
Dim SeqTest1, SeqSegWrite1Id, SeqSegRead1Id As Long
Dim loopCoun, sectCoun, elemCoun As Integer
Dim Display As String
Dim Display1 As String
Dim output(8) As Double

'***Create BASE SEGMENT***

'(Erase Base Segment)
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase1)
'(WriteOffset Base Segment)
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset1)
'(Write Base Segment)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_DATA_GEN,
SeqBaseSegWrite1)
'(Parameter Base Segment)
ErrorCheck hpe5022_seqBaseSegParameter(HpE5022, SeqBaseSegWrite1, hpe5022_PARAMETER_WRIT_CURR, WriteCurrent, VI_TRUE)
'(ReadOffset Base Segment)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset1)
'(Write/Read Base Segment)
ErrorCheck hpe5022_createSeqBaseSegWriteRead(HpE5022, VI_TRUE, hpe5022_PAT_HF, hpe5022_MODULE_DATA_GEN, writeRatio, _
                                        hpe5022_PAT_HF, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, _
                                        readRatio, SeqBaseSegstab)

'***Create SEGMENT ***
'(Segment 1)
ErrorCheck hpe5022_createSeqSeg(HpE5022, 1, SeqSegWrite1)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite1, SeqBaseSegWriteOffset1, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite1, SeqBaseSegErase1, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite1, SeqBaseSegWrite1, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite1, SeqBaseSegReadOffset1, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWrite1, hpe5022_SEQ_END, dummy)

'(Segment 2)
ErrorCheck hpe5022_createSeqSeg(HpE5022, 2, SeqSegRead1)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegRead1, SeqBaseSegstab, stabId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegRead1, hpe5022_SEQ_END, dummy)

'***Create SEQUENCE***
ErrorCheck hpe5022_createSeq(HpE5022, numOfsec, SeqTest1)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest1, SeqSegWrite1, SeqSegWrite1Id)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest1, SeqSegRead1, SeqSegRead1Id)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest1, hpe5022_SEQ_END, dummy)

ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest1)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest1)
```

```
'PRINT QUERY DATA
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest1, SeqSegRead1Id, stabId, loopCoun, sectCoun, elemCoun)
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest1, SeqSegRead1Id, stabId, output(0))

'Display Measurement Result of TAA
Display1$ = "TAA Measurement :" + vbCrLf _
          + "sector1,loop1, elem1 :" + Format(output(0), "0.00E-##") + "[V]" + vbCrLf _
          + "sector1 loop1, elem2 :" + Format(output(1), "0.00E-##") + "[V]" + vbCrLf _
          + "sector1 loop2, elem1 :" + Format(output(2), "0.00E-##") + "[V]" + vbCrLf _
          + "sector1 loop2, elem2 :" + Format(output(3), "0.00E-##") + "[V]" + vbCrLf _
          + "sector2 loop1, elem1 :" + Format(output(4), "0.00E-##") + "[V]" + vbCrLf _
          + "sector2 loop1, elem2 :" + Format(output(5), "0.00E-##") + "[V]" + vbCrLf _
          + "sector2 loop2, elem1 :" + Format(output(6), "0.00E-##") + "[V]" + vbCrLf _
          + "sector2 loop2, elem2 :" + Format(output(7), "0.00E-##") + "[V]" + vbCrLf _
          + "TAA of (sector1, loop1) :" + Format(output(0) + output(1), "0.00E-##") + "[V]" + vbCrLf _
          + "TAA of (sector1, loop2) :" + Format(output(2) + output(3), "0.00E-##") + "[V]" + vbCrLf _
          + "TAA of (sector2, loop1) :" + Format(output(4) + output(5), "0.00E-##") + "[V]" + vbCrLf _
          + "TAA of (sector2, loop2) :" + Format(output(6) + output(7), "0.00E-##") + "[V]" + vbCrLf


lblresult.Caption = Display1$

ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest1)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest1)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegWrite1)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegRead1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWrite1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegstab)


End Sub
```

Table 1-10 **Segment 1 : SeqSegWrite1**

Segment 1: (Loop Count =1)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegWrite1	dummy	Erase	erases the track
SeqSegWrite1	dummy	Write Offset	moves the head to its writing position
SeqSegWrite1	dummy	Write	writes HF
SeqSegWrite1	dummy	Read Offset	moves the head to its reading position
SeqSegWrite1	dummy		closes the registry

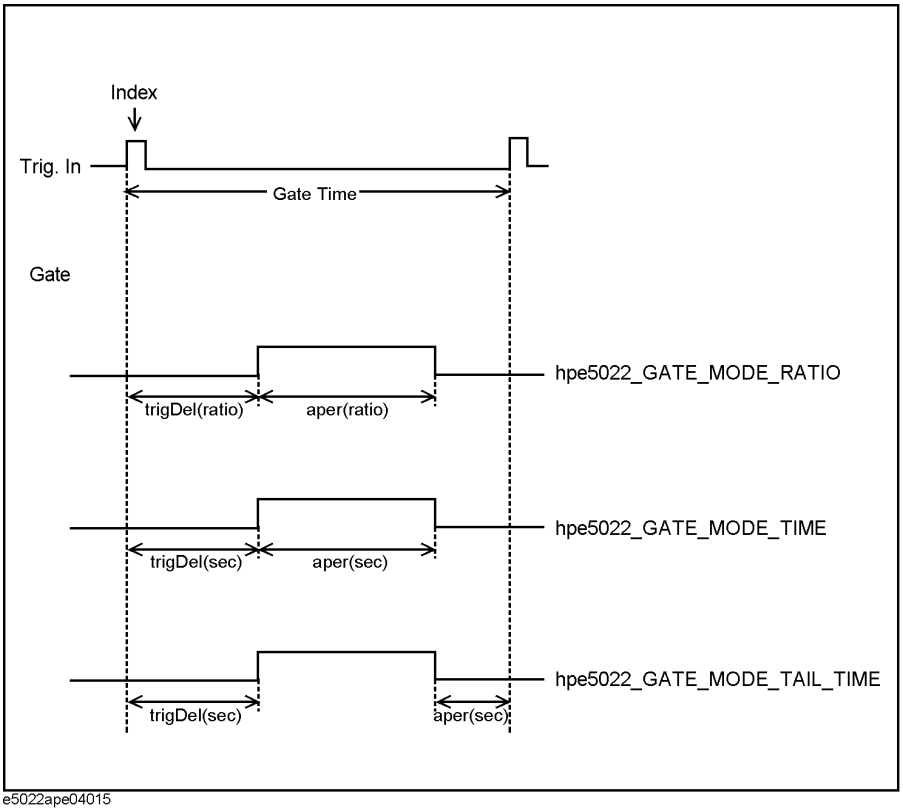
Segment 1 integrates the base segment and handles the Erase, Write Offset,Write and Read Offset base segment functions. The function `hpe5022_createSeqSeg (HpE5022, 1, SeqSegWrite1)` erases a track and writes the HF data for this segment. All base segment IDs will be considered as dummy. They are named as such because its data has no purpose of use.

Table 1-11 **Segment 2 : SeqSegRead1**

Segment 2: (Loop Count =2)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegRead1	stabId	Write&Read	measures the TAA
SeqSegRead1	dummy		closes the registry

Segment 2 integrates the base segment and handles the Write&Read base segment function. The function `hpe5022_createSeqSeg (HpE5022, 1, SeqSegRead1)` reads the written data once. The **stabId** of the `hpe5022_addSeqBaseSeg(HpE5022, SeqSegRead1, SeqBaseSegstab,stabId)` function contains the measured data. During its reading process only 40 percent of the data will be read per sector as defined by the read ratio and `hpe5022_createSeq(HpE5022, numOfSec, SeqTest1)` number of sector.

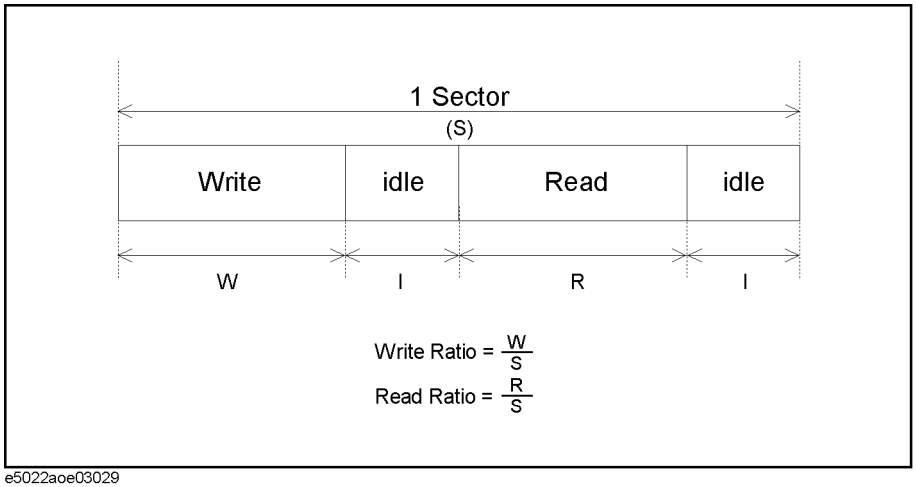
Figure 1-10 **Sequence 1 : SeqTest1**



The figure above shows Sequence 1 having two segments . The sequence function `hpe5022_createSequence (Hpe5022,numOfSec, SeqTest1)` integrates the segment and all base segments registered with it for stability measurement. You will notice that there are two sectors for this measurement as defined by `hpe5022_createSequence(Hpe5022,numOfSec, SeqTest1)` this will translate to a speedy stability measurement. When you add a segment to a sequence using hpe5022_addSeqSeg function make sure that its third parameter agrees with the created sequence handle. The function `hpe5022_executeSeq` executes the program under test.

The figure below shows how the read and write ratios are situated along the sector.

Figure 1-11 **Sectoring**



During its reading process the read head will only read a portion of the written data as defined by read ratio, in this case 40 percent (0.4) and will bypass the remaining 40 percent of write ratio. The remaining 20 percent will be considered as idle between ratios.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sector count and elemCoun. In this case, the returned value for elemCoun becomes two since two resulting values will be returned TAA+ and TAA-. LoopCoun is defined as the number of segment loops. If the base segment `hpe5022_createSeqBaseSeg(xxxx)` is sectored it will return a value equal to sectCoun, otherwise it returns a value of one. In this example it is equal to 2 which means the track will be divided into two sectors translating for speedy stability measurement. The more sectors you divide the track the shorter time it takes to measure. If you have an unsectored eight loop count to measure the data, by sectoring the track to two this will cut the measurement time by half provided that the loop count is 4. Refer to “User Defined Sequence Functions” of the programming manual for its detailed description.

- hpe5022_result_Q

  This function returns the array data as the measurement result of the specified base segment. The specified base segment stabId should be the same as the “basId” returned by `hpe5022_addSeqBaseSeg(HpE5022, SeqSegRead1, SeqBaseSegstab, stabId)`. Refer to the “User Defined Sequence Functions” of the programming manual for its detailed description.

After executing the program it is advisable that you delete all sequence, segments and base segments in order to clean up the memory from the previously instored data.

NOTE The result also depends on the parameters set on the cassette, the drive on and set up parameters. Be sure that these parameters are set according to the type of cassette being used.

Case Study : Write Current Sweep

Write a program that computes the track average amplitude twice (TAA) such that write current is increased four times at a rate of 10[mA] /rev.

Analysis

To solve this problem, we must find a way to form the number of loops where TAA is measured for every increase in write current.

Data Requirements

Problem Input : Write current start.

Write current stop.

Number of loops.

Problem Output : Measure Taa (Taa+, Taa-).

Algorithm

1. Input the write current values, number of loops and other parameters.
2. Compute and display the average Taa.

As mentioned in Chapter 3 of the programming manual write current sweep can also be measured under a single function hpe5022_measureWriteCurrent Sweep(xxxx). However, its measuring sequence ability is limited. Under UDS a user can have the freedom to create his own sequence. For example, if you want to sweep the write current four times and measure the Taa twice for every sweep and get the average. This can be done under UDS, where as before the number of Taa results will be equal to the number of loops or sweep points. Now the user has the option to modify a measurement sequence that was impossible before.

For simplicity, the UDS program for Write Current Sweep has been provided, in such a way that the user can do some modifications. You can modify the constant parameters. You can also modify the sequence, let's say you want to have a three write current sweep and measure the Taa four times for each sweep.

Implementation

To create the UDS follow the steps and procedures listed on "Process for Creating UDS"on page 39. First we have to create the base segment such as move, erase, write offset, write, read offset and read. And set the necessary parameters to operate them as defined by each base segment function. Then we integrate them into a segment, where we define the number of loops, the number of sweeps, or as to how many times we want to execute the base segments registered under each segment. Also, the required parameters to execute these segment must be defined. After that, we integrate the segment into a sequence for program execution.

Description

The purpose of this measurement is to provide a design tool for selecting the proper write current to optimize the performance of the head/disk combination.

This is a typical example of a write current sweep. The current is swept at different values as it writes data along the track. As write current increases the resulting flux imparted on the media will also increase in depth and latitude. This magnetization is sensed by the parameter under test.

Table 1-12 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Write	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 4	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 5	Read (2 times)	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_PARAM
		Read Option	hpe5022_MEAS_TAA

Step 1. Move the write head to its writing position.

Step 2. Erase the track without sectoring.

Step 3. Write an HF data pattern without sectoring. The data signal is supplied by the hpe5022_MODULE_DATA_GEN.

Step 4. Move the read head to its reading position.

Step 5. Read the TAA of the HF data pattern twice without sectoring. The read data is to be measured by the parametric module.

Listed in the table above are five steps to follow in order to create the necessary base segments. The sectored flag operation is defined as VI_FALSE because of its unsectored operation. Since the written data is HF (Step 3) it should also be read as HF, as defined by `hpe5022_createSeqBaseSegRead (HpE5022, **hpe5022_PAT_HF**, VI_FALSE, hpe5022_MODULE_ PARAM, hpe5022 _MEAS_TAA, seqBaseSegMeasTaa)` base segment. If you use **`hpe5022_PAT_DEFAULT`** as your data pattern, the selected pattern in the hpe5022_selectPattern function will be used. For Step 5 there is no need to create two separate read base segments, instead we will designate two base segment IDs for one base segment handle to distinct one from the other.

Example 1-16 Sample Program (Write Current Sweep)

```
Private Sub cmdWriteCurrSweep_Click()

Const start As Double = 0#
Const WRITE_CURR_SWEEP_NOP As Integer = 4
Const SEG_WRIT_CURR_STOP As Double = 0.03
Const numOfSec As Integer = 1

Dim SeqBaseSegWriteOffset, SeqBaseSegErase, seqBaseSegWcsWrite As Long
Dim SeqBaseSegMove, SeqBaseSegReadOffset, SeqSegWriteCurrSweep, dummy As Long
Dim SeqBaseSegMeasTaa, SeqBaseSegMeasTaa1Id As Long
Dim SeqBaseSegMeasTaa2Id   As Long
Dim SeqTest, SeqSegWriteCurrSweepId As Long
Dim i, loopCoun, sectCoun, elemCoun As Integer
Dim res1(8) As Double
Dim res2(8) As Double


'Create  Base Segment
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         seqBaseSegWcsWrite)
ErrorCheck hpe5022_seqBaseSegParameter(HpE5022, seqBaseSegWcsWrite, hpe5022_PARAMETER_WRIT_CURR, 0#, VI_FALSE)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, _
                                        SeqBaseSegMeasTaa)

'SEGMENT 1
ErrorCheck hpe5022_createSeqSeg(HpE5022, WRITE_CURR_SWEEP_NOP, SeqSegWriteCurrSweep)
ErrorCheck hpe5022_seqSegParameterSweep(HpE5022, SeqSegWriteCurrSweep, hpe5022_PARAMETER_WRIT_CURR, start, _
                                        SEG_WRIT_CURR_STOP)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, seqBaseSegWcsWrite, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, SeqBaseSegMeasTaa1Id)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, SeqBaseSegMeasTaa2Id)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, hpe5022_SEQ_END, dummy)

'SEQUENCE
ErrorCheck hpe5022_createSeq(HpE5022, numOfSec, SeqTest)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegWriteCurrSweep, SeqSegWriteCurrSweepId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, hpe5022_SEQ_END, dummy)

ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest)


'PRINT QUERY DATA
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegWriteCurrSweepId, SeqBaseSegMeasTaa1Id, loopCoun, _
                                 sectCoun, elemCoun)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegWriteCurrSweepId, SeqBaseSegMeasTaa2Id, loopCoun, _
                                 sectCoun, elemCoun)

ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegWriteCurrSweepId, SeqBaseSegMeasTaa1Id, res1(0))
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegWriteCurrSweepId, SeqBaseSegMeasTaa2Id, res2(0))
```

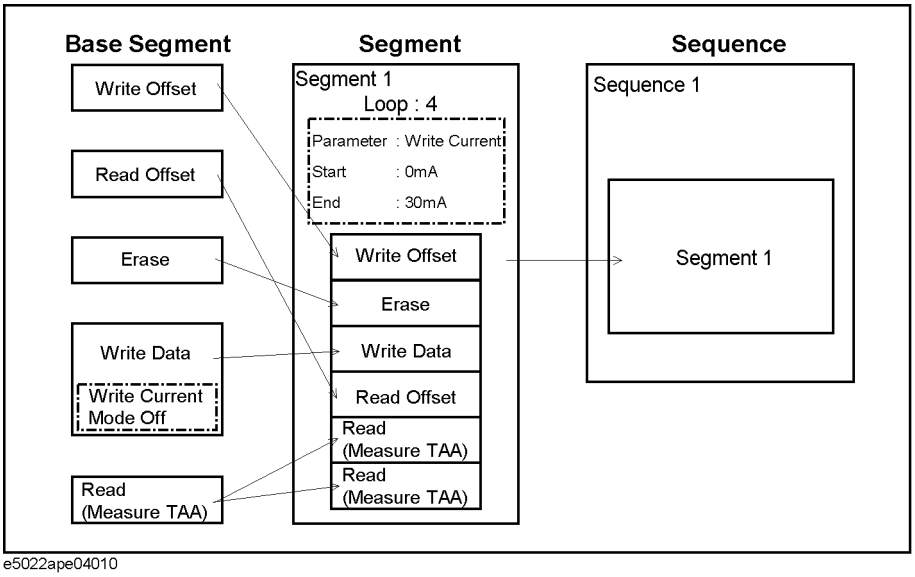
```
'Measure Average Taa
lbltaa.Caption = "TAA Loop1 " + Format(res1(0), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res1(1), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res2(0), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res2(1), "0.00E-##") + "[V]" + vbCrLf _

lbltaa.Caption = "TAA Loop2 " + Format(res1(2), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res1(3), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res2(2), "0.00E-##") + "[V]" + vbCrLf _
                        + Format(res2(3), "0.00E-##") + "[V]" + vbCrLf _


lblaveTaa.Caption = "Ave.TAA for Loop1  " + Format((res1(0) + res1(1)), "0.00E-##") + "[V]" + vbCrLf _
                                          + Format((res2(0) + res2(1)), "0.00E-##") + "[V]"


ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegWriteCurrSweep)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, seqBaseSegWcsWrite)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasTaa)

