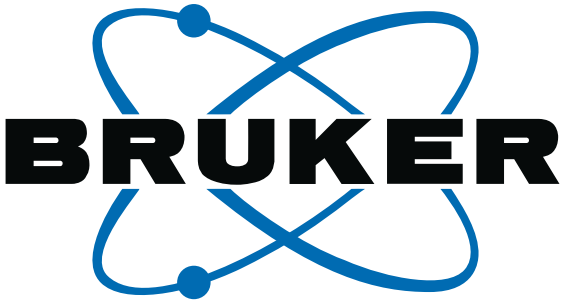


ProDeL

Procedure Description Language

Command Reference Guide

ELEXSYS User.s Guide for the ProDeL programming environment
Manual Version 1.0

Forward

The following is a compilation of the help documentation available from within the ProDeL programming environment. The aim of this document is not to describe how to program with ProDeL, but to provide the user with a handy linked reference to the commands available from within ProDeL. Example programs are provided for an overview of dataset creation and mainpulation as well as experiment control. For a more detailed view of the ProDeL programming environment, refer to part D of the Xepr User's Manual.

While the original help files were copied directly, typographical errors due to reformatting and arrangement may have appeared. If by following the included help, a syntax error has occurred, then refer to the help from within the ProDeL programming environment to discover the source of the error.

Control of the spectrometer acquisition and acquisition parameters is dependent on the type of experiment which has been created in Xepr. An exhaustive overview of each type of experiment is therefore not possible. Examples are provided to illustrate ways of controlling standard CW and Pulse experiments.

Contents

Sections

General ProDeL Terms & Vocables

SEE ALSO

Introduction

PRODEL (PROcedure DEscription Language)

This term refers to a programming language embedded into the Xepr program. Using this language, you can

- access datasets,
- access individual data points,
- perform arbitrary calculations and string operations,
- control experiments and their parameters,
- call the Xepr command interpreter, and
- display your results in the Viewing & Processing Panel.

For more information, refer to the ProDeL specification in the Xepr User Manual.

Reserved PRODEL Keywords

program	program start mark
while, endwhile	loop construct
if, else, endif	conditional branch
label, goto	unconditional branch
return	program termination
real, int, short, char, pointer, boolean	variable types

PRODEL Operators

operators in order of precedence:

-, not()	negation, logical negation
^	power
*, /, div, mod	product, division, integer division, modulo
+, -	sum, difference
<, <=, >=, >, ==, <>	less than, less than or equal to, greater then or equal to, greater than, equal, different
and	logical and
or	logical or

Operators of same precedence are evaluated from left to right. Set parentheses to change the order of evaluation.

Variable Names

Names of ProDeL variables can contain any number of letters, digits or underscore ("_") characters. The first character must be a letter. The name cannot be one of the reserved keywords.

Valid names are:

name, total1, nr_of_points

Invalid names are:

12years	cannot start with a digit
while	is a reserved keyword
my-name	cannot contain "-" characters

Variable Types

Variables must be of one of the following types:

real	floating point numbers (e.g. 12.345, 1.54e-7)
int	integer number between -2.147.483.648 and 2.147.483.647
short	integer number between -32768 and 32767
char	a single character (e.g. 'a', '*')
pointer	pointers are always opaque or NIL.
boolean	either TRUE or FALSE

Variable Declarations

Variables must be declared before they are first referenced in a statement.

Arrays are declared adding the number of elements enclosed in brackets ([]).

ProDeL arrays have only one dimension, with indexes starting at 0.

EXAMPLES

```
char    c;
char    string[256];    array of 256 characters
pointer dsetP;
int     i, n, Sum, len;
real    my_weight;
real    topTen[10];     array of 10 reals
```

Assignments

Use the "=" sign to change the value of a variable. Conversions are automatically performed where possible, though the results are machine dependent (e.g. how a real value is converted to integer).