End Sub
```

Table 1-13 **Segment 1 : SeqSegWriteCurrSweep**

Segment 1: (Loop Count = 4)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegWriteCurrSweep	dummy	Write Offset	moves the head for writing position
SeqSegWriteCurrSweep	dummy	Erase	erase a track
SeqSegWriteCurrSweep	dummy	Write	writes HF
SeqSegWriteCurrSweep	dummy	Read Offset	moves the head for reading position
SeqSegWriteCurrSweep	SeqBaseSeg MeasTaa1Id	Read	reads the HF data
SeqSegWriteCurrSweep	MeasTaa2Id SegBaseSeg	Read	reads the HF data
SeqSegWriteCurrSweep	dummy		closes the registry

Segment 1 integrates the base segment and handles the WriteOffset, Erase,Write, Read Offset and Read base functions. Except for seqBaseSegMeasTaa1Id of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, SeqBaseSegMeasTaa1Id)` function and SeqBaseSegMeasTaa2Id of the `hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, SeqBaseSegMeasTaa2Id)` function which contains the read data, all other base segment IDs can be considered as dummy since it has no purpose of use. The function `hpe5022_create SeqSeg(HpE5022,WRITE_CURR_SWEEP_NOP, SeqSegWriteCurrSweep)` creates a four loop count as it writes the data, starting from 0[mA] to 30[mA] during which current increases by 10[mA] for every loop. And in each loop Taa is read twice by the read head as shown in Sequence 1 illustration.

Figure 1-12 **Sequence1**



The figure above illustrates how the sequence is created. The sequence function `hpe5022_createSequence (Hpe5022, numOfSec, SeqTest)` integrates the segment that contain base segment functions needed for measurement. When you add a segment to a sequence using hpe5022_addSeqSeg function make sure that its third parameter agrees with the created sequence handle. The function `hpe5022_ setupSeq(HpE5022, SeqTest)` and `hpe5022_ executeSeq (Hpe5022,SeqTest)` sets and executes the sequence program respectively.

NOTE The Common Parameter Setup, Drive On and Auto Configuration must be executed first before executing the program. Without the spinstand running the program will result to an error.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sectCoun and elemCoun. If the selected result for elemCoun is TAA, the returned value of elemCoun becomes two since two resulting values will be returned TAA+ and TAA-. LoopCoun is defined as the number of segment loops. If the base segment `hpe5022_createSeqBaseSeg(xxxx)` is sectored it will return a value equal to sectCoun, otherwise it returns a value of one.

  Refer to the “User Defined Sequence Functions” of the programming manual for its detailed description.

- hpe5022_result_Q

  This function returns the array data as the measurement result of the specified base segment. The specified base segment IDs **MeasTaa1Id** and **MeasTaa12d** should be the same as the "basId" returned by " `hpe5022_addSeqBaseSeg (HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, seqBaseSegMeasTaa1Id)` and `hpe5022_addSeqBaseSeg(HpE5022, SeqSegWriteCurrSweep, SeqBaseSegMeasTaa, seqBaseSegMeasTaa2Id)` function because it contains the measured data. Refer to "User Defined Sequence Functions" of the programming manual for its detailed description.

After executing the program it is advisable that you delete all sequences, segments and base segments in order to clean up the memory from previously instored data.

NOTE The results are also dependent on the parameters set by the cassette, the drive on and set up parameters. Be sure that these parameters are set according to the type of cassette being used.

Case Study : Overwrite

Overwrite is used to determine the amount of remaining residual signal from the previous write, when new data is written over it. One way of doing this, is to compute the residual LF after it is overwritten by an HF data pattern. In addition, a three track erase is also required.

Analysis

We must find a way to overwrite a data pattern with another data pattern and measure the residual component of the overwritten data.

Data Requirements

Problem Input : Write an LF data pattern.

Write an HF data pattern.

Number of loops for three track erase (five).

Equation for solving the Overwrite ratio.

Problem Output : Read the old and new LF data pattern.

Display the measured overwrite ratio.

Algorithm

1. Input the write data pattern and other parameters.
2. Read, compute and display the overwrite ratio for a single loop.

Measuring the overwrite ratio can be done easily under a single function as defined by `hpe5022_measureOverwrite (HpE5022, NofRev)`. However, this function has the disadvantage of returning the same number of values for the number of loops. If the number of write is equal to the number of to read then, the programming function `hpe5022_measureOverwrite (HpE5022, NofRev)` described in Chapter 3 of the Function Reference can be used. If not, then UDS is the alternative solution. As mentioned before UDS gives the user the flexibility to create his own measurement sequence.

Implementation

To create the UDS follow the steps and procedures listed on “Process for Creating UDS”on page 39. First create the base segment such as move, erase, write offset, write, read offset and read. And set the necessary parameters to operate them as defined by their respective base segment function. Then we integrate them into a segment, where we specify the desired number of loops, sweep, or as to how many times we execute the base segments registered under it. Since the number of loops for erase is different from measurement, we decided to have two segments, three track erase and overwrite segment. Then we integrate these two segments into a single sequence for program execution.

Description

To prevent errors in a disk drive, a head/disk combination must be capable of reducing the remnant signal to an acceptable level. When old data is overwritten with new data remanents of the previous data remain they are not completely erased. To Overwrite, write the LF data for one revolution and measure the LF rms signal amplitude. Overwrite it with HF data for one revolution, ensuring that all the original LF data has been overwritten, then measure the residual LF.

Table 1-14 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Move	Module	hpe5022_MODULE_PIEZO
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 4	Write LF Pattern	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 5	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 6	Read (original)	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_SA
		Read Option	hpe5022_MEAS_LEVEL
Step 7	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 8	Write HF Pattern	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 9	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 10	Read (residual)	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_SA
		Read Option	hpe5022_MEAS_LEVEL

Step 1. Move the head to erase track offset.

Step 2. Erase the track without sectoring. To erase the track the erase current must be supplied by the hpe5022_MODULE_DAT_GEN.

Step 3. Move the head to its write track offset.

Step 4. Write an LF data pattern without sectoring. To write the data, the write current must be supplied by the hpe5022_MODULE_DATA_GEN.

Step 5. Move the head to its read track offset.

Step 6. Read the written LF data pattern without sectoring. Data is to be measured by a spectrum analyzer.

Step 7. Move the head to its write track offset.

Step 8. Write an HF data pattern without sectoring. To write the data, the write current must be supplied by the hpe5022_MODULE_DATA_GEN.

Step 9. Move the head to its read track offset.

Step 10. Read the written HF data pattern without sectoring. Data is measured by a spectrum analyzer.

Described in the table above are ten steps to follow in order to register base segment into a segment with its required parameters. Steps 1 and 2 constitutes Segment 1 while Steps 3 to 10 constitutes Segment 2. It can be noticed that we created two write base segments, Write LF Pattern (Step 4) and Write HF Pattern (Step 8).This is true because the objective of our program is to measure the ratio of LF versus HF data pattern, that is residual component of LF when it is overwritten with an HF data pattern.The sectored flag operation is defined as VI_FALSE for steps 2,4,6,8 and 10 due to the absence of sectoring.

Since the module for read base segment is a spectrum analyzer the data pattern is irrelevant. Hence, there is no need to create an independent read base segment as Step 8, by creating two base segment IDs for Step 6 we will be able to execute the program under a single base segment handle.

Refer to Chapter 3 of the programming manual for the detailed description of various parameters and functions of each base segment.

NOTE

```
hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_LF, VI_FALSE,
hpe5022_MODULE_SA, hpe5022_MEAS_LEVEL, seqBaseSegMeasTaa)
```

For Overwrite read base segment, spectrum analyzer will be used as its module because the read option is Measure Level.

Example 1-17 Sample Program (Overwrite)

```
Private Sub cmdOverWrite_Click()

Const StartMove As Double = -0.000002    'meter
Const StopMove As Double = 0.000002 'meter
Const numOfSec As Integer = 1

Dim SeqBaseSegMove, SeqBaseSegWriteOffset, SeqBaseSegErase As Long
Dim SeqBaseSegWriteLF, SeqBaseSegReadOffset, SeqBaseSegMeasOverWrite As Long
Dim SeqBaseSegWriteHF, SeqSegThreeTrkErase, dummy As Long
Dim SeqSegOverwrite, SeqBaseSegMeasOriginalId, SeqBaseSegMeasResidualId As Long
Dim SeqTest, SeqSegThreeTrkEraseId, SeqSegOverwriteId As Long
Dim Original(1) As Double
Dim Residual(1) As Double
Dim ChannelBitRate As Double
Dim lf_t As Integer
Dim value As Double
Dim loopCoun, sectCoun, elemCoun As Integer

'Create Base Segment
ErrorCheck hpe5022_createSeqBaseSegMove(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegMove)
ErrorCheck hpe5022_seqBaseSegParameter(HpE5022, SeqBaseSegMove, hpe5022_PARAMETER_HEAD_POS, 0#, VI_FALSE)
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_LF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         SeqBaseSegWriteLF)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_LF, VI_FALSE, hpe5022_MODULE_SA, hpe5022_MEAS_LEVEL, _
                                        SeqBaseSegMeasOverWrite)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         SeqBaseSegWriteHF)

'SEGMENT 1
ErrorCheck hpe5022_createSeqSeg(HpE5022, 5, SeqSegThreeTrkErase)
ErrorCheck hpe5022_seqSegParameterSweep(HpE5022, SeqSegThreeTrkErase, hpe5022_PARAMETER_HEAD_POS, StartMove,
StopMove)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegThreeTrkErase, SeqBaseSegMove, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegThreeTrkErase, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegThreeTrkErase, hpe5022_SEQ_END, dummy)


'SEGMENT 2
ErrorCheck hpe5022_createSeqSeg(HpE5022, 1, SeqSegOverwrite)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegWriteLF, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasOriginalId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegWriteHF, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasResidualId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, hpe5022_SEQ_END, dummy)

'SEQUENCE
ErrorCheck hpe5022_createSeq(HpE5022, numOfSec, SeqTest)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegThreeTrkErase, SeqSegThreeTrkEraseId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegOverwrite, SeqSegOverwriteId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, hpe5022_SEQ_END, dummy)

ErrorCheck hpe5022_channelBitRate_Q(HpE5022, ChannelBitRate)
ErrorCheck hpe5022_overwriteLfPattern_Q(HpE5022, lf_t)
value = ChannelBitRate / (2 * lf_t)

ErrorCheck hpe5022_seqConfiguration(HpE5022, SeqTest, hpe5022_CONFIG_SA_CENT_FREQ, value)
ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest)

ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegOverwriteId, SeqBaseSegMeasOriginalId, loopCoun, _
                                 sectCoun, elemCoun)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegOverwriteId, SeqBaseSegMeasResidualId, loopCoun, _
                                 sectCoun, elemCoun)
```

```
'Original Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegOverwriteId, SeqBaseSegMeasOriginalId, Original(0))
'Residual Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegOverwriteId, SeqBaseSegMeasResidualId, Residual(0))
'Overwrite Ratio
lblOW.Caption = " Overwrite Ratio" + Format(((Residual(0) - Original(0))), "0.00E-##") + "dBm"

ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegThreeTrkErase)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegOverwrite)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMove)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteLF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteHF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasOverWrite)

End Sub
```

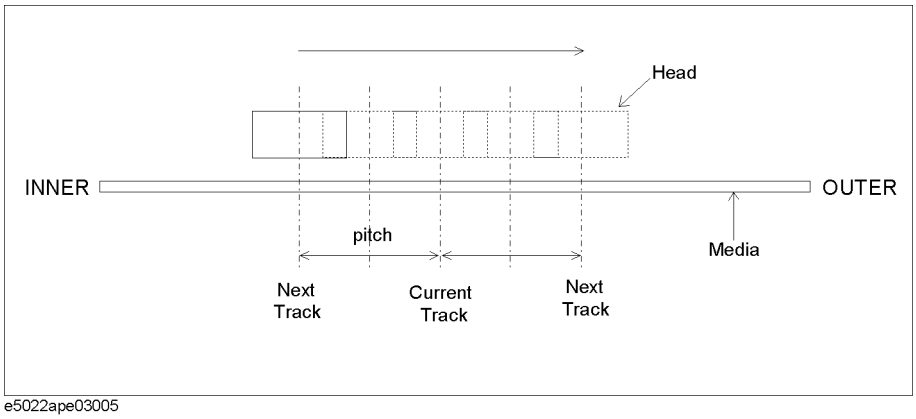
Table 1-15 **Segment 1 : SeqSegThreeTrkErase**

Segment 1: (Loop Count =5)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegThreeTrkErase	dummy	Move	moves the head
SeqSegThreeTrkErase	dummy	Erase	erases three track
SeqSegThreeTrkErase	dummy		closes the registry

Segment 1 integrates the base segment required for three track erase as defined by function `hpe5022_createSeqSeg (HpE5022,5,SeqSegThreeTrkErase)`. To accomplish this task the segment must execute a five loop rotation.

The figure below shows how a three track erase is performed.Three tracks are erased as defined by the track pitch starting from the inner track to the outer track.

Figure 1-13 **Three Track Erase**



e5022ape03005

Table 1-16 **Segment 2 : SeqSegOverwrite**

Segment 2 : (Loop Count = 1)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegOverwrite	dummy	WriteOffset	moves the head to its writing position
SeqSegOverwrite	dummy	Write LF	writes the LF data
SeqSegOverwrite	dummy	Read Offset	moves the head to its reading position
SeqSegOverwrite	SeqBaseSeg MeasOriginalId	Read (original)	reads the LF data
SeqSegOverwrite	dummy	Write Offset	moves the head to its writing position
SeqSegOverwrite	dummy	Write HF	writes the HF data
SeqSegOverwrite	dummy	Read Offset	moves the head to its reading position
SeqSegOverwrite	SeqBaseSeg MeasResidual1Id	Read (residual)	reads the LF data
SeqSegOverwrite	dummy		closes the registry

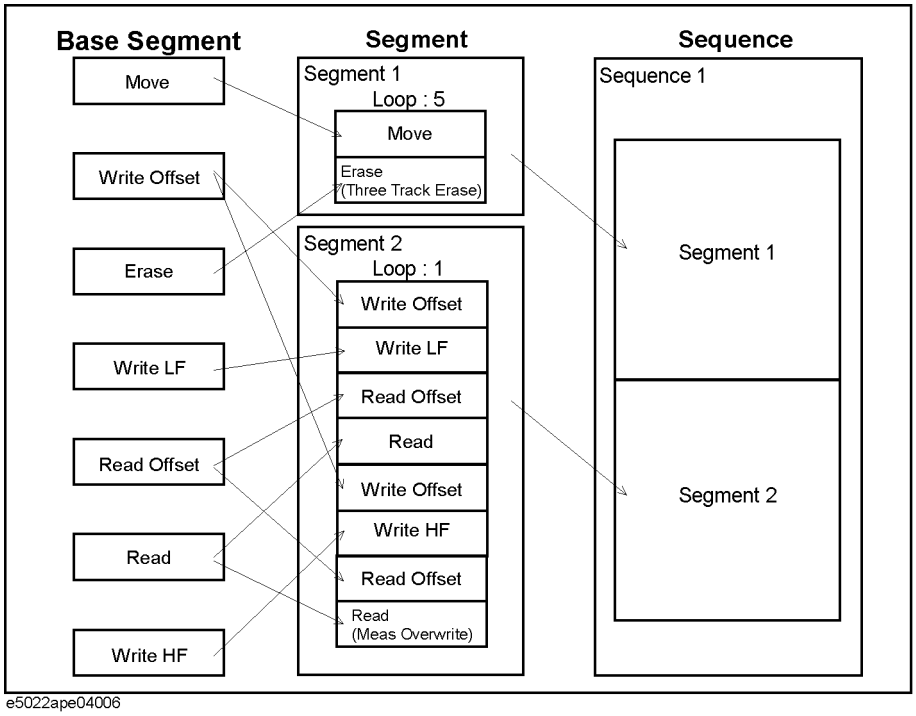
Segment 2 integrates the base segment and handles the WriteOffset , Write (LF), Read Offset, Read (original), Write Offset, Write (HF), Read Offset and Read (residual) base segment functions. Except for SeqBaseSegMeasOriginalId of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasOriginald)` and `SeqBaseSegMeasResidualId of the hpe5022_addSeqBaseSeg (HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasResidualId)` functions which contain the read data, all other base segment IDs can be considered as dummy since it has no purpose of use.

After executing the program it is recommended that you delete all sequences, segments and base segments in order to clean up the memory from the previously stored data.

NOTE Base segments that are registered in a segment are executed in accordance to its order of registry. The same rule for applies for segment registered in a sequence.

NOTE If you want to create multiple segments do it separately. It is invalid to add a segment in another segment. The same rule applies for multiple sequence.

Figure 1-14 **Sequence 1**



The figure above shows how Sequence 1 is formed, it contains two segments The sequence function hpe5022_createSequence(Hpe5022, numOfSec, SeqTest) integrates segment1 and segment2 which contain the base segments needed for measurement. When you add a segment into a sequence using hpe5022_addSeqSeg function, make sure that its third parameter agrees with the created sequence handle. Before executing the sequence, specify `hpe5022_seqConfiguration (Hpe5022,SeqTest, hpe5022_CONFIG_SA_CENT_FREQ, value)` in order to set the measurement frequency. The function `hpe5022_setupSeq(HpE5022, SeqTest)` and `hpe5022_executeSeq(HpE5022, SeqTest)` sets and executes the UDS program respectively.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sector count and elemCoun. Since the read option base segment is hpe5022_MEAS_LEVEL the returned value by elemCoun becomes one. LoopCoun is defined as the number of segment loops. If the base segment `hpe5022_createSeqBaseSeg(xxxx)` is sectored it will return a value equal to sectCoun, otherwise it returns a value of one. Refer to “User Defined Sequence Functions” of the programming manual for the detailed description.

- hpe5022_result_Q

  This function returns the array data as the measurement result of the specified base segment.The specified base segment **MeasOriginalId** and **MeasResidualId** should be the same as the "basId" returned by "`hpe5022_addSeqBaseSeg (HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasOriginalId)` and `hpe5022_addSeqBaseSeg(HpE5022, SeqSegOverwrite, SeqBaseSegMeasOverWrite, SeqBaseSegMeasResidualId)` function because it contains the measured data. Note that the unit of measurement returned by Residual( ) and Original( ) are expressed in [dbm] that's why subtraction is used. Refer to "User Defined Sequence Functions" of the programming manual for its detailed description.

NOTE The results are also dependent on the parameters set by the cassette, drive on and set up parameters. Be sure that these parameters are set according to the type of cassette being used.

Case Study : Signal To Noise Ratio

SNR measures the ratio between the amplitude of an HF signal written on a given track and the RMS value of the signal after the track is erased.

Analysis

To measure the noise of a particular track, a complete erasure of the track must be done before you overwrite it with an HF data pattern.

Data Requirements

Problem Input : Perform a dc erase track.

Write an HF data pattern.

Problem Output : Read the dc erased signal.

Read the HF data pattern.

Algorithm

1. Perform a dc erased track.
2. Write the HF data pattern and other base segment parameters.
3. Compute and display the SNR ratio.

Measuring the SNR can be done in two ways either by using the `hpe5022_measureSnr(HpE5022, NofRev)` function or UDS. UDS offers a more flexible measurement capability than the non-UDS sample program. It allows the user to pre-define the sequence of measurement as desired.

Implementation

To create the UDS follow the steps and procedures listed on “Process for Creating UDS”on page 39. In order to apply UDS you have to convert the algorithm into a programming language. This can be done by, creating the necessary base segments with its parameters as defined by hpe5022_createSeqBaseSeg(xxx) function. Then register the created base segments into a segment with its operating parameters. The next step is to register the segment into a sequence for program execution.

Description Total noise is a major contributor to poor error rate performance in disk recording system. Disk noise is known to be a significant portion of the total noise. A reasonable signal to noise ratio is necessary to keep random bit displacement within acceptable limits.

Table 1-17 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Write Offset	Module	hpe5022_MODULE_DATA_GEN
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 4	Read (noise)	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_PARAM
		Read Option	hpe5022_MEAS_TAA
Step 5	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 6	Write HF Pattern	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 7	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 8	Read (signal)	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_PARAM
		Read Option	hpe5022_MEAS_TAA

Step 1. Move the head to its write track offset.

Step 2. Erase the track without sectoring. To erase the track the erase current must be supplied by the hpe5022_MODULE_DAT_GEN.

Step 3. Move the read head to read track offset.

Step 4. Read the track after erasing it. The read data is measured by the parametric module.

Step 5. Move the head to its write track offset.

Step 6. Write the HF data pattern without sectoring. To write the data, the write current must be supplied by the hpe5022_MODULE_DATA_GEN.

Step 7. Move the head to its read track offset.

Step 8. Read the written HF data pattern without sectoring. The data is measured by the parametric module.

The table above describes the pre-defined sequence of an SNR measurement. Steps 1 to 8 constitute segment 1. Step 2 erases the track after setting the head to write offset position. Then it reads the erased track signal and measures the signal by parametric module. At step 6 an HF data pattern is written over it. Finally with Step 8 written HF signal is measured. From these two results you can get the SNR ratio.However, the resulting signal will be measured in [dbm]. Sectoring is defined as VI_FALSE for all base segments.

Example 1-18 Sample Program(SNR)

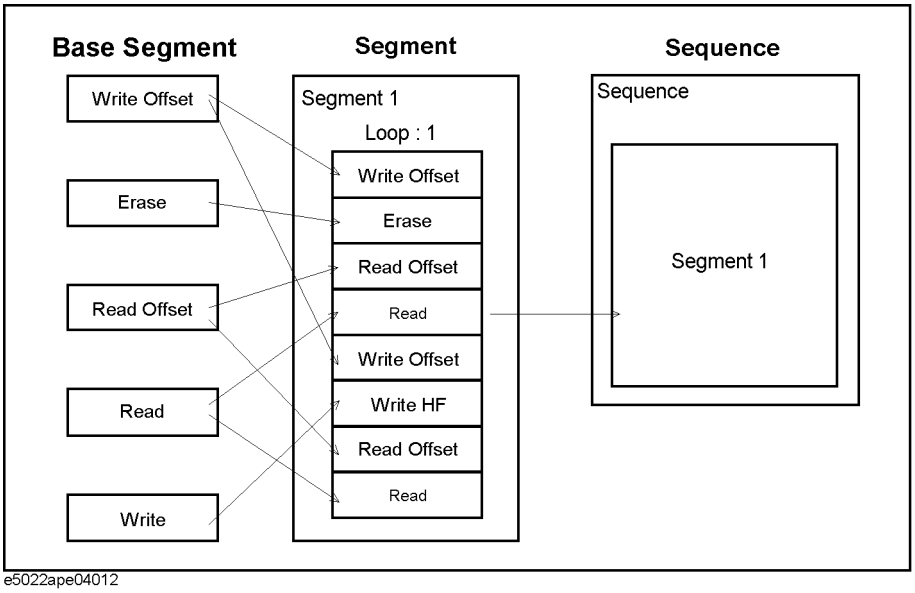
```
Private Sub cmdSNR_Click()

Const numOfSec As Integer = 1

Dim SeqBaseSegWriteOffset, SeqBaseSegErase, SeqBaseSegWriteHF As Long
Dim SeqBaseSegReadOffset, SeqBaseSegMeasSNR1, SeqBaseSegMeasSNR2 As Long
Dim SeqSegSNR, dummy As Long
Dim SeqBaseSegNoiseId, SeqBaseSegHFId, SeqTest, SeqSegSNRId As Long
Dim loopCoun, sectCoun, elemCoun As Integer
Dim Noise(1) As Double
Dim HF(1) As Double


'Create Base Segment
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         SeqBaseSegWriteHF)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_PARAM, _
                                        hpe5022_MEAS_TAA, SeqBaseSegMeasSNR1)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_PARAM, _
                                        hpe5022_MEAS_TAA, SeqBaseSegMeasSNR2)

'SEGMENT 1
ErrorCheck hpe5022_createSeqSeg(HpE5022, 1, SeqSegSNR)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegMeasSNR1, SeqBaseSegNoiseId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegWriteHF, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegMeasSNR2, SeqBaseSegHFId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, hpe5022_SEQ_END, dummy)

'SEQUENCE 1
ErrorCheck hpe5022_createSeq(HpE5022, numOfSec, SeqTest)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegSNR, SeqSegSNRId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, hpe5022_SEQ_END, dummy)

ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegSNRId, SeqBaseSegNoiseId, loopCoun, sectCoun, elemCoun)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegSNRId, SeqBaseSegHFId, loopCoun, sectCoun, elemCoun)

'Noise Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegSNRId, SeqBaseSegNoiseId, Noise(0))
'HF Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegSNRId, SeqBaseSegHFId, HF(0))
'SNR Ratio
lblOW.Caption = "SNR Ratio" + Format((20 * (Log(HF(0) / Noise(0)) / Log(10))) + _
                                     (20 * (Log(HF(1) / Noise(1)) / Log(10))), "0.00E-##") + "dbm"


ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegSNR)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteHF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasSNR1)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasSNR2)

End Sub
```

Table 1-18 **Segment 1 : SeqSegSNR**

Segment 1: (Loop Count =1)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegSNR	dummy	Write Offset	moves the head to its writing position
SeqSegSNR	dummy	Erase	erases the track
SeqSegSNR	dummy	Read Offset	moves the head to its reading position
SeqSegSNR	SeqBaseSegNoiseId	Read	reads the erased signal
SeqSegSNR	dummy	Write Offset	moves the head to its writing position
SeqSegSNR	dummy	Write	writes the HF pattern
SeqSegSNR	dummy	Read Offset	moves the head to its reading position
SeqSegSNR	SeqBaseSegMeasId	Read	reads the HF signal
SeqSegSNR	dummy		closes the registry

Segment 1 integrates and handles the Erase, Read Offset, Read, Write Offset and Write base segment functions as defined by `hpe5022_createSeqSeg (HpE5022, 1, SeqSegSNR)`. Except for SeqBaseSegNoiseId of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegMeasSNR1, SeqBaseSegNoiseId)` and SeqBaseSegMeasId of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegSNR, SeqBaseSegMeasSNR2, SeqBaseSegMeasId)` function which contains the measured data all other base segment IDs can be considered as dummy. When you register a base segment into a segment make sure that its second parameter agrees with the segment handler (i.e. SeqSegSNR). And its third parameter should agree with the base segment handler of each base segment.

Figure 1-15 **Sequence 1**



The figure above shows Sequence 1 containing one segment.The sequence function hpe5022_createSequence (Hpe5022, numOfSec, SeqTest) integrates Segment1 that contains the base segments needed for measurement. When you add a segment to a sequence using hpe5022_addSeqSeg function, make sure that its third parameter agrees with the sequence handler. The function `hpe5022_setupSeq(HpE5022, SeqTest)` and `hpe5022_executeSeq (HpE5022, SeqTest)` setups and executes the UDS program respectively.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sector count and elemCoun. Since the read option base segment is hpe5022_MEAS_TAA, the returned value by elemCoun becomes two. LoopCoun is defined as the number of segment loops. If base segment `hpe5022_createSeqBaseSeg(xxx)` is sectored it will return a value equal to sectCoun, otherwise it returns a value of one. Refer to “User Defined Sequence Functions” of the programming manual for its detailed descriptions.

- hpe5022_result_Q

  This function returns the array data as the measurement result of the specified base segment.The specified base segment **SeqBaseSegNoiseId** and **SeqBaseSegMeasId** should be the same as the “basId” returned by `hpe5022_addSeqBaseSeg (HpE5022, SeqSegSNR, SeqBaseSeg MeasSNR1,`**`SeqBaseSegNoiseId`**`)` and `hpe5022_addSeqBaseSeg (HpE5022, SeqSegSNR, SeqBaseSegMeasSNR2,` **`SeqBaseSegMeasId`**`)` function because it contains the measured data. Note that the unit of measurement returned by Noise( ) and HF( ) are expressed in volts and must be converted to its decibel equivalent [dbm].

Case Study : Resolution

Resolution is specified as the ratio of TAA resulting from two frequencies (HF and LF) written on the same track with the same write current.

Analysis

Write an HF data pattern and measure the TAA.Then, write and measure an LFdata pattern on the same track. Resolution is defined as,

Equation 1-1 Resolution Definition

$$Resolution = \frac{TAA_{HF}}{TAA_{LF}}$$

Data Requirements

Problem Input : Write an HF data pattern.

Write an LF data pattern.

Equation for solving Resolution.

Problem Output : Read the HF and LF data patterns.

Measure the computed results.

Algorithm

1. Input the write data pattern and other parameters.
2. Compute the read data pattern.
3. Display the results of the computed data.

Measuring the resolution can be done in two ways either by using the `hpe5022_measureResolution(HpE5022, NofRev)` function or UDS. UDS offers a more flexible measurement capability than the non-UDS sample program. For it allows the user to pre-define the sequence of measurement as desired.

Implementation

The algorithm above will have to be transformed into a programming language such as Visual Basic in order to implement UDS. To create the UDS follow the steps and procedures listed on “Process for Creating UDS”on page 39. To do so, first create the base segments needed to accomplish a measurement sequence for partial erasure. Then register the created base segments into a segment with its operating parameters. The next step is to register the segment into a sequence for program execution.

Description Resolution can be defined as the ratio of the highest frequency read back signal divided by the lowest frequency read back signal.

Table 1-19 **Base Segment**

Steps To Do	Base Segment	Parameter	Designated Parameter
Step 1	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 2	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 3	Write HF Pattern	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 4	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 5	Read	Data Pattern	hpe5022_PAT_HF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_PARAM
		Read Option	hpe5022_MEAS_TAA
Step 6	Write Offset	Module	hpe5022_MODULE_PIEZO
Step 7	Erase	Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 8	Write LF Pattern	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MODULE_DATA_GEN
Step 9	Read Offset	Module	hpe5022_MODULE_PIEZO
Step 10	Read	Data Pattern	hpe5022_PAT_LF
		Sectoring	VI_FALSE
		Module	hpe5022_MEAS_PARAM
		Read Option	hpe5022_MEAS_TAA

Step 1. Move the write head to its writing position.

Step 2. Erase the track without sectoring.

Step 3. Write the HF data pattern without sectoring. The data signal is supplied by the hpe5022_MODULE_DATA_GEN.

Step 4. Move the read head to its reading position.

Step 5. Read the TAA of the HF data without sectoring. The read data is measured by the parametric module.

Step 6. Move the write head to its writing position.

Step 7. Erase a track without sectoring.

Step 8. Write the LF data pattern without sectoring.

Step 9. Move the read head to its reading position.

Step 10. Read the TAA of the LF data without sectoring. The read data is measured by the parametric module.

The table above describes the pre-defined sequence for resolution measurement. Steps 1 to 10 constitute segment 1. Step 2 erases the track after setting the head to write offset position. Step 3 and 8 specifies the write base segments for HF and LF data patterns. Then it reads the data pattern and measures it by the parametric module as defined by steps 5 and 10. From these two results you can get the ratio. Sectoring is defined as VI_FALSE for all base segments. It can be noticed that steps 1 to 5 practically repeats itself from steps 6 to 10 except for the written data pattern.

Example 1-19 Sample Program(Resolution)

```
Private Sub cmdResolution_Click()

Const numOfSec As Integer = 1

Dim SeqBaseSegWriteOffset, SeqBaseSegErase, SeqBaseSegWriteHF As Long
Dim SeqBaseSegReadOffset, SeqBaseSegMeasHF, SeqBaseSegWriteLF As Long
Dim SeqBaseSegMeasLF, SeqSegResolution, dummy As Long
Dim SeqBaseSegMeasHFId, SeqBaseSegMeasLFId As Long
Dim SeqTest, SeqSegResolutionId As Long
Dim loopCoun, sectCoun, elemCoun As Integer
Dim HFresult(2) As Double
Dim LFresult(2) As Double


'Create Base Segment
ErrorCheck hpe5022_createSeqBaseSegMoveWriteOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_createSeqBaseSegErase(HpE5022, VI_FALSE, hpe5022_MODULE_DATA_GEN, SeqBaseSegErase)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         SeqBaseSegWriteHF)
ErrorCheck hpe5022_createSeqBaseSegMoveReadOffset(HpE5022, hpe5022_MODULE_PIEZO, SeqBaseSegReadOffset)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_HF, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, _
                                        SeqBaseSegMeasHF)
ErrorCheck hpe5022_createSeqBaseSegWrite(HpE5022, hpe5022_PAT_LF, VI_FALSE, hpe5022_MODULE_DATA_GEN, _
                                         SeqBaseSegWriteLF)
ErrorCheck hpe5022_createSeqBaseSegRead(HpE5022, hpe5022_PAT_LF, VI_FALSE, hpe5022_MODULE_PARAM, hpe5022_MEAS_TAA, _
                                        SeqBaseSegMeasLF)

'SEGMENT
ErrorCheck hpe5022_createSeqSeg(HpE5022, 2, SeqSegResolution)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegWriteHF, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegMeasHF, SeqBaseSegMeasHFId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegWriteOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegErase, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegWriteLF, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegReadOffset, dummy)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegMeasLF, SeqBaseSegMeasLFId)
ErrorCheck hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, hpe5022_SEQ_END, dummy)

'SEQUENCE
ErrorCheck hpe5022_createSeq(HpE5022, numOfSec, SeqTest)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, SeqSegResolution, SeqSegResolutionId)
ErrorCheck hpe5022_addSeqSeg(HpE5022, SeqTest, hpe5022_SEQ_END, dummy)
ErrorCheck hpe5022_setupSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_executeSeq(HpE5022, SeqTest)

ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegResolutionId, SeqBaseSegMeasHFId, loopCoun, _
                                 sectCoun, elemCoun)
ErrorCheck hpe5022_resultPoint_Q(HpE5022, SeqTest, SeqSegResolutionId, SeqBaseSegMeasLFId, loopCoun, _
                                 sectCoun, elemCoun)

'HF Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegResolutionId, SeqBaseSegMeasHFId, HFresult(0))
```

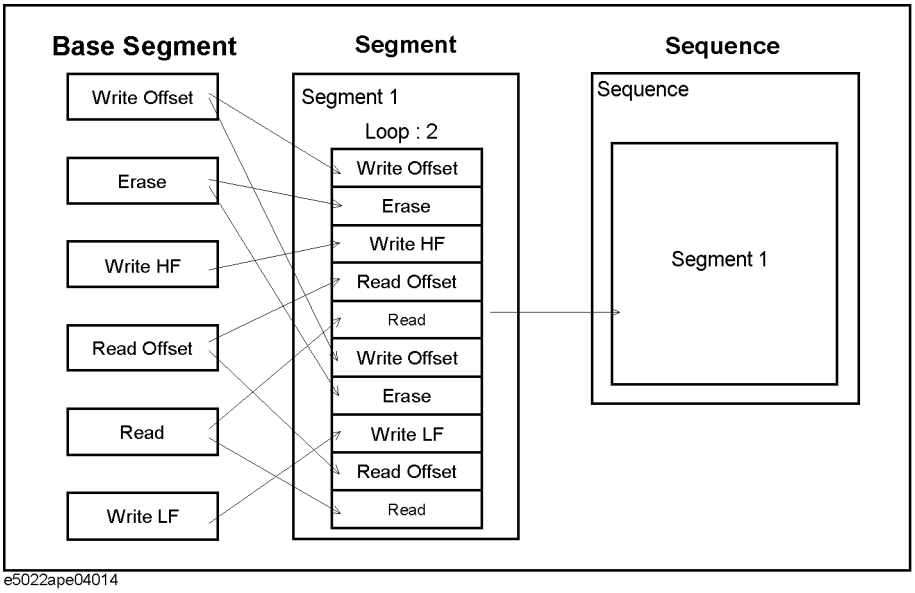
```
'LF  Data
ErrorCheck hpe5022_result_Q(HpE5022, SeqTest, SeqSegResolutionId, SeqBaseSegMeasLFId, LFresult(0))
'Resolution
lblOW.Caption = " Resolution " + Format(((HFresult(0) + HFresult(1)) / _
                                        (LFresult(0) + LFresult(1))) * 100, "0.00E-##") + "%"


ErrorCheck hpe5022_freeSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeq(HpE5022, SeqTest)
ErrorCheck hpe5022_deleteSeqSeg(HpE5022, SeqSegResolution)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegErase)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteHF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasHF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegReadOffset)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegWriteLF)
ErrorCheck hpe5022_deleteSeqBaseSeg(HpE5022, SeqBaseSegMeasLF)