EXAMPLES (see declarations in the previous page)

```
c = 'a';
string[3] = 'f';
dsetP = createDset(FALSE, len)
i = 9 + 4 * n ^ 2;
topTen[n] = topTen[n - 2] * topTen[n - 1];
```

NOTE: It is not possible to assign a value to all elements of an array in just one statement. Lines like the following will cause an error:

```
int         arr1[10], arr2[10], i;
char        string[256];
string = "hello world";        WRONG
arr1 = arr2;                   WRONG
```

Use this code instead:

```
strcpy(string, "hello world");
i = 0;
while(i < 10)
    arr1[i] = arr2[i]; i = i + 1;
endwhile;
```

Program Name Definition

Each ProDeL program must start with a line like the following one:

program <name> (<argumentList>);

<name> will be the new Xepr-Command to be entered on the command line to execute the ProDeL program.

<argumentList> is a list of variable declarations.

Executing the program will pop up a dialog box which will request the user to enter these values.

EXAMPLES

program my_ProDeL_program();

program create2D_Dataset(int width, height);

program addConstant(int handle; real constant);

IF Statement

The "if" statement allow omission of a sequence of statements or to choose between two code sequences under a specified condition.

SYNTAX

```
if(<boolean expression>) statement1; statement2; ...
endif;
```

or

```
if(<boolean expression>) statement1; statement2; ...
else
    statement1; statement2; ...
endif;
```

<boolean expression> must evaluate to a boolean value, i.e. either TRUE or FALSE

EXAMPLE: set x to its absolute value

```
if(x < 0) x = -x;
endif;
```

WHILE Statement

The "while" statement repeats a number of statements as long as an expression is TRUE.

SYNTAX

```
while(<boolean expression>) statement1; statement2; ...
endwhile;
```

<boolean expression> must evaluate to a boolean value, i.e. either TRUE or FALSE

EXAMPLE: initialize all elements of an array to 0

```
int     arr[10], i;
i = 0;
while(i < 10) arr[i] = 0; i = i + 1;
endwhile;
```

GOTO and LABEL Statement

The "label" statement allows you to mark any line in the code, and the "goto" statement allows to jump to such a labeled statement and resume execution from there.

SYNTAX

```
label <name>;
goto <name>;
```

EXAMPLE

```
if(x < 0) goto end;
endif;
...
...
label end;
```

RETURN Statement

The return statement ends the program, telling Xepr whether the program was successfully executed or an error occurred. In the former case, an entry is added to the Xepr-History. Omitting the return statement returns TRUE after the last sentence in the program.

EXAMPLE: accept only positive length

```
if(length < 0) printLn("length must be >= 0"); return(FALSE);
endif;
...
...
return(TRUE);
```

Comments

Comments can be inserted anywhere between symbols, and are delimited by hash signs ("#"). The same is true for spaces, tabs and new line characters. Symbols (i.e. keywords, variable names, numbers, special key sequences like ">=", "==", etc.) cannot contain comments nor spaces.

EXAMPLE

```
if(x < 0)     # if x is negative   # x = -x;# change it to positive #
endif;        # now x is surely positive #
```

EBNF-Notation

The following is the ProDeL-Syntax described using the EBNF (Extended Backus Naur Form) notation.

The special characters mean:

"..."	write the symbol exactly as it is here (without the double quotes)
[...]	optional part, may be omitted or included once
{...}	optional part, may be omitted or included several times.
|	or

Note that the expressions ("expr" production) are simplified, and the operator precedence is implicitly given by the line in the "operator" production.

```
prodelPrg = headLine varList stmtList.
headLine = "program" ident "(" argList ")" ";".
argList = { argDef }.
argDef = argType ident { "," ident } ";".
argType = "real" | "int" | "short" | "char" |   "boolean".
varList = { varDef }.
varDef = varType designatorDef {"," designatorDef}";".
varType = argType | "pointer".
stmtList = { [stmt] ";" }.
stmt = ifStmt | whileStmt | labelStmt | gotoStmt |  returnStmt | fctCall | assign.
ifStmt = "if" "(" expr ")" stmtList   [ "else" stmtList ] "endif".
whileStmt = "while" "(" expr ")" stmtList "endwhile".
labelStmt = "label" ident.
gotoStmt = "goto" ident.
returnStmt = "return" "(" expr ")".
fctCall = ident [ "(" exprList ")" ].
exprList = expr { "," expr }.
assign = ident "=" expr;
expr = factor { operator factor }.
factor = designator | number | fctCall | "(" expr ")".
designator = ident [ "[" expr "]" ].
designatorDef = ident [ "[" cardNr "]" ].
ident = letter { letter | digit | "_" }.
```