End Sub
```

Table 1-20 **Segment 1 : SeqSegResolution**

Segment 1: (Loop Count =2)			
Segment Handler	Base Segment ID	Base Segment	Description
SeqSegResolution	dummy	Write Offset	moves the head to its writing position
SeqSegResolution	dummy	Erase	erases the track
SeqSegResolution	dummy	Write HF	writes HF data pattern
SeqSegResolution	dummy	Read Offset	moves the head to its reading position
SeqSegResolution	SeqBaseSegMeasHFId	Read	reads the HFdata pattern
SeqSegResolution	dummy	Write Offset	moves the head to its writing position
SeqSegResolution	dummy	Erase	erases the track
SeqSegResolution	dummy	Write LF	writes LF data pattern
SeqSegResolution	dummy	Read Offset	moves the head to its reading position
SeqSegResolution	SeqBaseSegMeasLFId	Read	reads the LF data pattern
SeqSegResolution	dummy		closes the registry

Segment 1 integrates and handles the Erase, Read Offset, Read, Write Offset and Write base segment functions as defined by `hpe5022_createSeqSeg (HpE5022, 2, SeqSegResolution)`. Except for SeqBaseSegMeasHFId of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegMeas, SeqBaseSegMeasHFId)` and SeqBaseSegMeasLFId of `hpe5022_addSeqBaseSeg(HpE5022, SeqSegResolution, SeqBaseSegMeas, SeqBaseSegMeasLFId)` function which contains the measured data all other base segment IDs can be considered as dummy. When you register a base segment into a segment make sure that its second parameter agrees with the segment handler (i.e. , SeqSegResolution).

Figure 1-16 **Sequence 1**



The figure above shows the structure of Sequence 1 containing one segment.The sequence function hpe5022_createSequence (Hpe5022, numOfSec, SeqTest) integrates Segment1 which contains the base segments needed for measurement. When you add a segment to a sequence using hpe5022_addSeqSeg function, make sure that its third parameter agrees with the sequence handle. The function `hpe5022_setupSeq (HpE5022, SeqTest)` and `hpe5022_executeSeq (HpE5022, SeqTest)` sets and executes the UDS program respectively.

Print Query Data :

List the results of the read data.

- hpe5022_resultPoint_Q

  This function returns the size of data. The size of data is equal to the product of loop count, sector count and elemCoun. Since the read option base segment is hpe5022_MEAS_PARAM, the returned value by elemCoun becomes two. LoopCoun is defined as the number of segment loops, in this case it is two. If base segment `hpe5022_createSeqBaseSeg(xxx)` is sectored it will return a value equal to sectCoun, if not it will return a value of one. The fourth parameter of this function must be the same as the read base segment ID.

- hpe5022_result_Q

  This function returns the array data as the measurement result of the specified base segment. The specified base segment ID **SeqBaseSegMeasHFId** and **SeqBaseSegMeasLFId** should be the same as the "basId" returned by `hpe5022_addSeqBaseSeg(HpE5022,SeqSegResolution, SeqBaseSegMeas, SeqBaseSegMeasHFId)` and `hpe5022_addSeqBaseSeg (HpE5022, SeqSegResolution, SeqBaseSegMeas, SeqBaseSegMeasLFId)` function because it contains the measured data. Substituting the results of HFresult( ) and LFresult( ) in the equation formula, we can solve resolution.

2 User-defined Sequence Function Reference

This section describes the functions related with the user-defined sequence.
User-defined sequence for bit error measurement is not supported by the system.

Sequence Functions

hpe5022_createSeq

C Syntax

ViStatus hpe5022_createSeq(ViSession id, Vi Int16 numOfSec, ViPObject seqHndl);

Visual Basic Syntax

hpe5022_createSeq(ByVal id As Long, ByValue numOfSec as Integer, ByRef seqHndl As Long) As Long

Description

This function creates a new sequence that will integrate segments needed to perform a specific measurement.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- numOfSec

  Description — Specifies the number of sectors for base segment.

  The value should be:

  60 / Spindle Speed[rpm] / numOfSec ≥ 150 µsec

  Direction — IN

  Values

Name	Value
hpe5022_SEG_PER_TRACK_MIN	1
hpe5022_SEG_PER_TRACK_MAX	80

- seqHndl

  Description — Specifies the sequence handle of operation.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

See Also

“hpe5022_setupSeq”

“hpe5022_deleteSeq”

“hpe5022_executeSeq”

hpe5022_setupSeq

C Syntax

ViStatus hpe5022_setupSeq(ViSession id, ViObject seqHndl);

Visual Basic Syntax

hpe5022_setupSeq(ByVal id As Long, ByVal seqHndl As Long) As Long

Description

This function sets the sequence data to each module when the sequence becomes executable.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to setup. This is given by the “hpe5022_createSeq” function.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_SEQ_IN_EDIT	The specified sequence is not complete The sequence must end by a “hpe5022_SEQ_END”. See the “hpe5022_addSeqSeg” function.
hpe5022_ERROR_INV_SEQ	The spectrum measurement base segment must be one in a sequence and also it must locate at the last in a sequence.
hpe5022_ERROR_INV_PARAMETER	The number of total base segments exceeds 1024. Reduce the number of base segment in the sequence. If you use the gate related functions such as “hpe5022_seqBaseSegGateConfig”, check the gate parameters.

See Also

“hpe5022_createSeq”

“hpe5022_addSeqSeg”

“hpe5022_seqBaseSegGateConfig”

“hpe5022_seqBaseSegWriteReadGateConfig”

“hpe5022_seqSaGateConfig”

hpe5022_freeSeq

C Syntax

ViStatus hpe5022_freeSeq(ViSession id, ViObject seqHndl);

Visual Basic Syntax

hpe5022_freeSeq(ByVal id As Long, ByVal seqHndl As Long) As Long

Description

This function releases the sequence data from each set up module.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to release. This is given by the “hpe5022_createSeq” function.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.

See Also

“hpe5022_createSeq”

“hpe5022_setupSeq”

hpe5022_deleteSeq

C Syntax

ViStatus hpe5022_deleteSeq(ViSession id, ViObject seqHndl);

Visual Basic Syntax

hpe5022_deleteSeq(ByVal id As Long, ByVal seqHndl As Long) As Long

Description

This function deletes the sequence.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to be removed. This is given by the “hpe5022_createSeq” function.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.

See Also

“hpe5022_createSeq”

hpe5022_executeSeq

C Syntax

ViStatus hpe5022_executeSeq (ViSession id, ViObject seqHndl);

Visual Basic Syntax

hpe5022_executeSeq (ByVal id As Long, ByVal seqHndl As Long) As Long

Description

This function executes the sequence, provided that the sequence is complete and its setup has ended successfully. Before execution, the sequence must be set up by the “hpe5022_setupSeq” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to execute the program. This is given by the “hpe5022_createSeq” function.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_SEQ_IN_EDIT	The specified sequence is not complete The sequence must be end by “hpe5022_SEQ_END”. See the “hpe5022_addSeqSeg” function.
hpe5022_ERROR_SEQ_NOT_SETUP	The specified sequence is not setup. Before performing this function, the sequence should have been set up by the “hpe5022_setupSeq” function

See Also

“hpe5022_createSeq”

“hpe5022_setupSeq”

“hpe5022_addSeqSeg”

hpe5022_addSeqSeg

C Syntax

ViStatus hpe5022_addSeqSeg(ViSession id, ViObject seqHndl, ViObject segHndl, ViPObject segId)

Visual Basic Syntax

hpe5022_addSeqSeg(ByVal id As Long, ByVal seqHndl As Long, ByVal segHndl As Long, ByRef segId As Long) As Long

Description

This function adds a segment into a sequence. To add a segment, the segment must be complete provided that the sequence is not yet complete. If segHndl is “hpe5022_SEQ_END”, this will end the sequence.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle of operation. This is given by the “hpe5022_createSeq” function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to be added to the sequence. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- segId

  Description — Specifies the ID of the segment in the sequence.The data retrieved from the measurement results can be distinguished by the handle ID.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.

Error Code	Description
hpe5022_ERROR_INV_SEG_IN_EDIT	The specified segment is not complete. The segment must end by "hpe5022_SEQ_END" before adding into the sequence. See the "hpe5022_addSeqBaseSeg" function.
hpe5022_ERROR_SEQ_COMPLETE	The "hpe5022_SEQ_END" is already set for the specified sequence. Once "hpe5022_SEQ_END" is set, it is impossible to add a segment into the sequence.

See Also

"hpe5022_createSeq"

"hpe5022_createSeqSeg"

"hpe5022_addSeqBaseSeg"

hpe5022_seqConfiguration

C Syntax

ViStatus hpe5022_seqConfiguration(ViSession id, ViObject seqHndl, ViInt16 config, ViReal64 value);

Visual Basic Syntax

hpe5022_seqConfiguration(ByVal id As Long, ByVal seqHndl As Long, ByVal config As Integer, ByRef value As Double) As Long

Description

This function controls the configuration of the sequence. It specifies the parameters used for oscilloscope and spectrum analyzer measurements.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle of operation. This is given by the “hpe5022_createSeq” function.

  Direction — IN

- config

  Description — Specifies the type of configuration for spectrum analyzer and oscilloscope.

  Direction — IN

  Value

Parameter	Value	Description
hpe5022_CONFIG_SA_CENT_FREQ	0	Center Frequency for narrow band measurement by the spectrum Analyzer
hpe5022_CONFIG_SA_RBW	1	Resolution Band Width for narrow band measurement by the spectrum Analyzer
hpe5022_CONFIG_DSO_BIT_RATE	2	Channel Bit Rate of the Oscilloscope
hpe5022_CONFIG_DSO_REP_LEN	3	The length of one cycle of bit pattern.
hpe5022_CONFIG_SENS_CURR_POL_MODE	4	Specifies the type of sense current mode of the head amplifier.

Parameter	Value	Description
hpe5022_CONFIG_SA_START_FREQ	5	Start Frequency for the spectrum measurement by the spectrum analyzer
hpe5022_CONFIG_SA_STOP_FREQ	6	Stop Frequency for the spectrum measurement by the spectrum analyzer
hpe5022_CONFIG_SA_DET_TYPE	7	Detection Type for the spectrum measurement by the spectrum analyzer.
hpe5022_CONFIG_SA_SPEC_RBW	8	Resolution Band Width Frequency for spectrum measurement by the Spectrum Analyzer.
hpe5022_CONFIG_SA_SPEC_VBW	9	Video Band Width Frequency for spectrum measurement by the Spectrum Analyzer.
hpe5022_CONFIG_WRIT_DATA_POL_MODE	24	Specifies the write current direction mode.
hpe5022_CONFIG_BER_REP_COUN		Specifies the read count of BER measurement.
hpe5022_CONFIG_BER_ERROR_THR		Specifies the maximum byte error rate to stop BER measurement.

- value

  Description Specifies the parameter value. It is dependent on the parameter ‘config’.

  — When “hpe5022_CONFIG_SA_CENT_FREQ” is selected for the parameter ‘config’.

  Specifies the center frequency of the spectrum analyzer.

Parameter	Value
hpe5022_NB_TAA_FREQ_MIN	0

Parameter	Value
hpe5022_NB_TAA_FREQ_MAX	500×10^{6}

— hpe5022_CONFIG_SA_RBW

Specifies the resolution band width of the spectrum analyzer.

Parameter	Value
hpe5022_NB_TAA_BW_MIN	3×10^{3}
hpe5022_NB_TAA_BW_MAX	100×10^{3}

— hpe5022_CONFIG_DSO_BIT_RATE

Specifies the bit rate of the oscilloscope. Generally, this value is set to (channel bit rate).

Parameter	Value
hpe5022_DSO_BIT_RATE_MIN	4×10^{6}

— hpe5022_CONFIG_DSO_REP_LEN

Specifies the length of one cycle of bit pattern.

Generally, this value is set by (data transition period) × 2 for the repetitive pattern. (data length) for the user-defined pattern and pseudo random data pattern.

Parameter	Value
hpe5022_DSO_REP_LEN_MIN	2

— hpe5022_CONFIG_SENS_CURR_POL_MODE

This is the same as the stability configuration. See the “hpe5022_stabilityConfig” function.

Specifies the sense current direction mode.

Parameter	Val.	Description
hpe5022_SENS_CURR_POL_NORM	0	Same polarity
hpe5022_SENS_CURR_POL_REV	1	Reverse polarity

Parameter	Val.	Description
hpe5022_SENS_CURR_POL_OFF	2	Off Mode

— hpe5022_CONFIG_SA_START_FREQ

Specifies the start frequency for the spectrum measurement by the spectrum analyzer.

Parameter	Value
hpe5022_SPEC_FREQ_MIN	0
hpe5022_SPEC_FREQ_MAX	500×10^6

— hpe5022_CONFIG_SA_STOP_FREQ

Specifies the stop frequency for the spectrum measurement by the spectrum analyzer.

Parameter	Value
hpe5022_SPEC_FREQ_MIN	0
hpe5022_SPEC_FREQ_MAX	500×10^6

— hpe5022_CONFIG_SA_DET_TYPE

This is the same as the detection configuration for spectrum measurement. See the “hpe5022_measureSpectralSnr” function.

Specifies the detection mode as shown below.

Name	Val.	Description
hpe5022_SPEC_DET_SAMP	1	Sample Mode
hpe5022_SPEC_DET_POS_PEAK	2	Positive Peak Mode

— hpe5022_CONFIG_SA_SPEC_RBW

Specifies the resolution band width frequency for the spectrum measurement by the spectrum analyzer. See “hpe5022_spectrumBandWidth” function for setting

values.

Parameter	Value
hpe5022_SPEC_BW_MIN	3×10^{3}
hpe5022_SPEC_BW_MAX	1×10^{6}

— hpe5022_CONFIG_SA_SPEC_VBW

Specifies the video band width frequency for the spectrum measurement by the spectrum analyzer. See "hpe5022_spectrumBandWidth" function for setting values.

Parameter	Value
hpe5022_SPEC_VBW_MIN	10
hpe5022_SPEC_VBW_MAX	1×10^{6}

— hpe5022_CONFIG_WRIT_DATA_POL_MODE

Specifies the direction of write current.

Name	Value	Description
hpe5022_WRIT_DATA_POL_FIXED	0	The write current direction at the beginning of each write operation is always the same.
hpe5022_WRIT_DATA_POL_CONTINUOUS	1	The write current direction at the beginning of write operation is the same as the last current direction of the previous write operation.

— hpe5022_CONFIG_BER_REP_COUN

Specifies the read count of BER measurement.

Parameter	Value
hpe5022_BER_REP_COUN_MIN	1
hpe5022_BER_REP_COUN_MAX	1×10^{6}

— hpe5022_CONFIG_BER_ERROR_THR

Specifies the maximum byte error rate to stop the BER measurement. This function is only available for the E5039C bit error test module. If the E5039A/B is used as

a bit error test module, this function is not supported.

Parameter	Value
hpe5022_BER_ERROR_THR_MIN	0
hpe5022_BER_ERROR_THR_MAX	1

Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by 'seqHndl' is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter 'config' is out of range.
hpe5022_ERROR_NSUP_SENS_CURR_POL	Specified polarity mode is not supported for the installed head amplifier.

See Also

"hpe5022_createSeq"

"hpe5022_stabilityConfig"

"hpe5022_spectrumBandWidth"

hpe5022_seqConfiguration_Q

C Syntax

ViStatus hpe5022_seqConfiguration_Q(ViSession id, ViObject seqHndl, ViInt16 config, ViPReal64 value);

Visual Basic Syntax

hpe5022_seqConfiguration_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal config As Integer, ByRef value As Double) As Long

Description

This function returns the specified configuration of the sequence.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle of operation. This is given by the “hpe5022_createSeq” function.

  Direction — IN

- config

  Description — Specifies the type of sequence configuration.

  Direction — IN

  Value — Same as “hpe5022_seqConfiguration” function.

- value

  Description — Returns the configuration value.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.

See Also

“hpe5022_createSeq”

“hpe5022_seqConfiguration”

hpe5022_seqSaGateConfig

C Syntax

ViStatus hpe5022_seqSaGateConfig(ViSession id, ViObject seqHndl, ViInt16 gateMode, ViReal64 trigDel, ViReal64 aper);

Visual Basic Syntax

hpe5022_seqSaGateConfig(ByVal id As Long, ByVal seqHndl As Long, ByVal gateMode As Integer, ByVal trigDel As Double, ByVal aper As Double) As Long

Description

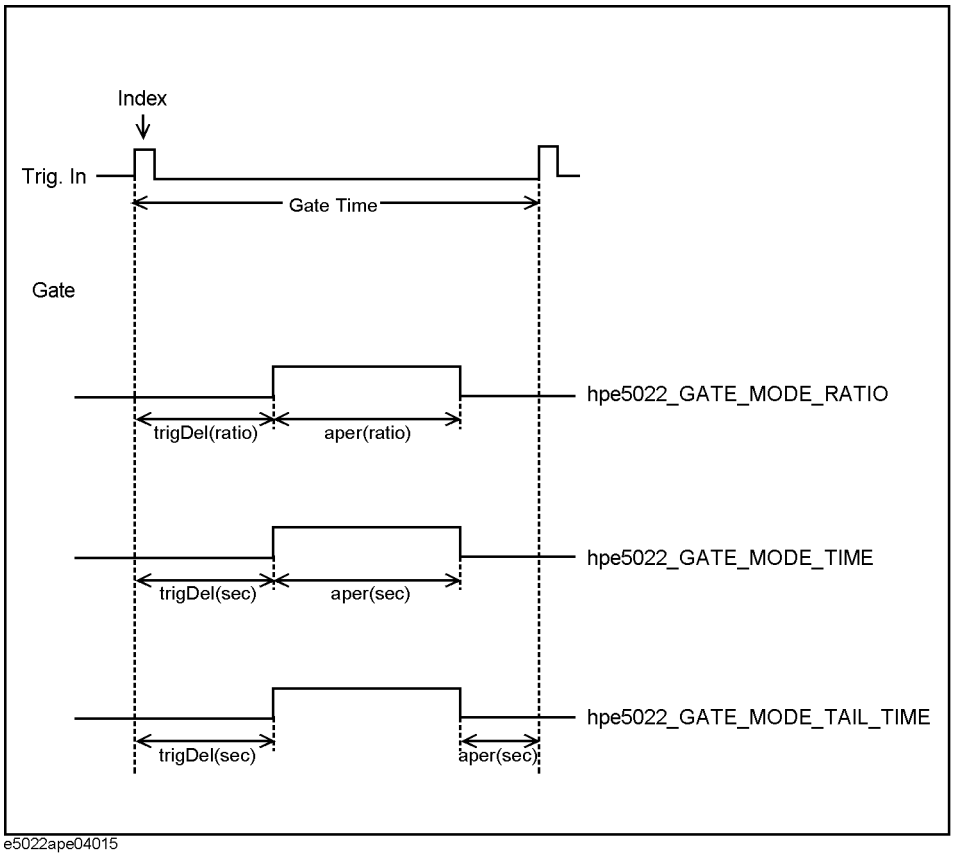
This function specifies the gate for a sequence which has a spectrum measurement. This function allows you to read a signal by a spectrum measurement only during the specified period on a track.

When you want to specify a gate to write data or measure by a parametric module, see the “hpe5022_seqBaseSegGateConfig” or “hpe5022_seqBaseSegWriteReadGateConfig” function.

The sequence defined by this function must not be sectored.

The definition of trigger delay and aperture is shown in Figure 1-1.

Figure 2-1 Trigger Delay and Aperture



e5022ape04015

The range of gate is shown below. This range is a limitation from the software viewpoint. This might be narrowed depending on its hardware, such as a head amplifier. When the “hpe5022_setupSeq” or “hpe5022_executeSeq” function is

executed, the specified value is checked, then “hpe5022_ERROR_INV_PARAMETER” is displayed if the value is over the range.

Delay	≥ 1.0 µsec.
Aperture	≥ 0.8 msec.
	≥ 1.40 msec. @ hpe5022_MEAS_SPEC and resolution band width is set to 3.0kHz
Delay + Aperture	≤ (60 / Spindlespeed [rpm]) - 100 µsec.

Parameters

- id

  Description Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction IN

- seqHndl

  Description Specifies the sequence segment handle. This is given by the “hpe5022_createSeq” function.

  Direction IN

- gateMode

  Description Specifies the gate mode.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_GATE_MODE_AUTO	0	Gate (delay, aperture) is automatically set. In this mode, `trigDel' and `aper' are ignored.
hpe5022_GATE_MODE_RATIO	1	Gate is specified as ratio.
hpe5022_GATE_MODE_TIME	2	Gate is specified as time. Unit is second.
hpe5022_GATE_MODE_TAIL_TIME	3	Trigger delay is specified as time in sec. Aperture is specified as the time between gate end and next trigger.

- trigDel

  Description Specifies the delay time from trigger. Range value is

provided below.

gateMode	Value
hpe5022_GATE_MODE_AUTO	N/A
hpe5022_GATE_MODE_RATIO	0 ≤ trigDel < 1.0
hpe5022_GATE_MODE_TIME	0 ≤ trigDel
hpe5022_GATE_MODE_TAIL_TIME	0 ≤ trigDel

Direction IN

- aper

  Description Specifies the aperture time of the gate. Range value is shown below.

gateMode	Value
hpe5022_GATE_MODE_AUTO	N/A
hpe5022_GATE_MODE_RATIO	0.0 < trigDel + aper < 1.0
hpe5022_GATE_MODE_TIME	0 ≤ aper
hpe5022_GATE_MODE_TAIL_TIME	N/A

  Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter is out of range.

See Also

“hpe5022_createSeq”

“hpe5022_seqSaGateConfig_Q”

“hpe5022_seqBaseSegGateConfig”

“hpe5022_seqBaseSegWriteReadGateConfig”

hpe5022_seqSaGateConfig_Q

C Syntax

ViStatus hpe5022_seqSaGateConfig_Q(ViSession id, ViObject seqHndl, ViPInt16 gateMode, ViPReal64 trigDel, ViPReal64 aper);

Visual Basic Syntax

hpe5022_seqSaGateConfig_Q(ByVal id As Long, ByVal seqHndl As Long, ByRef gateMode As Integer, ByRef trigDel As Double, ByRef aper As Double) As Long

Description

This function returns the gate setting for a sequence which is specified by the “hpe5022_seqBaseSegGateConfig” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence segment handle. This is given by the “hpe5022_createSeq” function.

  Direction — IN

- gateMode

  Description — Returns the specified gate mode.

  Direction — OUT

  Value — Same as the ‘gateMode’ parameter in the “hpe5022_seqSaGateConfig” function.

- trigDel

  Description — Returns the trigger delay time.

  Direction — OUT

- aper

  Description — Returns the aperture time of the gate.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.

See Also “hpe5022_seqSaGateConfig”

Segment Functions

hpe5022_createSeqSeg

C Syntax

ViStatus hpe5022_createSeqSeg(ViSession id, ViInt16 loopCoun, ViPObject segHndl);

Visual Basic Syntax

hpe5022_createSeqSeg(ByVal id As Long, ByVal loopCoun As Integer, ByRef segHndl As Long) As Long

Description

This function creates the segment that integrates base segments needed to perform a specific measurement.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- loopCoun

  Description — Specifies the number segment loops.

  Direction — IN

- segHndl

  Description — Specifies the segment handle of operation.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

See Also

“hpe5022_addSeqBaseSeg”

hpe5022_deleteSeqSeg

C Syntax

ViStatus hpe5022_deleteSeqSeg(ViSession id, ViObject segHndl);

Visual Basic Syntax

hpe5022_deleteSeqSeg(ByVal id As Long, ByVal segHndl As Long) As Long

Description

This function deletes the segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to be deleted. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.

See Also

“hpe5022_createSeqSeg”

hpe5022_addSeqBaseSeg

C Syntax

ViStatus hpe5022_addSeqBaseSeg(ViSession id, ViObject segHndl, ViObject basHndl, ViPObject basId);

Visual Basic Syntax

hpe5022_addSeqBaseSeg(ByVal id As Long, ByVal segHndl As Long, ByVal basHndl As Long, ByRef basId As Long) As Long

Description

This function adds a base segment to a segment. To add a base segment, the segment must be incomplete (i.e. hpe5022_addSeqSeg). If basHndl is “hpe5022_SEQ_END”, this means the end of segment and it has been completed.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle of operation. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to be added to the segment. This is given by either the “hpe5022_createSeqBaseSegMove”, the “hpe5022_createSeqBaseSegMoveWriteOffset”, the “hpe5022_createSeqBaseSegMoveReadOffset”, the “hpe5022_createSeqBaseSegErase”, the “hpe5022_createSeqBaseSegWrite”, the “hpe5022_createSeqBaseSegRead” or the “hpe5022_createSeqBaseSegWriteRead” functions.

  Direction — IN

- basId

  Description — Specifies the ID of base segment in the segment.The data retrieved from measurement results can be distinguished by the identifier ID.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_BAS_HNDL	The handle specified by ‘basHndl’ is invalid.
hpe5022_ERROR_SEG_COMPLETE	The “hpe5022_SEQ_END” is already set for the specified segment. Once “hpe5022_SEQ_END” is set, it is impossible to add a base segment into the segment.

See Also

“hpe5022_addSeqSeg”

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_seqSegParameter

C Syntax

ViStatus hpe5022_seqSegParameter(ViSession id, ViObject segHndl, ViInit16 param, ViReal64 value);

Visual Basic Syntax

hpe5022_seqSegParameter(ByVal id As Long, ByVal segHndl As Long, ByVal param As Integer, ByVal value As Double) As Long

Description

This function assigns a constant parameter value to a segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to set the parameter value. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- param

  Description — Specifies the type of parameter to be selected.

  Direction — IN

  Value — The parameter to be selected is dependent on the base segment as shown below.

— hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1

Parameter	Value	Description
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- value

  Description Specifies the constant parameter value.

  Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘param’ is out of range.

See Also

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_seqSegParameterSweep

C Syntax

ViStatus hpe5022_seqSegParameterSweep(ViSession id, ViObject segHndl, ViInit16 param, ViReal64 startVal,ViReal64 stopVal);

Visual Basic Syntax

hpe5022_seqSegParameterSweep(ByVal id As Long, ByVal segHndl As Long, By Val param As Integer, ByVal startVal As Double, ByVal stopVal As Double) As Long

Description

This function sets a sweep parameter value to a segment. The number of sweep points must be equal to the number of loops of the segment. You should set to “VI_FALSE” on the parameter ‘fixed’ in the “hpe5022_seqBaseSegParameter” function for the corresponding paramter.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to set the parameter. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- param

  Description — Specifies the type of parameter sweep.

  Direction — IN

  Value — Same as the ‘param’ parameter in the “hpe5022_seqSegParameter” function.

  — hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- startVal

  Description Specifies the starting parameter value of the sweep.

  Direction IN

- stopVal

  Description Specifies the stopping parameter value of the sweep.

Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘param’ is out of range.

See Also

“hpe5022_seqBaseSegParameter”

“hpe5022_seqSegParameter”

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_seqSegParameterList

C Syntax

ViStatus hpe5022_seqSegParameterList(ViSession id, ViObject segHndl, ViInit16 param, ViInt16 poin,ViReal64 valList[ ]);

Visual Basic Syntax

hpe5022_seqSegParameterList(ByVal id As Long, ByVal segHndl As Long, ByVal param As Integer, ByVal poin As Integer, ByRef valList As Double) As Long

Description

This function sets the parameter value list to a segment. Sweep points is equal to the number of loop counts in a segment. If poin is less than the number of sweep points, then the last value of valList will be used as an extension. You should set to “VI_FALSE” on the parameter ‘fixed’ in the “hpe5022_seqBaseSegParameter” function for the corresponding paramter.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to set the parameter value. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- param

  Description — Specifies the type of parameter sweep.

  Direction — IN

  Value — Same as the ‘param’ parameter in the “hpe5022_seqSegParameter” function.

  — hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- poin

  Description Specifies the length of valList.

  Direction IN

- ValList

  Description Specifies the parameter value list.

  Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘param’ is out of range.

See Also

“hpe5022_seqBaseSegParameter”

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_seqSegParameterPoint_Q

C Syntax

ViStatus hpe5022_seqSegParameterPoint_Q(ViSession id, ViObject segHndl, ViPInt16 poin);

Visual Basic Syntax

hpe5022_seqSegParameterPoint_Q(ByVal id As Long, ByVal segHndl As Long, ByRef poin As Integer) As Long

Description

This function returns the size of the parameter value in array form from the number of loops of a segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- poin

  Description — Returns the length of the segment parameter value.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.

See Also

“hpe5022_createSeqSeg”

hpe5022_seqSegParameter_Q

C Syntax

ViStatus hpe5022_seqSegParameter_Q(ViSession id, ViObject segHndl, ViInt16 param, ViReal64 valList[ ]);

Visual Basic Syntax

hpe5022_seqSegParameter_Q(ByVal id As Long, ByVal segHndl As Long, ByVal param As Integer, ByRef valList As Double) As Long

Description

This function lists the extracted data of the segment parameter value.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle of operation. This is given by the “hpe5022_createSeqSeg” function.

  Direction — IN

- param

  Description — Specifies the type of parameter to get the results.

  Direction — IN

  Value — The parameter to be selected is dependent on the base segment as shown below. Same as the ‘param’ in the “hpe5022_seqSegParameter”

— hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1

Parameter	Value	Description
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- valList

  Description — Returns the segment parameter value that are set in array. The “hpe5022_seqSegParameterPoint_Q” function returns the size of the array.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.

See Also

“hpe5022_createSeqSeg”

“hpe5022_seqSegParameterPoint_Q”

Base Segment Functions

hpe5022_createSeqBaseSegMove

C Syntax

ViStatus hpe5022_createSeqBaseSegMove (ViSession id, ViInt16 module, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegMove(ByVal id As Long, ByVal module As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment for head movement. This base segment moves the head. The distance of movement is defined by the “hpe5022_seqBaseSegParameter” function. The distance is defined as absolute offset from the center of track.

Parameters

- id

  Description Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction IN

- module

  Description Assigns the module. This value must be selected for this function.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_MODULE_PIEZO	0	Piezo on the spinstand

- basHndl

  Description Returns the base segment handle.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘module’ is out of range.

See Also “hpe5022_seqBaseSegParameter”

hpe5022_createSeqBaseSegMoveWriteOffset

C Syntax

ViStatus hpe5022_createSeqBaseSegMoveWriteOffset(ViSession id, ViInt16 module, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegMoveWriteOffset(ByVal id As Long, ByVal module As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment of the write track offset head movement. (The write track offset value is defined by the “hpe5022_writeTrackOffset” function).

Parameters

- id

  Description Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction IN

- module

  Description Assigns the module. This value must be selected for this function.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_MODULE_PIEZO	0	Piezo on the spinstand

- basHndl

  Description Returns the base segment handle.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘module’ is out of range.

See Also

“hpe5022_writeTrackOffset”

hpe5022_createSeqBaseSegMoveReadOffset

C Syntax

ViStatus hpe5022_createSeqBaseSegMoveReadOffset(ViSession id, ViInt16 module, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegMoveReadOffset(ByVal id As Long, ByVal module As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment of the read offset head movement. (The read track offset value is defined by the “hpe5022_readTrackOffset”).

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- module

  Description — Assigns the module. This following value must be selected for this function.

  Direction — IN

  Value

Parameter	Value	Description
hpe5022_MODULE_PIEZO	0	Piezo on the spinstand

- basHndl

  Description — Returns the base segment handle.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘module’ is out of range.

See Also

“hpe5022_readTrackOffset”

hpe5022_createSeqBaseSegErase

C Syntax

ViStatus hpe5022_createSeqBaseSegErase (ViSession id, ViBoolean sectored, ViInt16 module, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegErase(ByVal id As Long, ByVal sectored As Integer, ByVal module As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment to erase.

Parameters

- id

  Description: Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction: IN

- sectored

  Description: Specifies the number of track divisions. The number of sector is defined by the “hpe5022_createSeq”

  Generally, this is set to “VI_FALSE”.

  Direction: IN

  Value

Parameter	Value	Description
VI_TRUE	1	Sectored
VI_FALSE	0	No sector

- module

  Description: Assigns the module. The following value must be selected for this function.

  Direction: IN

  Value

Parameter	Value	Description
hpe5022_MODULE_DATA_GEN	2	Data Generator Module

- basHndl

  Description: Returns the base segment handle.

  Direction: OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘sectored’ and/or ‘module’ is out of range.

See Also

“hpe5022_createSeq”

hpe5022_createSeqBaseSegWrite

C Syntax

ViStatus hpe5022_createSeqBaseSegWrite (ViSession id, ViInt16 datPat, ViBoolean sectored, ViInt16 module, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegWrite(ByVal id As Long, ByVal datPat As Integer, ByVal sectored As Integer, ByVal module As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment to write a data pattern

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- datPat

  Description — Specifies the data pattern to be written on the media.

  Direction — IN

  Value — If ‘module’ is hpe5022_MODULE_DATA_GEN or hpe5022_MODULE_EXTERNAL, the following patterns are available:

Name	Value	Description
hpe5022_PAT_DEFAULT	-1	Use the pattern specified by the “hpe5022_selectPattern” function.
hpe5022_PAT_HF	0	HF pattern
hpe5022_PAT_LF	1	LF pattern
hpe5022_PAT_ISO	2	Isolated pulse pattern
hpe5022_PAT_PRBS	3	Pseudo random pattern
hpe5022_PAT_REP	4	Repetitive pattern
hpe5022_PAT_NLTS_5TH	5	NLTS 5th pattern
hpe5022_PAT_OWHF	6	Overwrite HF pattern
hpe5022_PAT_OWLF	7	Overwrite LF pattern
hpe5022_PAT_REP_2	8	Repetitive pattern 2
hpe5022_PAT_REP_3	9	Repetitive pattern 3
hpe5022_PAT_REP_4	10	Repetitive pattern 4
hpe5022_PAT_USER	20	User-defined Pattern

Name	Value	Description
hpe5022_PAT_USER_2	21	User-defined Pattern 2
hpe5022_PAT_USER_3	22	User-defined Pattern 3
hpe5022_PAT_USER_4	23	User-defined Pattern 4
hpe5022_PAT_ERASE	101	Use the erase pattern specified by the “hpe5022_eraseType” function.
hpe5022_PAT_ERASE_DC_NEG	102	DC- Erase Pattern
hpe5022_PAT_ERASE_DC_POS	103	DC+ Erase Pattern
hpe5022_PAT_ERASE_AC	104	AC Erase Pattern

If ‘module’ is hpe5022_MODULE_BER, the following patterns are available:

Name	Value	Description
hpe5022_PAT_DEFAULT	-1	Use the pattern specified by the “hpe5022_selectPattern” function.
hpe5022_BER_PAT_1	31	BER pattern 1
hpe5022_BER_PAT_2	32	BER pattern 2
hpe5022_BER_PAT_3	33	BER pattern 3
hpe5022_BER_PAT_4	34	BER pattern 4
hpe5022_BER_PAT_5	35	BER pattern 5
hpe5022_BER_PAT_6	36	BER pattern 6

If the E5039C is used as a bit error test module, hpe5022_BER_PAT_5 and hpe5022_BER_PAT_6 are not available.

- sectored

  Description — Specifies the number of track divisions. The number of sector is defined by “hpe5022_createSeq” function.

  Generally, this is set to VI_FALSE.

  Direction — IN

  Value

Parameter	Value	Description
VI_TRUE	1	Sectored

Parameter	Value	Description
VI_FALSE	0	No sector

- module

  Description Assigns the module. This value must be selected for this function.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_MODULE_DATA_GEN	2	Data Generator Module

- basHndl

  Description Returns the base segment handle.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘datPat’, ‘sectored’ and/or ‘module’ is out of range.

See Also

“hpe5022_selectPattern”

“hpe5022_createSeq”

hpe5022_createSeqBaseSegRead

C Syntax

ViStatus hpe5022_createSeqBaseSegRead (ViSession id, ViInt16 datPat, ViBoolean sectored, ViInt16 module, ViInt16 readOption, ViPObject basHndl);

Visual Basic Syntax

hpe5022_createSeqBaseSegRead(ByVal id As Long, ByVal datPat As Integer, ByVal sectored As Integer, ByVal module As Integer, ByVal readOption As Integer, ByRef basHndl As Long) As Long

Description

This function creates a base segment to measure.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- datPat

  Description — Specifies the data pattern to be read from the media.

  Direction — IN

  Value — Same as the ‘datPat’ parameter in the “hpe5022_createSeqBaseSegWrite” function.

- sectored

  Description — Specifies the number of track divisions. The number of sector is defined by the “hpe5022_createSeq” function.

  Generally, this is set to VI_FALSE. When the hpe5022_MODULE_DSO or hpe5022_MODULE_SA is selected as module, this must be VI_FALSE.

  Direction — IN

  Value

Parameter	Value	Description
VI_TRUE	1	Sectored
VI_FALSE	0	No sector

- module

  Description — Assigns the module. The parameter to be selected depends on the parameter ‘readOption’. The following value must be selected for this function.

  Direction — IN

  Value — The parameter to be selected depends on the parameter ‘readOption’.

— When ‘readOption’ is set to either “hpe5022_MEAS_TAA”, “hpe5022_MEAS_PW” or “hpe5022_MEAS_BASE”.

Parameter	Value	Description
hpe5022_MODULE_PARAM	3	Parametric Module

— When ‘readOption’ is set to“hpe5022_MEAS_WAVE”

Parameter	Value	Description
hpe5022_MODULE_DSO	4	Oscilloscope

— When ‘readOption’ is set to“hpe5022_MEAS_LEVEL” or “hpe5022_MEAS_SPEC”.

Parameter	Value	Description
hpe5022_MODULE_SA	5	Spectrum Analyzer

- readOption

  Description Specifies the type of read option when reading the data.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_MEAS_TAA	1	TAA of Parametric Module
hpe5022_MEAS_PW	2	PW of Parametric Module
hpe5022_MEAS_BASE	6	Baseline of Parametric Module
hpe5022_MEAS_LEVEL	7	Level of Spectrum Analyzer
hpe5022_MEAS_WAVE	8	Waveform of Oscilloscope
hpe5022_MEAS_SPEC	9	Spectrum data of Spectrum Analyzer

- basHndl

  Description Returns the base segment handle.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘datPat’, ‘sectored’, ‘module’ and/or ‘read option’ is out of range.

See Also

“hpe5022_selectPattern”

“hpe5022_createSeq”

hpe5022_createSeqBaseSegWriteRead

C Syntax

ViStatus hpe5022_createSeqBaseSegWriteRead (ViSession id, ViBoolean sectored, ViInt16 writPat, ViInt16 writeModule, ViReal64 writeRatio, ViInt16 readPat, ViInt16 readModule, ViInt16 readOption, ViReal64 readRatio, ViPObject basHndl);

Visual Basic Syntax

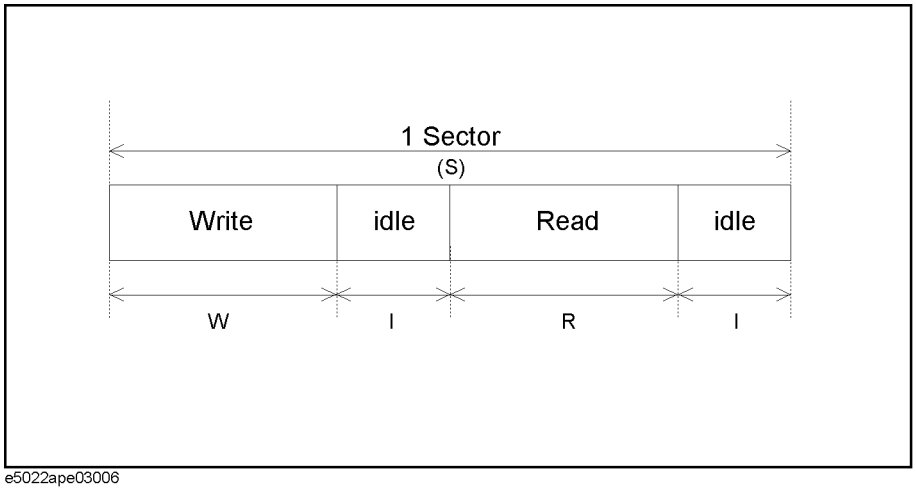
hpe5022_createSeqBaseSegWriteRead(ByVal id As Long, ByVal sectored As Integer, ByVal writdatPat As Integer, ByVal writeModule As Integer, ByVal writeRatio As Double, ByVal readdatPat As Integer, ByVal readModule As Integer, ByVal readOption As Integer, ByVal readRatio As Double, ByRef basHndl As Long) As Long

Description

This function creates a base segment for write/read measurement. Generally this is used for stability measurement.

The write ratio is defined as the aperture ratio of the write period in one segment and the read ratio is defined as one of the read period in one segment. The portion besides the read and the write sections are defined as idle portion as shown in Figure 1-2. The idle portion is located between the write and read portion. Both idle portions are of the same length.

Figure 2-2 **Read and Write Ratios**



e5022ape03006

The value of ‘writeRatio’ + ‘readRatio’ must be less than the value of (hpe5022_TAA_STABILITY_RATIO_MAX + hpe5022_TAA_STABILITY_RATIO_MIN)

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- sectored

  Description — Specifies the sectoring of the track. The number of sector is defined by the “hpe5022_createSeq” function. When the hpe5022_MODULE_DSO or hpe5022_MODULE_SA is selected as module, this must be VI_FALSE.

  Direction — IN

  Value

Parameter	Value	Description
VI_TRUE	1	Sectored
VI_FALSE	0	No sector

- writPat

  Description — Specifies the data pattern to be written on the media.

  Direction — IN

  Value — Same as the ‘datPat’ parameter in the “hpe5022_createSeqBaseSegWrite” function.

- writeModule

  Description — Assigns the module. The value specified below must be selected for this function.

  Direction — IN

  Value

Parameter	Value	Description
hpe5022_MODULE_DATA_GEN	2	Data Generator Module

- writeRatio

  Description — Specifies the write aperture ratio.

  Direction — IN

  Preset value — 0.4

  Values

Name	Value
hpe5022_STABILITY_RATIO_MIN	0.01
hpe5022_STABILITY_RATIO_MAX	0.98

- readPat

  Description — Specifies the data pattern to be read from the media.

  Direction — IN

  Value — Same as the ‘datPat’ parameter in the “hpe5022_createSeqBaseSegWrite” function.

- readModule

  Description — Assigns the module for read. The parameter to be selected depends on the parameter ‘readOption’. The following value must be selected for this function.

  Direction — IN