number = cardNr ["." cardNr] ["e" ["+" | "-"] cardNr].

cardNr = digir { digit }.

operator = "^" | "*" | "/" | "div" | "mod" |

"+" | "-" | "<" | "<=" | ">=" | ">" | "==" | "<>" | "and" | "or".

comment = "#" anything "#".

All spaces, tabs, new line characters and comments

between symbols are ignored.

ProDeL Acquisition Library

Constants:

- For experiment states:
    - AQ_EXP_CLOSED
    - AQ_EXP_EDIT
    - AQ_EXP_INSTALLED
    - AQ_EXP_ACTIVE
    - AQ_EXP_RUNNING
    - AQ_EXP_PAUSED

- For sample aggregation states:
    - AQ_SPL_CRYSTAL
    - AQ_SPL_POWDER
    - AQ_SPL_AMORPHOUS
    - AQ_SPL_LIQUID
    - AQ_SPL_GASEOUS

- For functional unit data type:
    - AQ_DT_UNKNOWN
    - AQ_DT_NUMERIC
    - AQ_DT_COMPLEX
    - AQ_DT_BOOLEAN
    - AQ_DT_ENUM
    - AQ_DT_STRING

Functions:

aqExpBuild

aqGetExpForDs

aqGetSelectedExp

aqSetSelectedExp

aqGetExpByHandle

aqGetExpByName

aqGetExpName

aqGetExpDataset

aqExpSync

aqMbcFineTune

aqMbcStandby

aqMbcOperate

aqGetExpState

aqExpEdit

aqExpInstall

aqExpActivate

aqExpRun

aqExpAbort

aqExpStop

aqExpPause

aqExpRunAndWait

aqGetComment

aqGetSplFormula

aqGetSplAggrState

aqSetComment

aqSetSplName

aqSetSplFormula

aqSetSplAggrState

aqGetParType

aqGetParNbDim

aqGetParDimSize

aqIsParLocked

aqGetParLabel

aqGetParUnits

aqGetParMinValue

aqGetParMaxValue

aqGetParFineSteps

aqGetParCoarseSteps

aqGetBoolParValue

aqGetIntParValue

aqGetRealParValue

aqGetStrParValue

aqSetBoolParValue

aqSetIntParValue

aqSetRealParValue

aqSetStrParValue

aqStepParValue

pointer aqExpBuild (pointer name, pointer type, pointer axs1, pointer axs2, pointer oxs1, boolean addTheslameter, boolean addGoniometer, boolean addVtu)

Builds an experiment for the above arguments. "name" will be the name of the new experiment. type, axs1, axs2, oxs1 are string arguments and any valid Bessst keyword for these SPL entries can be supplied. In addition, the theslameter, the goniometer and/or the v.t.u. can be added as static devices.

The return value can be used to refer the experiment when using the functions listed below (referred below as the exp argument of the functions).

pointer aqGetExpForDs(int handle)

Returns a reference to an experiment that can be used to acquire new datasets with the configuration used to generate the dataset supplied as an argument.

The return value can be used to refer the experiment when using the functions listed below (referred below as the exp argument of the functions).

pointer aqGetSelectedExp(int viewport)

An experiment can be linked to each viewport. This function will return a pointer to the current experiment of the selected viewport. 0 is returned if the operation fails.