  Value — The parameter to be selected depends on the parameter ‘readOption’.

  — When ‘readOption’ is set toeither “hpe5022_MEAS_TAA”, “hpe5022_MEAS_PW” or “hpe5022_MEAS_BASE”.

Parameter	Value	Description
hpe5022_MODULE_PARAM	3	Parametric Module

  — When ‘readOption’ is set to “hpe5022_MEAS_WAVE”

Parameter	Value	Description
hpe5022_MODULE_DSO	4	Oscilloscope

  — When ‘readOption’ is set to “hpe5022_MEAS_LEVEL”

Parameter	Value	Description
hpe5022_MODULE_SA	5	Spectrum Analyzer

- readOption

  Description — Specifies the type of read option when reading the data.

  Direction — IN

  Value

Parameter	Value	Description
hpe5022_MEAS_TAA	1	TAA
hpe5022_MEAS_PW	2	PW

Parameter	Value	Description
hpe5022_MEAS_BASE	6	Baseline
hpe5022_MEAS_LEVEL	7	Level
hpe5022_MEAS_WAVE	8	Waveform

- readRatio

  Description Specifies the read aperture ratio.

  Direction IN

  Preset value 0.4

  Values

Name	Value
hpe5022_STABILITY_RATIO_MIN	0.01
hpe5022_STABILITY_RATIO_MAX	0.98

- basHndl

  Description Returns the base segment handle.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_PARAMETER	The parameter ‘datPat’, ‘sectored’, ‘module’ and/or ‘readoption’ is out of range.

See Also

“hpe5022_selectPattern”

“hpe5022_createSeq”

hpe5022_deleteSeqBaseSeg

C Syntax

ViStatus hpe5022_deleteSeqBaseSeg(ViSession id, ViObject basHndl);

Visual Basic Syntax

hpe5022_deleteSeqBaseSeg(ByVal id As Long, ByVal basHndl As Long) As Long

Description

This function deletes a base segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to delete. This is given by either the “hpe5022_createSeqBaseSegMove”, the “hpe5022_createSeqBaseSegMoveWriteOffset”, the “hpe5022_createSeqBaseSegMoveReadOffset”, the “hpe5022_createSeqBaseSegErase”, the “hpe5022_createSeqBaseSegWrite”, the “hpe5022_createSeqBaseSegRead” or the “hpe5022_createSeqBaseSegWriteRead” functions.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.

See Also

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_seqBaseSegParameter

C Syntax

ViStatus hpe5022_seqBaseSegParameter (ViSession id, ViObject basHndl, ViInt16 param, ViReal64 value, ViBoolean fixed);

Visual Basic Syntax

hpe5022_seqBaseSegParameter(ByVal id As Long, ByVal basHndl As Long, ByVal param As Integer, ByVal value As Double, ByVal fixed As Integer) As Long

Description

This function specifies a value to a base segment. If the parameter ‘fixed’ is set to “VI_TRUE”, then the value is fixed at the parameter ‘value’ for the specified base segment. If the parameter ‘fixed’ is set to “VI_FALSE”, then the ‘value’ is changeable when a segment is looped. The “hpe5022_seqSegParameterSweep” and “hpe5022_seqSegParameterSweep” function can change the value of parameter.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to set the parameter value. This is given by either the “hpe5022_createSeqBaseSegMove”, the “hpe5022_createSeqBaseSegMoveWriteOffset”, the “hpe5022_createSeqBaseSegMoveReadOffset”, the “hpe5022_createSeqBaseSegErase”, the “hpe5022_createSeqBaseSegWrite”, the “hpe5022_createSeqBaseSegRead” or the “hpe5022_createSeqBaseSegWriteRead” functions.

  Direction — IN

- param

  Description — Specifies the type of parameter to be selected.

  Direction — IN

  Value — The parameter to be selected is dependent on the base segment as shown below. Same as the ‘param’ parameter in the “ hpe5022_seqSegParameter” function.

  — hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- value

  Description — Specifies the constant parameter value. If the parameter 'fixed' is set to "VI_FALSE", this should be set to 0, otherwise the value will be the sum of this value and the value specified by the "hpe5022_seqSegParameterSweep" and "hpe5022_seqSegParameterList" function.

  Direction — IN

- fixed

  Description — Specifies the type of parameter.

  Direction — IN

  Value — When this parameter is set to "VI_TRUE", the specified 'value' is not changed. In that case the "hpe5022_seqSegParameter", the "hpe5022_seqSegParameterSweep" and the "hpe5022_seqSegParameterList" functions do not affect the parameter value.

  Value

Parameter	Value	Description
VI_TRUE	1	The parameter is fixed
VI_FALSE	0	The parameter is variable.

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.
hpe5022_ERROR_INV_PARAMETER	The parameter is invalid.

See Also

"hpe5022_createSeqBaseSegMove"

"hpe5022_createSeqBaseSegMoveWriteOffset"

"hpe5022_createSeqBaseSegMoveReadOffset"

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

“hpe5022_seqSegParameter”

“hpe5022_seqSegParameterSweep”

“hpe5022_seqSegParameterList”

hpe5022_seqBaseSegParameterPoint_Q

C Syntax

ViStatus hpe5022_seqBaseSegParameterPoint_Q (ViSession id, ViObject basHndl, ViInt16 sectCoun, ViPInt16 poin);

Visual Basic Syntax

hpe5022_seqBaseSegParameterPoint_Q(ByVal id As Long, ByVal basHndl As Long, ByVal sectCoun As Integer, ByRef poin As Integer) As Long

Description

This function returns the number of elements returned by “hpe5022_seqBaseSegParameter_Q” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle. The handle should be the same as one returned by either the “hpe5022_createSeqBaseSegMove”, the “hpe5022_createSeqBaseSegMoveWriteOffset”, the “hpe5022_createSeqBaseSegMoveReadOffset”, the “hpe5022_createSeqBaseSegErase”, the “hpe5022_createSeqBaseSegWrite”, the “hpe5022_createSeqBaseSegRead” or the “hpe5022_createSeqBaseSegWriteRead” functions.

  Direction — IN

- sectCoun

  Description — Specifies the sector number.

  Direction — IN

- poin

  Description — Returns the number of points by the “hpe5022_seqBaseSegParameter_Q”. If base segment is not sectored, it returns a value of one. If sectored, it returns a value equal sectCoun.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.

See Also

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegWriteRead”

“hpe5022_createSeqBaseSegRead”

hpe5022_seqBaseSegParameter_Q

C Syntax

ViStatus hpe5022_seqBaseSegParameter_Q (ViSession id, ViObject basHndl, ViInt16 sectCoun, ViInt16 param,ViPReal64 valList[]);

Visual Basic Syntax

hpe5022_seqBaseSegParameter_Q(ByVal id As Long, ByVal basHndl As Long, ByVal sectCoun As Integer, ByVal param as Integer, ByRef valList As Double) As Long

Description

This function returns the query list of base segment parameter value. The “hpe5022_seqBaseSegParameterPoint_Q” function returns the length of the list.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle. The handle should be the same as one returned by either the “hpe5022_createSeqBaseSegMove”, the “hpe5022_createSeqBaseSegMoveWriteOffset”, the “hpe5022_createSeqBaseSegMoveReadOffset”, the “hpe5022_createSeqBaseSegErase”, the “hpe5022_createSeqBaseSegWrite”, the “hpe5022_createSeqBaseSegRead” or the “hpe5022_createSeqBaseSegWriteRead” functions.

  Direction — IN

- sectCoun

  Description — Specifies the sector number.

  Direction — IN

- param

  Description — Specifies the type of parameter to be selected.

  Direction — IN

  Value — The parameter to be selected is dependent on the base segment as shown below. Same as the ‘param’ parameter in the “ hpe5022_seqSegParameter” function.

  — hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

— hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- valList

  Description — Returns the base segment parameter values that are set in array. The “hpe5022_seqBaseSegParameterPoint_Q” function returns the size of the array.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.

See Also

“hpe5022_createSeqSeg”

“hpe5022_createSeqBaseSegMove”

“hpe5022_createSeqBaseSegMoveWriteOffset”

“hpe5022_createSeqBaseSegMoveReadOffset”

“hpe5022_createSeqBaseSegErase”

“hpe5022_createSeqBaseSegWrite”

“hpe5022_createSeqBaseSegWriteRead”

“hpe5022_createSeqBaseSegRead”

hpe5022_seqBaseSegGateConfig

C Syntax

ViStatus hpe5022_seqBaseSegGateConfig(ViSession id, ViObject basHndl, ViInt16 gateMode, ViReal64 trigDel, ViReal64 aper);

Visual Basic Syntax

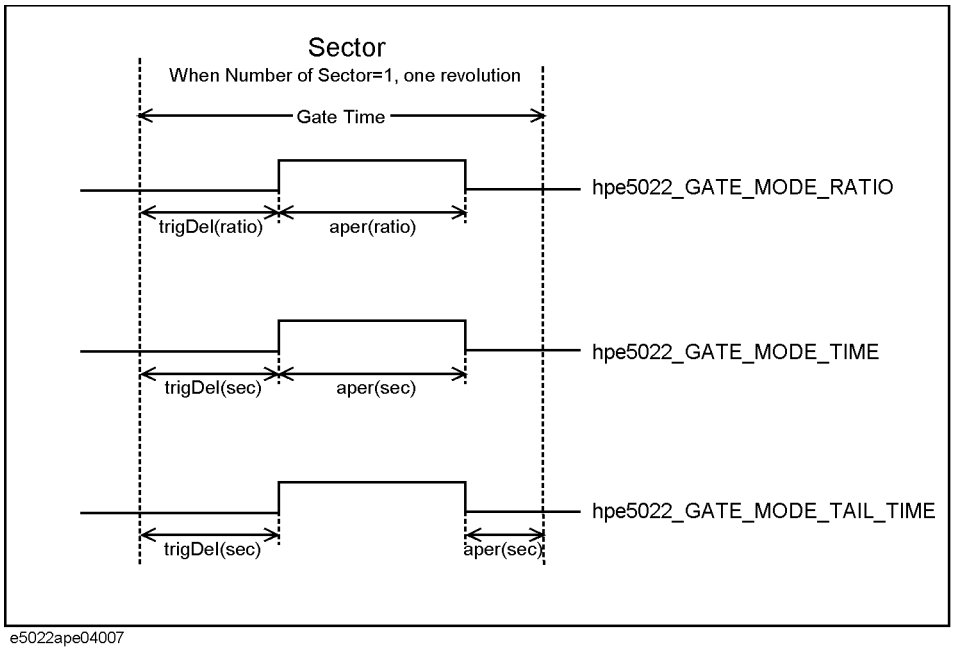
hpe5022_seqBaseSegGateConfig(ByVal id As Long, ByVal basHndl As Long, ByVal gateMode As Integer, ByVal trigDel As Double, ByVal aperAs Double) As Long

Description

This function specifies the gate for base segment. This funciton allows you to read or write data during the specified period on a track. When a base segment is defined by the “hpe5022_createSeqBaseSegWriteRead” function, use the “hpe5022_seqBaseSegWriteReadGateConfig” function instead of this function. This function is only available for use with the E5039C bit error test module.

The definition of trigger delay and aperture is shown in Figure 2-3.

Figure 2-3 Trigger Delay and Aperture



The range of gate are shown below. Since this range is a limitation from the software viewpoint, it might be narrowed depending on its hardware, such as a head amplifier. In other word, you might have an “hpe5022_ERROR_INV_PARAMETER” error even if the value of the parameters, delay and aperture, are within the range. When “hpe5022_setupSeq” or “hpe5022_executeSeq” function is executed, the specified value is checked, then the error is displayed if the value is over the range. If you have the error, change the value of parameters.

- When a module for the base segment is “hpe5022_MODULE_DATA_GEN”

Delay	≥ 0.0 sec.

Aperture	≥ 1.0 µsec.
Delay + Aperture	≤ 25msec.[a] @ 'numOfSec' = 1
	≤ GateTime - 100 µsec. @ 'numOfSec' > 1

a. When 'numOfSec' is set to 1, a value for more than one revolution are accepted. In other word, you can write data for not only one revolution but also next revolution.

- When a module for the base segment is "hpe5022_MODULE_PARAM"

Delay	≥ 1.0 µsec. @ measFunc is set to "hpe5022_MEAS_TAA" or "hpe5022_MEAS_PW"
	≥ 100 µsec. @ measFunc is set to "hpe5022_MEAS_BASELINE"
Aperture	≥ 20 µsec.
Delay + Aperture	≤ GateTime - 30 µsec. @ measFunc is set to "hpe5022_MEAS_TAA" or "hpe5022_MEAS_PW"
	≤ GateTime - 300 µsec @ measFunc is set to "hpe5022_MEAS_BASELINE"

- When a module for the base segment is "hpe5022_MODULE_EXTERNAL"

Delay	≥ 0.0 sec.
Aperture	≥ 1.0 µsec.
Delay + Aperture	≤ GateTime - 100 µsec.

Equation 2-1 Gate Time formula

$$GateTime = \frac{(60/SpindleSpeed[rpm])}{NumberOfSector}$$

The number of sector (numOfSec) is specified by the "hpe5022_createSeq" function.

Parameters

- id

  Description Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction IN

- basHndl

  Description Specifies the base segment handle to set the parameter value. This is given by either the

"hpe5022_createSeqBaseSegErase", "hpe5022_createSeqBaseSegWrite", or "hpe5022_createSeqBaseSegRead" functions.

Direction IN

- gateMode

  Description Specifies the gate mode

  Direction IN

  Value

Parameter	Value	Description
hpe5022_GATE_MODE_AUTO	0	Gate (delay, aperture) is automatically set. In this mode, `trigDel' and `aper' are ignored.
hpe5022_GATE_MODE_RATIO	1	Gate is specified as ratio for gate time.
hpe5022_GATE_MODE_TIME	2	Gate is specified as time. Unit is second.
hpe5022_GATE_MODE_TAIL_TIME	3	Trigger delay is specified as time. Unit is second. Aperture is specified as the time between gate end and next trigger.

- trigDel

  Description Specifies the delay time from a trigger. Range of value is shown below.

gateMode	Value
hpe5022_GATE_MODE_AUTO	N/A
hpe5022_GATE_MODE_RATIO	$0 \leq trigDel < 1.0$
hpe5022_GATE_MODE_TIME	$0 \leq trigDel$
hpe5022_GATE_MODE_TAIL_TIME	$0 \leq trigDel$

  Direction IN

- aper

Description Specifies the aperture time.

gateMode	Value
hpe5022_GATE_MODE_AUTO	N/A
hpe5022_GATE_MODE_RATIO	0 ≤ aper
hpe5022_GATE_MODE_TIME	0 ≤ aper
hpe5022_GATE_MODE_TAIL_TIME	N/A

Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.
hpe5022_ERROR_INV_PARAMETER	The parameters is invalid.

See Also

“hpe5022_seqBaseSegGateConfig_Q”

“hpe5022_seqBaseSegWriteReadGateConfig”

“hpe5022_seqSaGateConfig”

hpe5022_seqBaseSegGateConfig_Q

C Syntax

ViStatus hpe5022_seqBaseSegGateConfig_Q(ViSession id, ViObject basHndl, ViPInt16 gateMode, ViPReal64 trigDel, ViPReal64 aper);

Visual Basic Syntax

hpe5022_seqBaseSegGateConfig_Q(ByVal id As Long, ByVal basHndl As Long, ByRef gateMode As Integer, ByRef trigDel As Double, ByRef aper As Double) As Long

Description

This function returns the gate setting for base segment which is specified by the “hpe5022_seqBaseSegGateConfig” function. This function is only available for use with the E5039C bit error test module.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to set the parameter value. This is given by either the “hpe5022_createSeqBaseSegErase”, “hpe5022_createSeqBaseSegWrite”, or “hpe5022_createSeqBaseSegRead” functions.

  Direction — IN

- gateMode

  Description — Returns the gate mode

  Direction — OUT

  Value — Same as the ‘gateMode’ parameter in the “hpe5022_seqBaseSegGateConfig” function.

- trigDel

  Description — Returns a delay time from a trigger. When ‘gateMode’ set to “hpe5022_GATE_AUTO”, “hpe5022_NAN” is returned.

  Direction — OUT

- aper

  Description — Returns a delay time from a trigger. When ‘gateMode’ set to “hpe5022_GATE_AUTO”, “hpe5022_NAN” is returned.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.

See Also

"hpe5022_seqBaseSegGateConfig"

hpe5022_seqBaseSegWriteReadGateConfig

C Syntax

ViStatus hpe5022_seqBaseSegWriteReadGateConfig(ViSession id, ViObject basHndl, ViInt16 writGateMode, ViReal64 writDel, ViReal64 writAper, ViInt16 readGateMode, ViReal64 readDel,ViReal64 readAper);

Visual Basic Syntax

hpe5022_seqBaseSegWriteReadGateConfig(ByVal id As Long, ByVal basHndl As Long, ByVal writeGateMode As Integer, ByVal writeDel As Double, ByVal writAper As Double, ByVal readGateMode As Integer, ByVal readDel As Double, ByVal readAper As Double) As Long

Description

This function specifies the gate for base segment. This function allows you to read-and-write during the specified period on a track. This function is used for the base segment which is specified by the “hpe5022_createSeqBaseSegWriteRead” function. For the other kind of base segment, use the “hpe5022_seqBaseSegGateConfig” function.

The definition of trigger delay and aperture and its setting range are the same as the “hpe5022_seqBaseSegGateConfig” function. When “hpe5022_setupSeq” or “hpe5022_executeSeq” function is executed, the gate limitation is checked, then “hpe5022_ERROR_INV_PARAMETER” is displayed.

The gate for write and read are independent from each other. A read period can overlap with a write period.

Parameters

- id

  Description Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction IN

- basHndl

  Description Specifies the base segment handle to set the parameter value. This is given by the “hpe5022_createSeqBaseSegWriteRead” function.

  Direction IN

- writGateMode

  Description Specifies the write gate mode.

  Direction IN

  Value

Parameter	Value	Description
hpe5022_GATE_MODE_RATIO	1	Gate is specified as ratio.
hpe5022_GATE_MODE_TIME	2	Gate is specified as time. Unit is second.

Parameter	Value	Description
hpe5022_GATE_MODE_TAIL_TIME	3	Trigger delay is specified as time in sec. Aperture is specified as the time between gate end and next trigger.

- writDel

  Description — Specifies the write delay time from a trigger. Range of value is shown below.

gateMode	Value
hpe5022_GATE_MODE_RATIO	0 ≤ writDel < 1.0
hpe5022_GATE_MODE_TIME	0 ≤ writDel
hpe5022_GATE_MODE_TAIL_TIME	0 ≤ writDel

  Direction — IN

- writAper

  Description — Specifies the write aperture time. Range of value is shown below.

gateMode	Value
hpe5022_GATE_MODE_RATIO	0 ≤ writAper
hpe5022_GATE_MODE_TIME	0 ≤ writAper
hpe5022_GATE_MODE_TAIL_TIME	N/A

  Direction — IN

- readGateMode

  Description — Specifies the read gate mode.

  Direction — IN

  Value — Same as the ‘writGateMode’ parameter

- readDel

  Description — Specifies the read delay time from a trigger. Range of value is shown below.

gateMode	Value
hpe5022_GATE_MODE_RATIO	0 ≤ readDel < 1.0

gateMode	Value
hpe5022_GATE_MODE_TIME	0 ≤ readDel
hpe5022_GATE_MODE_TAIL_TIME	0 ≤ readDel

Direction IN

- readAper

Description Specifies the read aperture time. Range of value is shown below.

gateMode	Value
hpe5022_GATE_MODE_RATIO	0 ≤ readAper
hpe5022_GATE_MODE_TIME	0 ≤ readAper
hpe5022_GATE_MODE_TAIL_TIME	N/A

Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.
hpe5022_ERROR_INV_PARAMETER	The parameter is invalid.

See Also

“hpe5022_seqBaseSegWriteReadGateConfig_Q”

“hpe5022_seqBaseSegGateConfig”

“hpe5022_seqSaGateConfig”

hpe5022_seqBaseSegWriteReadGateConfig_Q

C Syntax

ViStatus hpe5022_seqBaseSegWriteReadGateConfig_Q(ViSession id, ViObject basHndl, ViPInt16 writGateMode, ViPReal64 writDel, ViPReal64 writAper, ViPInt16 readGateMode, ViPReal64 readDel,ViPReal64 readAper);

Visual Basic Syntax

hpe5022_seqBaseSegWriteReadGateConfig_Q(ByVal id As Long, ByVal basHndl As Long, ByVal writeGateMode As Integer, ByVal writeDel As Double, ByVal writAper As Double, ByVal readGateMode As Integer, ByVal readDel As Double, ByVal readAper As Double) As Long

Description

This function returns the gate setting for a read-write base segment which is specified by the “hpe5022_seqBaseSegWriteReadGateConfig” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to set the parameter value. This is given by the “hpe5022_createSeqBaseSegWriteRead” function.

  Direction — IN

- writGateMode

  Description — Returns the write gate mode.

  Direction — OUT

  Value — Same as the ‘writGateMode’ parameter in the “hpe5022_seqBaseSegWriteReadGateConfig” function.

- writDel

  Description — Returns the write delay time from a trigger.

  Direction — OUT

- writAper

  Description — Returns the write aperture time.

  Direction — OUT

- readGateMode

  Description — Returns the read gate mode.

  Direction — OUT

- readDel

Description Returns the read delay time from a trigger.

Direction OUT

- readAper

  Description Returns the read aperture time.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The specified base segment handle is invalid. Check if the base segment handle is correct.

See Also

“hpe5022_seqBaseSegWriteReadGateConfig”

Query Functions

hpe5022_testParameterPoint_Q

C Syntax

ViStatus hpe5022_testParameterPoint_Q (ViSession id, ViObject seqHndl, ViObject segId, ViPInt16 poin);

Visual Basic Syntax

hpe5022_testParameterPoint_Q (ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Integer, ByRef poin As Integer) As Long

Description

This function returns the size of test parameter value list. The value should be the sum of all base segment step multiplied by number of loops of the segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- poin

  Description — Returns the size of test parameter value list array

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The specified segment handle is invalid. Check if the segment handle is correct.

See Also

“hpe5022_createSeq”

“hpe5022_addSeqBaseSeg”

“hpe5022_testParameter_Q”

hpe5022_testParameter_Q

C Syntax

ViStatus hpe5022_testParameter_Q (ViSession id, ViObject seqHndl, ViObject segId, ViInt16 param, ViReal64 valList[]);

Visual Basic Syntax

hpe5022_testParameter_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal param as Integer, ByRef valList As Double) As Long

Description

This function returns the list of parameter value. The “hpe5022_testParameterPoint_Q” function returns the length of the list.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ by the “hpe5022_addSeqSeg” functions.

  Direction — IN

- param

  Description — Specifies the type of parameter to be selected.

  Direction — IN

  Value — The parameter to be selected is dependent on the base segment as shown below. Same as the ‘param’ parameter in the “hpe5022_seqSegParameter” function.

  — hpe5022_createSeqBaseSegMove

Parameter	Value	Description
hpe5022_PARAMETER_HEAD_POS	7	Head Position

  — hpe5022_createSeqBaseSegErase

Parameter	Value	Description
hpe5022_PARAMETER_ERASE_CURR	1	Erase Current

— hpe5022_createSeqBaseSegWrite

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3

— hpe5022_createSeqBaseSegRead

Parameter	Value	Description
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

— hpe5022_createSeqBaseSegWriteRead

Parameter	Value	Description
hpe5022_PARAMETER_WRIT_CURR	2	Write Current
hpe5022_PARAMETER_CHAN_BIT_RATE	0	Channel Bit Rate
hpe5022_PARAMETER_PREC_VAL1	4	Precompensation Delay 1
hpe5022_PARAMETER_PREC_VAL2	5	Precompensation Delay 2
hpe5022_PARAMETER_PREC_VAL3	6	Precompensation Delay 3
hpe5022_PARAMETER_SENS_CURR	3	MR bias (Current)
hpe5022_PARAMETER_SENS_VOLT	9	MR bias (Voltage)[a]
hpe5022_PARAMETER_SENS_POW	10	MR bias (Power)*[a]

a. You can use a MR bias by the voltage or the power only if the installed head amplifier has the capability.

- valList

  Description — Returns the base segment parameter values that are set in array. The “hpe5022_testParameterPoint_Q” function returns the size of the array.

Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The specified segment handle is invalid. Check if the segment handle is correct.

See Also

“hpe5022_createSeqSeg”

“hpe5022_addSeqSeg”

“hpe5022_testParameterPoint_Q”

hpe5022_resultPoint_Q

C Syntax

ViStatus hpe5022_resultPoint_Q (ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViPInt16 loopCoun, ViPInt16 sectCoun, ViPInt16 elemCoun);

Visual Basic Syntax

hpe5022_resultPoint_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef loopCoun As Integer, ByRef sectCoun As Integer, ByRef elemCoun As Integer) As Long

Description

This function returns the size of the data returned by the “hpe5022_result_Q” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the “hpe5022_addSeqBaseSeg” function.

  Direction — IN

- loopCoun

  Description — Returns the number of loops of the specified segment. This number is the same as ‘loopCoun’ specified by “hpe5022_createSeqSeg” function.

  Direction — OUT

- sectCoun

  Description — Returns the number of sector of the sequence. This number is specified by “hpe5022_createSeq”function.

Direction OUT

- elemCoun

  Description Returns the number of results for one measurement point. The returned value is dependent on the read option of the base segment (See “hpe5022_createSeqBaseSegRead” or “hpe5022_createSeqBaseSegWriteRead” functions) as shown below.

Read Option of Base Segment	elemCoun
hpe5022_MEAS_TAA	2
hpe5022_MEAS_PW	2
hpe5022_MEAS_BASELINE	2
hpe5022_MEAS_LEVEL	1

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_createSeq”

“hpe5022_addSeqSeg”

“hpe5022_addSeqBaseSeg”

“hpe5022_result_Q”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_result_Q

C Syntax

ViStatus hpe5022_result_Q (ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViPReal64 resList[]);

Visual Basic Syntax

hpe5022_result_Q(ByVal id As Long,ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef resList As Double) As Long

Description

This function returns the measurement result of the specified base segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the “hpe5022_addSeqBaseSeg” function.

  Direction — IN

- resList

  Description — Returns the array data as the measurement result of the specified base segment. The returned value depends on the read option of base segment (See “hpe5022_createSeqBaseSegRead” or “hpe5022_createSeqBaseSegWriteRead” functions) as shown below.

Read Option of Base Segment	Returned Result
hpe5022_MEAS_TAA	TAA Positive, TAA Negative
hpe5022_MEAS_PW	PW Positive, PW Negative

Read Option of Base Segment	Returned Result
hpe5022_MEAS_BASELINE	Baseline Positive, Baseline Negative
hpe5022_MEAS_LEVEL	Level

The size of array data is returned by the "hpe5022_resultPoint_Q" function. The data size should be equal to the product of loopCoun, sectCoun and elemCoun.

Direction OUT

Unit Volt (hpe5022_MEAS_TAA, hpe5022_MEAS_BASELINE)

Second (hpe5022_MEAS_PW)

dBm (hpe5022_MEAS_LEVEL)

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by 'seqHndl' is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by 'segId' is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by 'basId' is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

"hpe5022_createSeq"

"hpe5022_addSeqSeg"

"hpe5022_addSeqBaseSeg"

"hpe5022_resultPoint_Q"

hpe5022_wavePoint_Q

C Syntax

ViStatus hpe5022_wavePoint_Q (ViSession id, ViObject seqHndl, ViObject segId, ViObject basId,ViPInt32 poin, ViPInt16 elemCoun);

Visual Basic Syntax

hpe5022_wavePoint_Q(ByVal id As Long, ByVal seqHndl As Long,ByVal segId As Long,ByVal basId As Long, ByRef poin As Integer,ByRef elemCoun As Integer) As Long

Description

This function returns the size of the data returned by the “hpe5022_wave_Q” function.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the “hpe5022_addSeqBaseSeg” function.

  Direction — IN

- poin

  Description — Returns the number of measurement points.

  Direction — OUT

- elemCoun

  Description — Returns the number of results for one measurement point. The returned value depends on the read option of the base segment (See “hpe5022_createSeqBaseSegRead” or “hpe5022_createSeqBaseSegWriteRead” functions) as

shown below.

Read Option of Base Segment	elemCoun
hpe5022_MEAS_TAA	2
hpe5022_MEAS_PW	2
hpe5022_MEAS_BASELINE	2
hpe5022_MEAS_WAVE	Depend on “hpe5022_channelBitRate”, “hpe5022_waveAverage”, “hpe5022_waveOverSampleRate”
hpe5022_MEAS_SPEC	Depend on start/stop frequencies and resolution band width They are specified by the “hpe5022_spectrumFrequency”, “hpe5022_spectrumBandWidth” or “hpe5022_seqConfiguration” functions.

Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_createSeq”

“hpe5022_addSeqSeg”

“hpe5022_addSeqBaseSeg”

“hpe5022_wave_Q”

“hpe5022_createSeqBaseSegRead”

“hpe5022_createSeqBaseSegWriteRead”

hpe5022_wave_Q

C Syntax

ViStatus hpe5022_wave_Q (ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViPReal64 wave[])

Visual Basic Syntax

hpe5022_wave_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef wave As Double) As Long

Description

This function returns the measurement result of the specified base segment. Generally, this function allows you to get the waveform data of the oscilloscope. You can also get the raw data of TAA, PW and histogram data of the baseline.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be same as ‘basId’ returned by the “hpe5022_addSeqBaseSeg” function.

  Direction — IN

- wave

  Description — Returns the array data as the measurement result of the specified base segment. The returned value is dependent on the read option of base segment (See “hpe5022_createSeqBaseSegRead” or “hpe5022_createSeqBaseSegWriteRead” functions) as shown below.

Read Option of Base Segment	Returned Result
hpe5022_MEAS_TAA	TAA Raw data Positive, TAA Raw Data Negative
hpe5022_MEAS_PW	PW Raw Data Positive, PW Raw Data Negative
hpe5022_MEAS_BASELINE	Baseline histogram Positive Data, Baseline histogram Negative Data
hpe5022_MEAS_WAVE	Waveform
hpe5022_MEAS_SPEC	Spectrum data

The size of array data is returned by the "hpe5022_wavePoint_Q" function. The number of data is equal to the product of elemCoun and poin.

Direction OUT

Unit Volt (TAA, PW, Baseline, waveform)

dBm (Spectrum Data)

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by 'seqHndl' is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by 'segId' is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by 'basId' is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

"hpe5022_createSeq"

"hpe5022_addSeqSeg"

"hpe5022_addSeqBaseSeg"

"hpe5022_wavePoint_Q"

hpe5022_measStatus_Q

C Syntax

ViStatus hpe5022_measStatus_Q (ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViStatus statList[])

Visual Basic Syntax

hpe5022_measStatus_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef statList As Double) As Long

Description

This function returns the measurement status of each measurement sequence in the uds, i.e, overflow, underflow etc.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle. The handle should be the same as ‘seqHndl’ returned by the “hpe5022_createSeq” function.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the “hpe5022_addSeqSeg” function.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the “hpe5022_addSeqBaseSeg” function.

  Direction — IN

- statList

  Description — Returns the array data as the measurement result of the specified base segment. The returned value is dependent on the read option of the base segment.

Error Code	Returned Result
hpe5022_ERROR_OVERFLOW	An overflow is detected in the parametric module. Check if the parameter setting is correct.
hpe5022_ERROR_UNDERFLOW	An underflow is detected in the parametric module. Check if the parameter setting is correct.
hpe5022_ERROR_PLL_UNLOCK	The PLL is unlocked. Check if parameter setting is correct.
hpe5022_ERROR_THERMAL_ASPERITY	Thermal Asperity Detected. Check the head and media. This error occurs when the overflow is detected but the average of measurement is under the limit.

Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_createSeq”

“hpe5022_addSeqSeg”

“hpe5022_addSeqBaseSeg”

BER Functions

hpe5022_BER_seqSegRegister

C Syntax

ViStatus hpe5022_BER_seqSegRegister(ViSession id, ViObject segHndl, ViInt16 elemCoun, const ViInt32 addr[], const ViInt32 upperBitPos[], const ViInt32 lowerBitPos[], const ViInt32 data[]);

Visual Basic Syntax

hpe5022_BER_seqSegRegister(ByVal id As Long, ByVal segHndl As Long, ByVal elemCoun As Integer, ByRef addr As Long, ByRef upperBitPos As Long, ByRef lowerBitPos As Long, ByRef data As Long) As Long

Description

This function sets the constant parameter value to a segment. You can change the number of registers (i.e. ‘elemCoun’) from corresponding base segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle added to the sequence. This is given by the function “hpe5022_createSeqSeg” on page 113.

  Direction — IN

- elemCoun

  Description — Specifies the number of setting registers per read or write operation.

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_ELEM_COUN_MIN	1
hpe5022_BER_REG_ELEM_COUN_MAX	16

- addr

  Description — Specifies the channel IC register address array. If this parameter is set to “hpe5022_BER_REG_ADDR_NULL,” the register setups are not executed. The array size is specified by ‘elemCoun.’

  Direction — IN

Values

Name	Value
hpe5022_BER_REG_ADDR_NULL	–1
hpe5022_BER_REG_ADDR_MIN	0
hpe5022_BER_REG_ADDR_MAX	0x3ffff

- upperBitPos

  Description Specifies the upper bit position array of the segment register. The upper limit of this parameter depends on the installed read channel IC and is checked when “hpe5022_setupSeq” on page 94 or “hpe5022_executeSeq” on page 98 is called. The array size is specified by ‘elemCoun.’

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- lowerBitPos

  Description Specifies the lower bit position array of the segment register. The upper limit of this parameter is the same as that of ‘upperBitPos.’ The array size is specified by ‘elemCoun.’

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- data

  Description Specifies the channel IC register value array. The array size is specified by ‘elemCoun.’

  Direction IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘elemCoun,’ ‘addr,’ ‘upperBitPos,’ lowerBitPos,’ or ‘data’ is out of range.

See Also

“hpe5022_BER_seqSegRegisterSweep” on page 188

hpe5022_BER_seqSegRegisterSweep

C Syntax

ViStatus hpe5022_BER_seqSegRegisterSweep(ViSession id, ViObject segHndl, ViInt16 elemCoun, const ViInt32 addr[], const ViInt32 upperBitPos[], const ViInt32 lowerBitPos[], const ViInt32 startData[], const ViInt32 stopData[]);

Visual Basic Syntax

hpe5022_BER_seqSegRegisterSweep(ByVal id As Long, ByVal segHndl As Long, ByVal elemCoun As Integer, ByRef addr As Long, ByRef upperBitPos As Long, ByRef lowerBitPos As Long, ByRef startData As Long, ByRef stopData As Long) As Long

Description

This function sets the register sweep parameter values to a segment.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle added to the sequence. This is given by the function “hpe5022_createSeqSeg” on page 113.

  Direction — IN

- elemCoun

  Description — Specifies the number of setting registers per read or write operation.

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_ELEM_COUN_MIN	1
hpe5022_BER_REG_ELEM_COUN_MAX	16

- addr

  Description — Specifies the channel IC register address array. If this parameter is set to “hpe5022_BER_REG_ADDR_NULL,” the register setups are not executed. The array size of this parameter is specified by ‘elemCoun.’

  Direction — IN

Values

Name	Value
hpe5022_BER_REG_ADDR_NULL	–1
hpe5022_BER_REG_ADDR_MIN	0
hpe5022_BER_REG_ADDR_MAX	0x3ffff

- upperBitPos

  Description — Specifies the upper bit position array of the segment register. The upper limit of this parameter depends on the installed read channel IC and is checked when “hpe5022_setupSeq” on page 94 or “hpe5022_executeSeq” on page 98 is called. The array size of this parameter is specified by ‘elemCoun.’

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- lowerBitPos

  Description — Specifies the lower bit position array of the segment register. The upper limit of this parameter is the same as that of ‘upperBitPos.’ The array size of this parameter is specified by ‘elemCoun.’

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- startData

  Description — Specifies the start parameter value of the sweep. The array size of this parameter is specified by ‘elemCoun.’

  Direction — IN

Values

Name	Value
hpe5022_BER_REG_DATA_MIN	0
hpe5022_BER_REG_DATA_MAX	65535

- stopData

  Description Specifies the stop parameter value of the sweep.

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_MIN	0
hpe5022_BER_REG_DATA_MAX	65535

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘elemCoun,’ ‘addr,’ ‘upperBitPos,’ lowerBitPos,’ ‘startData,’ or ‘stopData’ is out of range.

See Also

“hpe5022_BER_seqSegRegister” on page 185

hpe5022_BER_seqSegRegisterList

C Syntax

ViStatus hpe5022_BER_seqSegRegisterList(ViSession id, ViObject segHndl, ViInt16 sweepCoun, ViInt16 elemCoun, const ViInt32 addrList[], const ViInt32 upperBitPosList[], const ViInt32 lowerBitPosList[], const ViInt32 dataList[]);

Visual Basic Syntax

hpe5022_BER_seqSegRegister(ByVal id As Long, ByVal segHndl As Long, ByVal sweepCoun, ByVal elemCoun As Integer, ByRef addrList As Long, ByRef upperBitPosList As Long, ByRef lowerBitPosList As Long, ByRef dataList As Long) As Long

Description

This function sets the parameter value list to a segment. If the length of the sweep counts is less than the number of sweep points, the register setups are not executed for the rest of the sweep.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to set the parameter value. This is given by the function “hpe5022_createSeqSeg” on page 113.

  Direction — IN

- sweepCoun

  Description — Specifies the length of the sweep counts. This value must be greater than 1.

  Direction — IN

- elemCoun

  Description — Specifies the number of setting registers per read or write operation.

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_ELEM_COUN_MIN	1
hpe5022_BER_REG_ELEM_COUN_MAX	16

- addrList

  Description — Specifies the channel IC register address array. If this value

is set to (hpe5022_BER_REG_ADDR_NULL), the register setting is not executed. The list length is ‘sweepCoun’ × ‘elemCoun.’

Direction IN

Values

Name	Value
hpe5022_BER_REG_ADDR_NULL	–1
hpe5022_BER_REG_ADDR_MIN	0
hpe5022_BER_REG_ADDR_MAX	0x3ffff

- upperBitPosList

  Description Specifies the upper bit position array of the segment register. The upper limit of this parameter depends on the installed read channel IC and is checked when “hpe5022_setupSeq” or “hpe5022_executeSeq” is called. The list length is ‘sweepCoun’ × ‘elemCoun.’

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- lowerBitPosList

  Description Specifies the lower bit position array of the segment register. The upper limit of this parameter is the same as that of upperBitPosList. The list length is ‘sweepCoun’ × ‘elemCoun.’

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- dataList

  Description Specifies the channel IC register value array. The list length is ‘sweepCoun’ × ‘elemCoun.’

Direction IN

Values

Name	Value
hpe5022_BER_REG_DATA_MIN	0
hpe5022_BER_REG_DATA_MAX	65535

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter ‘sweepCoun,’ ‘elemCoun,’ ‘addrList,’ ‘upperBitPosList,’ ‘lower BitPosList,’ or ‘dataList’ is out of range.

See Also

“hpe5022_BER_seqSegRegister” on page 185

“hpe5022_BER_seqSegRegisterSweep” on page 188

hpe5022_BER_seqSegRegisterPoint_Q

C Syntax

ViStatus hpe5022_BER_seqSegRegisterPoint_Q(ViSession id, ViObject segHndl, ViPInt16 poin);

Visual Basic Syntax

hpe5022_BER_seqSegRegisterPoint_Q(ByVal id As Long, ByVal segHndl As Long, ByRef poin As Integer) As Long

Description

This function returns the query list length of the segment parameter value.

Parameters

- id

  Description — Specifies the system identifier. This is given by the "hpe5022_init" function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle to set the parameter value. This is given by the function "hpe5022_createSeqSeg" on page 113.

  Direction — IN

- poin

  Description — Returns the list length of the segment parameter value.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by 'segHndl' is invalid.

See Also

"hpe5022_BER_seqSegRegister" on page 185

"hpe5022_BER_seqSegRegisterSweep" on page 188

"hpe5022_BER_seqSegRegisterList" on page 191

hpe5022_BER_seqSegRegister_Q

C Syntax

ViStatus hpe5022_BER_seqSegRegister_Q(ViSession id, ViObject segHndl, ViPInt16 elemCoun, ViInt32 addrList[], ViInt32 upperBitPosList[], ViInt32 lowrBitPosList[], ViInt32 dataList[]);

Visual Basic Syntax

hpe5022_BER_seqSegRegister_Q(ByVal id As Long, ByVal segHndl As Long, ByRef elemCoun As Integer, ByRef addrList As Long, ByRef upperBitPosList As Long, ByRef lowerBitPosList As Long, ByRef dataList As Long) As Long

Description

This function returns the list of segment register parameter values.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- segHndl

  Description — Specifies the segment handle added to the sequence. This is given by the function “hpe5022_createSeqSeg” on page 113.

  Direction — IN

- elemCoun

  Description — Returns the number of setting registers per read or write operation.

  Direction — OUT

- addrList

  Description — Returns the channel IC register address array. The list length is the value returned by “hpe5022_BER_seqSegRegisterPoint_Q” on page 194.

  Direction — OUT

- upperBitPosList

  Description — Returns the upper bit position array of the segment register. The list length is the value returned by “hpe5022_BER_seqSegRegisterPoint_Q” on page 194.

  Direction — OUT

- lowerBitPosList

  Description — Returns the lower bit position array of the segment register. The list length is the value returned by “hpe5022_BER_seqSegRegisterPoint_Q” on page 194.

  Direction — OUT

- dataList