The return value can be used to refer the experiment when using the functions listed below (referred below as the exp argument of the functions).

void aqSetSelectedExp(int viewport, pointer name)

Assigns the desired experiment for the selected viewport.

pointer aqGetExpByHandle(int number)

pointer aqGetExpByName(pointer name)

Return a pointer to the experiment. The experiment can be searched via its number or via its name. The experiment numbers are listed in the first column of the experiment table. 0 is returned if the operation fails.

The return value can be used to refer the experiment when using the functions listed below (referred below as the exp argument of the functions).

pointer aqGetExpName(pointer exp)
Return the name of an experiment.

pointer aqGetExpDataset(pointer exp)
Return a pointer to the dataset used by the experiment during an acquisition. You should not modify the content of the dataset.

void aqExpSync(pointer exp)
Synchronize the experiment with the server.

void aqMbcFineTune(pointer exp)
Perform a fine tuning of the microwave bridge.

void aqMbcStandby(pointer exp)
Puts the microwave bridge in standby mode.

void aqMbcOperate(pointer exp)
Puts the microwave bridge in operate mode.

int aqGetExpState(pointer expName)
Return the current state of the experiment. The possible return values of this function are listed above and start with AQ_EXP_.

void aqExpEdit(pointer exp)

void aqExpInstall(pointer exp)

void aqExpActivate(pointer exp)

void aqExpRun(pointer exp)

void aqExpAbort(pointer exp)

void aqExpStop(pointer exp)

void aqExpPause(pointer exp)

void aqExpRunAndWait(pointer exp)

Change the experiment state. Except for the last function, all will start the operation and return immediately. The last one will start the acquisition and wait until it is done or fails.

void aqGetComment(pointer buffer, int bufferSize)

void aqGetSplName(pointer buffer, int bufferSize)

void aqGetSplFormula(pointer buffer, int bufferSize)

int aqGetSplAggrState()

void aqSetComment(pointer buffer)

void aqSetSplName(pointer buffer)

void aqSetSplFormula(pointer buffer)

void aqSetSplAggrState(int state)

Various functions to change or query sample information. The buffer argument is a pointer to a string and the bufferSize argument indicates the size of the string. The return value or the argument values to get or set the sample aggregation state are listed above and start with AQ_SPL_.

int aqGetParType(pointer exp, pointer parName)

Return the data type of a parameter. The parName argument is a pointer to a string giving the name of the desired parameter. The valid return values are listed above and start with AQ_DT_.

int aqGetParNbDim(pointer exp, pointer parName)

Return the number of dimensions of a parameter. The parName argument is a pointer to a string giving the name of the desired parameter.

int aqGetParDimSize(pointer exp, pointer parName, int dimension)

Return the dimension size of a parameter. The parName argument is a pointer to a string giving the name of the desired parameter. The dimension argument indicates the dimension for which the size is desired. It should be greater or equal 0 and less than the number of dimensions returned by the previous function.

boolean aqIsParLocked(pointer exp, pointer parName)

Returns 0 if the parameter is not currently locked. If a parameter is locked any request to set its value is denied. The parName argument is a pointer to a string giving the name of the desired parameter.

void aqGetParLabel(pointer exp, pointer parName, pointer buffer, int bufferSz)

void aqGetParUnits(pointer exp, pointer parName, pointer buffer, int bufferSz)

Return the parameter description label or the parameter units.

real aqGetParMinValue(pointer exp, pointer parName)

real aqGetParMaxValue(pointer exp, pointer parName)

real aqGetParFineSteps(pointer exp, pointer parName)

real aqGetParCoarseSteps(pointer exp, pointer parName)

Return useful information about a parameter.

boolean aqGetBoolParValue(pointer exp, pointer parName, int dimSz, pointer dim)

int aqGetIntParValue(pointer exp, pointer parName, int dimSz, pointer dim)

real aqGetRealParValue(pointer exp, pointer parName, int dimSz, pointer dim)

boolean aqGetStrParValue(pointer exp, pointer parName, int dimSz, pointer dim, pointer buffer, int bufferSize)