  Description — Returns the channel IC register value array. The list length is the value returned by "hpe5022_BER_seqSegRegisterPoint_Q" on page 194.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by 'segHndl' is invalid.

See Also

"hpe5022_BER_seqSegRegister" on page 185"

hpe5022_BER_seqBaseSegRegister

C Syntax

ViStatus hpe5022_BER_seqBaseSegRegister(ViSession id, ViObject basHndl, ViInt16 elemCoun, const ViInt32 addr[], const ViInt32 upperBitPos[], const ViInt32 lowerBitPos[], const ViInt32 data[], ViBoolean fixed);

Visual Basic Syntax

hpe5022_BER_seqBaseSegRegister(ByVal id As Long, ByVal basHndl As Long, ByVal elemCoun As Integer, ByRef addr As Long, ByRef upperBitPos As Long, ByRef lowerBitPos As Long, ByRef data As Long, ByVal fixed As Integer) As Long

Description

This function sets constant parameter values to base segment registers.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to be added to the segment. This is given by either “hpe5022_createSeqBaseSegWrite” or “hpe5022_createSeqBaseSegRead” functions.

  Direction — IN

- elemCoun

  Description — Specifies the number of setting registers per read or write operation.

  Direction — IN

  Values

Name	Value
hpe5022_BER_REG_ELEM_COUN_MIN	1
hpe5022_BER_REG_ELEM_COUN_MAX	16

- addr

  Description — Specifies the channel IC register address. If this parameter is set to “hpe5022_BER_REG_ADDR_NULL,” the register setups are not executed.

  Direction — IN

Values

Name	Value
hpe5022_BER_REG_ADDR_NULL	–1
hpe5022_BER_REG_ADDR_MIN	0
hpe5022_BER_REG_ADDR_MAX	0x3ffff

- upperBitPos

  Description Specifies the upper bit position array of the segment register. The upper limit of this parameter depends on the installed read channel IC and is checked when “hpe5022_setupSeq” on page 94 or “hpe5022_executeSeq” on page 98 is called.

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- lowerBitPos

  Description Specifies the lower bit position array of the segment register. The upper limit of this parameter is the same as that of ‘upperBitPos.’

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_BIT_MIN	0
hpe5022_BER_REG_DATA_BIT_MAX	15

- data

  Description Specifies the channel IC register value array.

  Direction IN

  Values

Name	Value
hpe5022_BER_REG_DATA_MIN	0
hpe5022_BER_REG_DATA_MAX	65535

- fixed

  Description — If 'fixed' is VI_TRUE, base segment parameter value is not affected by the segment parameter value.

  Direction — IN

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_NOT_INIT	The resource of Agilent E5039A/B/C can not be opened during initialization. Check if Agilent E5039A/B/C is included in the rsrcArray of the "hpe5022_init" function.
hpe5022_ERROR_INV_BAS_HNDL	The handle specified by 'basHndl' is invalid.
hpe5022_ERROR_INV_PARAMETER	The parameter 'elemCoun,' 'addr,' 'upperBitPos,' lowerBitPos,' 'data,' or 'fixed' is out of range.

See Also

"hpe5022_BER_seqBaseSegRegister_Q" on page 201

hpe5022_BER_seqBaseSegRegisterPoint_Q

C Syntax

ViStatus hpe5022_BER_seqBaseSegRegisterPoint_Q(ViSession id, ViObject basHndl, ViPInt16 poin);

Visual Basic Syntax

hpe5022_BER_seqBaseSegRegisterPoint_Q(ByVal id As Long, ByVal basHndl As Long, ByRef poin As Integer) As Long

Description

This function returns the query list length of the base segment parameter value.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to be added to the segment. This is given by either “hpe5022_createSeqBaseSegWrite” or “hpe5022_createSeqBaseSegRead” function.

  Direction — IN

- poin

  Description — Returns the list length of the base segment parameter value.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The handle specified by ‘basHndl’ is invalid.

See Also

“hpe5022_BER_seqBaseSegRegister” on page 197

hpe5022_BER_seqBaseSegRegister_Q

C Syntax

ViStatus hpe5022_BER_seqBaseSegRegister_Q(ViSession id, ViObject basHndl, ViPInt16 elemCoun, ViInt32 addr[], ViInt32 upperBitPos[], ViInt32 lowerBitPos[], ViInt32 data[]);

Visual Basic Syntax

hpe5022_BER_seqBaseSegRegister_Q(ByVal id As Long, ByVal basHndl As Long, ByRef elemCoun As Integer, ByRef addr As Long, ByRef upperBitPos As Long, ByRef lowerBitPos As Long, ByRef data As Long) As Long

Description

This function returns the list of base segment register parameter values.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- basHndl

  Description — Specifies the base segment handle to be added to the segment. This is given by either “hpe5022_createSeqBaseSegWrite” or “hpe5022_createSeqBaseSegRead” function.

  Direction — IN

- elemCoun

  Description — Returns the number of setting registers per read or write operation.

  Direction — OUT

- addr

  Description — Returns the channel IC register address array. The array size is the value returned by “hpe5022_BER_seqBaseSegRegisterPoint_Q” on page 200.

  Direction — OUT

- upperBitPos

  Description — Returns the upper bit position array of the segment register. The array size is the value returned by “hpe5022_BER_seqBaseSegRegisterPoint_Q” on page 200.

  Direction — OUT

- lowerBitPos

  Description — Returns the lower bit position array of the segment register. The array size is the value returned by “hpe5022_BER_seqBaseSegRegisterPoint_Q” on page 200.

Direction OUT

- data

  Description Returns the channel IC register value array. The array size is the value returned by “hpe5022_BER_seqBaseSegRegisterPoint_Q” on page 200.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_NOT_INIT	The resource of Agilent E5039A/B/C can not be opened during initialization. Check if Agilent E5039A/B/C is included in the rsrcArray of the “hpe5022_init” function.
hpe5022_ERROR_INV_BAS_HNDL	The handle specified by ‘basHndl’ is invalid.

See Also

“hpe5022_BER_seqBaseSegRegister” on page 197“

hpe5022_BER_testRegisterPoint_Q

C Syntax

ViStatus hpe5022_BER_testRegisterPoint_Q(ViSession id, ViObject seqHndl, ViObject segId, ViPInt16 poin);

Visual Basic Syntax

hpe5022_BER_testRegisterPoint_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByRef poin As Integer) As Long

Description

This function returns the length of test parameter value list.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- poin

  Description — Returns the length of test parameter value list.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_BAS_HNDL	The handle specified by ‘basHndl’ is invalid.

See Also

“hpe5022_BER_testRegister_Q” on page 204

hpe5022_BER_testRegister_Q

C Syntax

ViStatus hpe5022_BER_testRegister_Q(ViSession id, ViObject seqHndl, ViObject segId, ViPInt16 elemCoun, ViInt32 addrList[], ViInt32 upperBitPosList[], ViInt32 lowerBitPosList[], ViInt32 dataList[]);

Visual Basic Syntax

hpe5022_BER_testRegister_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByRef elemCoun As Integer, ByRef addrList As Long, ByRef upperBitPosList As Long, ByRef lowerBitPosList As Long, ByRef dataList As Long) As Long

Description

This function returns the list of test parameter values.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- elemCoun

  Description — Returns the number of setting registers per read or write operation.

  Direction — OUT

- addrList

  Description — Returns the channel IC register address array. The array size is the value returned by “hpe5022_BER_testRegisterPoint_Q” on page 203.

  Direction — OUT

- upperBitPosList

  Description — Returns the upper bit position array of the register. The array size is the value returned by “hpe5022_BER_testRegisterPoint_Q” on page 203.

Direction OUT

- lowerBitPosList

  Description Returns the lower bit position array of the register The array size is the value returned by “hpe5022_BER_testRegisterPoint_Q” on page 203..

  Direction OUT

- dataList

  Description Returns the channel IC register value array.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEG_HNDL	The handle specified by ‘segHndl’ is invalid.

See Also

“hpe5022_BER_testRegisterPoint_Q” on page 203“

hpe5022_BER_errorPoint_Q

C Syntax

ViStatus hpe5022_BER_errorPoint_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViPInt32 points);

Visual Basic Syntax

hpe5022_BER_errorPoint_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef points As Long) As Long

Description

This function returns the result size (number of points) of the BER raw data.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- points

  Description — Returns the result size (number of points) of the BER raw data.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also “hpe5022_BER_error_Q” on page 208“

hpe5022_BER_error_Q

C Syntax

ViStatus hpe5022_BER_error_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViInt32 sector[], ViInt32 locate[], ViInt32 readData[], ViInt32 writeData[]);

Visual Basic Syntax

hpe5022_BER_error_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByRef sector As Long, ByRef locate As Long, ByRef readData As Long, ByRef writeData As Long) As Long

Description

This function returns the error data. When the E5039A/B is used as a bit error test module, and the number of sectors is 1, ‘readData’ and ‘writeData’ are always 0 (‘sector’ and ‘locate’ are the only data obtained).

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- sector

  Description — Returns the sector number array where an error occurs. Sector number of the first sector is 0. The array size is the value returned by “hpe5022_BER_errorPoint_Q” on page 206.

  Direction — OUT

- locate

  Description — Returns the location array where an error occurs. Byte

number of the first data byte is 0. The array size is the value returned by “hpe5022_BER_errorPoint_Q” on page 206.

Direction OUT

- readData

  Description Returns the read back data array. The array size is the value returned by “hpe5022_BER_errorPoint_Q” on page 206.

  Direction OUT

- writeData

  Description Returns the write data array. The array size is the value returned by “hpe5022_BER_errorPoint_Q” on page 206.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_BER_errorPoint_Q” on page 206

hpe5022_BER_sectorErrorPoint_Q

C Syntax

ViStatus hpe5022_BER_sectorErrorPoint_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViInt32 sector, ViPInt32 points);

Visual Basic Syntax

hpe5022_BER_sectorErrorPoint_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByVal sector As Long, ByRef points As Long) As Long

Description

This function returns the result size (number of points) of the BER raw data.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- sector

  Description — Specifies the sector number.

  Direction — IN

- points

  Description — Returns the result size (number of points) of the BER raw data.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_BER_sectorError_Q” on page 212“

hpe5022_BER_sectorError_Q

C Syntax

ViStatus hpe5022_BER_sectorError_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViInt32 sector, ViInt32 locate[], ViInt32 readData[], ViInt32 writeData[]);

Visual Basic Syntax

hpe5022_BER_sectorError_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByVal sector As Long, ByRef locate As Long, ByRef readData As Long, ByRef writeData As Long) As Long

Description

This function returns the error data of specified sector. When the E5039A/B is used as a bit error test module, and the number of sectors is 1, ‘readData’ and ‘writeData’ are always 0 (‘locate’ is the only data obtained).

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- sector

  Description — Specifies the sector number.

  Direction — IN

- locate

  Description — Returns the location array where an error occurs. Byte number of the first data byte is 0. The array size is the value returned by “hpe5022_BER_sectorErrorPoint_Q” on page 210.

Direction OUT

- readData

  Description Returns the read back data array. The array size is the value returned by “hpe5022_BER_sectorErrorPoint_Q” on page 210.

  Direction OUT

- writeData

  Description Returns the write data array. The array size is the value returned by “hpe5022_BER_sectorErrorPoint_Q” on page 210.

  Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_BER_sectorErrorPoint_Q” on page 210

hpe5022_BER_sectorDataPoint_Q

C Syntax

ViStatus hpe5022_BER_sectorDataPoint_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViInt32 sector, ViPInt32 points);

Visual Basic Syntax

hpe5022_BER_sectorDataPoint_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByVal sector As Long, ByRef points As Long) As Long

Description

This function returns the result size (number of points) of the BER raw data. This function is only available for use with the E5039C bit error test module.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- sector

  Description — Specifies the sector number.

  Direction — IN

- points

  Description — Returns the result size (number of points) of the BER raw data.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by ‘segId’ is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by ‘basId’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.

See Also

“hpe5022_BER_sectorData_Q” on page 216“

hpe5022_BER_sectorData_Q

C Syntax

ViStatus hpe5022_BER_sectorData_Q(ViSession id, ViObject seqHndl, ViObject segId, ViObject basId, ViInt32 sector, ViInt32 readData[], ViInt32 writeData[]);

Visual Basic Syntax

hpe5022_BER_sectorError_Q(ByVal id As Long, ByVal seqHndl As Long, ByVal segId As Long, ByVal basId As Long, ByVal sector As Long, ByRef readData As Long, ByRef writeData As Long) As Long

Description

This function returns the read and write data of specified sector. This function is only available for use with the E5039C bit error test module.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- segId

  Description — Specifies the segment ID in the specified sequence. The ID should be the same as ‘segId’ returned by the function “hpe5022_addSeqSeg” on page 99.

  Direction — IN

- basId

  Description — Specifies the base segment ID in the specified segment. The ID should be the same as ‘basId’ returned by the function “hpe5022_addSeqBaseSeg” on page 115.

  Direction — IN

- sector

  Description — Specifies the sector number.

  Direction — IN

- readData

  Description — Returns the read back data array. The array size is the value returned by “hpe5022_BER_sectorDataPoint_Q” on page 214.

  Direction — OUT

- writeData

Description Returns the write data array. The array size is the value returned by "hpe5022_BER_sectorDataPoint_Q" on page 214.

Direction OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by 'seqHndl' is invalid.
hpe5022_ERROR_INV_SEG_ID	The ID specified by 'segId' is invalid.
hpe5022_ERROR_INV_BAS_ID	The ID specified by 'basId' is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.
hpe5022_ERROR_INACCURATE_BER	The error memory overflow occurs.

See Also

"hpe5022_BER_sectorDataPoint_Q" on page 214

hpe5022_BER_errorHistogram_Q

C Syntax

ViStatus hpe5022_BER_errorHistogram_Q(ViSession id, ViObject seqHndl, ViInt32 data[]);

Visual Basic Syntax

hpe5022_BER_sectorError_Q(ByVal id As Long, ByVal seqHndl As Long, ByRef data As Long) As Long

Description

This function returns the result list of wave data. This function is only available for use with the E5039A/B bit error test module.

Parameters

- id

  Description — Specifies the system identifier. This is given by the “hpe5022_init” function.

  Direction — IN

- seqHndl

  Description — Specifies the sequence handle to set up. This is given by the function “hpe5022_createSeq” on page 92.

  Direction — IN

- data

  Description — Returns the result list of wave data. The length of the list is fixed to 10.

  Direction — OUT

Return Values

Completion Code	Description
VI_SUCCESS	No Error

Error Code	Description
hpe5022_ERROR_INV_SEQ_HNDL	The handle specified by ‘seqHndl’ is invalid.
hpe5022_ERROR_DATA_CORRUPT	The data is corrupt.
hpe5022_ERROR_NSUP_FUNC	This function is not supported by the module currently in use.

See Also

A

add
  sequence, 99

E

execute
  sequence, 98

H

hpe5022_addSeqBaseSeg, 115
hpe5022_addSeqSeg, 99
hpe5022_BER_error_Q, 208
hpe5022_BER_errorHistogram_Q, 218
hpe5022_BER_errorPoint_Q, 206
hpe5022_BER_sectorData_Q, 216
hpe5022_BER_sectorDataPoint_Q, 214
hpe5022_BER_sectorError_Q, 212
hpe5022_BER_sectorErrorPoint_Q, 210
hpe5022_BER_seqBaseSegRegister, 197
hpe5022_BER_seqBaseSegRegister_Q, 201
hpe5022_BER_seqBaseSegRegisterPoint_Q, 200
hpe5022_BER_seqSegRegister, 185
hpe5022_BER_seqSegRegister_Q, 195
hpe5022_BER_seqSegRegisterList, 191
hpe5022_BER_seqSegRegisterPoint_Q, 194
hpe5022_BER_seqSegRegisterSweep, 188
hpe5022_BER_testRegister_Q, 204
hpe5022_BER_testRegisterPoint_Q, 203
hpe5022_createSeq, 92
hpe5022_createSeqBaseSegErase, 134
hpe5022_createSeqBaseSegMove, 130
hpe5022_createSeqBaseSegMoveReadOffset, 133
hpe5022_createSeqBaseSegMoveWriteOffset, 132
hpe5022_createSeqBaseSegRead, 139
hpe5022_createSeqBaseSegWrite, 136
hpe5022_createSeqBaseSegWriteRead, 143
hpe5022_createSeqSeg, 113
hpe5022_deleteSeq, 97
hpe5022_deleteSeqBaseSeg, 147
hpe5022_deleteSeqSeg, 114
hpe5022_executeSeq, 98
hpe5022_freeSeq, 96
hpe5022_measStatus_Q, 183
hpe5022_result_Q, 177
hpe5022_resultPoint_Q, 175
hpe5022_seqBaseSegGateConfig, 158
hpe5022_seqBaseSegGateConfig_Q, 162
hpe5022_seqBaseSegParameter, 149
hpe5022_seqBaseSegParameter_Q, 155
hpe5022_seqBaseSegParameterPoint_Q, 153
hpe5022_seqBaseSegWriteReadGateConfig, 164
hpe5022_seqBaseSegWriteReadGateConfig_Q, 167
hpe5022_seqConfiguration, 101
hpe5022_seqConfiguration_Q, 107
hpe5022_seqSaGateConfig, 108
hpe5022_seqSaGateConfig_Q, 111
hpe5022_seqSegParameter, 117
hpe5022_seqSegParameter_Q, 127
hpe5022_seqSegParameterList, 123
hpe5022_seqSegParameterPoint_Q, 126
hpe5022_seqSegParameterSweep, 120
hpe5022_setupSeq, 94
hpe5022_testParameter_Q, 172
hpe5022_testParameterPoint_Q, 170
hpe5022_wave_Q, 181
hpe5022_wavePoint_Q, 179

O

Overwrite (UDS), 66

R

Resolution (UDS), 82

S

sequence
  add, 99
  execute, 98
SNR (UDS), 75
Stability (UDS), 50

T

TAA and PW (UDS), 44

W

Write Current Sweep (UDS), 59

REGIONAL SALES AND SUPPORT OFFICES

For more information about Agilent Technologies test and measurement products, applications, services, and for a current sales office listing, visit our web site: http://www.agilent.com/find/tmdir. You can also contact one of the following centers and ask for a test and measurement sales representative. *11/29/99*

United States:
Agilent Technologies
Test and Measurement Call Center
P.O.Box 4026
Englewood, CO 80155-4026
(tel) 1 800 452 4844

Canada:
Agilent Technologies Canada Inc.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
(tel) 1 877 894 4414

Europe:
Agilent Technologies
Test & Measurement
European Marketing Organization
P.O.Box 999
1180 AZ Amstelveen
The Netherlands
(tel) (31 20) 547 9999

Japan:
Agilent Technologies Japan Ltd.
Call Center
9-1, Takakura-Cho, Hachioji-Shi,
Tokyo 192-8510, Japan
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
Agilent Technologies
Latin American Region Headquarters
5200 Blue Lagoon Drive, Suite #950
Miami, Florida 33126
U.S.A.
(tel) (305) 267 4245
(fax) (305) 267 4286

Australia/New Zealand:
Agilent Technologies Australia Pty Ltd
347 Burwood Highway
Forest Hill, Victoria 3131
(tel) 1-800 629 485 (Australia)
(fax) (61 3) 9272 0749
(tel) 0 800 738 378 (New Zealand)
(fax) (64 4) 802 6881

Asia Pacific:
Agilent Technologies
24/F, Cityplaza One, 1111 King's Road,
Taikoo Shing, Hong Kong
(tel) (852)-3197-7777
(fax) (852)-2506-9284