Get a parameter value. For array parameters, dimSz indicates the number of dimension indices supplied via the dim argument. The dim argument is a pointer to an array of int. For non array parameters, use 0 as an argument for dimSz and dim.

boolean aqSetBoolParValue(pointer exp, pointer parName, int dimSz, pointer dim, boolean value)

boolean aqSetIntParValue(pointer exp, pointer parName, int dimSz, pointer dim, int value)

boolean aqSetRealParValue(pointer exp, pointer parName, int dimSz, pointer dim, real value)

boolean aqSetStrParValue(pointer exp, pointer parName, int dimSz, pointer dim, pointer value)

Set a parameter value. For array parameters, dimSz indicates the number of dimension indices supplied via the dim argument. The dim argument is a pointer to an array of int. For non array parameters, use 0 as an argument for dimSz and dim. If no error occurred, 0 is returned.

boolean aqStepParValue(pointer exp, pointer parName, int dimSz, pointer dim, boolean fineSteps, int steps)

Step a parameter value. For array parameters, dimSz indicates the number of dimension indices supplied via the dim argument. The dim argument is a pointer to an array of int. For non array parameters, use 0 as an argument for dimSz and dim. If fineSteps is set to false fine steps will be performed otherwise coarse steps will be performed. steps indicates the number of steps to increment or decrement the parameter. If no error occurred, 0 is returned.

ProDeL Standard Library

Constants:

TRUE, FALSE, NIL, PI, EUL

Numeric Operators

^, *, /, div, mod, +, -

Boolean Operators

<, <=, >, >=, ==, <>, and, or, not()

Trigonometric Functions

sin, cos, tan, cot, asin, acos, atan, acot

Hyperbolic Functions

sinh, cosh, sech, cosech

Logarithmic and Exponential Functions

log, ln, exp

Miscellaneous Functions

sqrt, abs, neg, pow, round, trunc

Random Number Generation

seed, random

String Manipulation Functions

strcpy, strcat, strcmp, strlen

Output Functions

print, printLn

Numeric Operators

Numeric operators may be applied to numeric values, and they yield a numeric result. Numeric values are any variable or constant of type "real", "int", "short" or "char". For this purpose, characters are converted to their ascii value.

Boolean Operators

Boolean operators always yield a boolean result. Some may be applied only to numeric values, while others (and, or, not()) only make sense on boolean values.

Operator Precedence

Operators with higher precedence are evaluated first, while operators of same precedence are evaluated from left to right. Use parentheses to change the order of evaluation.

Operators in Order of Precedence

num means numeric value, bool means boolean value.

int means a numeric value of integer type (i.e. type "int", "short" or "char").

\- num = -num

Same as function neg(num)

^ num = num ^ num (Power)

2 ^ 3 (= 8)

Same as function pow(2, 3)

* / div mod

num = num * num

num = num / num

int = num div num (Integer Division)

7 div 3 (= 2)

int = num mod num (Modulo)

7 mod 3 (= 1)

+ - num = num + num
num = num - num

< <= > >= == <> bool = num op num
< less than
<= less than or equal to
> greater than
>= greater than or equal to
== equal to
<> different from

and or bool = bool and bool
bool = bool or bool
and gives TRUE if both are TRUE
or gives FALSE if both are FALSE

Note that the operator NOT is implemented as function and thus has highest precedence.

not bool = not(bool)
gives TRUE if argument is FALSE

Trigonometric Functions

All trigonometric functions assume that the angles are radians, and all return floating point values (i.e. of type "real").

PI	constant pi = 3.141592, as precise as possible on the given machine	

r = sin(a)	sine	
r = cos(a)	cosine	
r = tan(a)	tangent	(a <> (2n+1) * pi/2)
r = cot(a)	cotangent	(a <> n * pi)

The corresponding inverse functions are

a = asin(r)	arcsine	(r in [-1, 1])
a = acos(r)	arccosine	(r in [-1, 1])
a = atan(r)	arctangent	
a = acot(r)	arccotangent	

Assume r (result) and a (angle) are of type real.

Hyperbolic Functions

All trigonometric functions return floating point values (i.e. of type "real").

r = sinh(x)	hyperbolic sine
	r = (exp(x) - exp(-x)) / 2;
r = cosh(x)	hyperbolic cosine
	r = (exp(x) + exp(-x)) / 2;
r = sech(x)	hyperbolic secant
	r = 2 / (exp(x) + exp(-x));
r = cosech(x)	hyperbolic cosecant
	r = 2 / (exp(x) - exp(-x));

Assume r and x are of type real.

Logarithmic and Exponential Functions

All logarithmic and exponential functions return floating point values (i.e. of type "real").

EUL constant e = 2.718281, as precise as possible on the given machine

r = log(x) common logarithm with base 10
(x > 0)

r = ln(x) natural logarithm with base e
(x > 0)

r = exp(x) exponential
e raised to the power of x.

Assume r and x are of type real.

Miscellaneous Functions

r = sqrt(x) square root (x >= 0)

r = abs(x) absolute value

r = neg(x) negation, same as -x

r = pow(x, y) x raised to the power of y, same as r = x ^ y;

i = round(x) round the real value x to the
nearest integer value
.5 is rounded up
round(1.3) = 1
round(1.8) = 2
round(1.5) = 2

i = trunc(x) integer part of the real value x
trunc(1.3) = 1
trunc(1.8) = 1

Assume r, x and y are of type real and i is of type int

Random Number Generation

The function random generates pseudo-random numbers in the range [0..1]. The library maintains an internal state for random numbers that is modified each time you call the function. This state can be set explicitly by the function seed. From a given seed value, the sequence of generated numbers will always be the same. Please note that the spectral properties of the random number generator are very limited.

seed(i) initialize random number generator
r = random obtain next random number
Assume r is of type real and i is of type int

ProDeL Strings

ProDeL strings are declared as arrays of characters, and are terminated by a NULL character, i.e. by a character of (ascii) value 0.

EXAMPLE: declaration of two string of 100 characters.

```
char s1[100], s2[100];
```

To fill a string character by character, remember to terminate it.
EXAMPLE: set s1 to the string "hi"

```
s1[0] = 'h';
s1[1] = 'i';
s1[2] = 0;# make sure to terminate string #
```

the same is obtained using the function strcpy:

```
strcpy(s1, "hi");
```

String Manipulation Functions

These function assume null terminated strings. They do not check if the strings are large enough to contain the result (strcpy and strcat), or if they are correctly terminated. Numeric parameters are converted to their string representation, while boolean values are converted to the string "true" or "false".

strcpy(s1, s2);

copy string s2 to s1.
s1 is null terminated.

strcat(s1, s2);

append string s2 to s1.
s1 is null terminated.

i = strlen(s1);

counts the characters of s1 preceding the null terminating character.

i = strcmp(s1, s2);

compares string s1 and s2.
Returns an integer greater than, equal to, or less than 0 if s1 is lexicographically greater than, equal to or less than s2.

Round to Error Range

This function fills the user supplied strings valStr and errStr with the string representation of val and err rounded to the position of the first significant digit of err. valStr and errStr must be at least 16 characters long.

roundErr(val, err, valStr, errStr);

EXAMPLES

Suppose a computation returns 1234.5678 as result, with an error (or tolerance) of 12.34.
It would be more useful to output it as 1230 +- 10 than with all the insignificant digits.

```
char valStr[16], errStr[16];

roundErr(1234.5678, 12.34, valStr, errStr);
    valStr = "1230", errStr = "10";
```

roundErr(0.123, 0.0472, valStr, errStr);
valStr = "0.12", errStr = "0.05";
roundErr(0.178, 0.143, valStr, errStr);
valStr = "0.2", errStr = "0.1";
roundErr(478, 123, valStr, errStr);
valStr = "500", errStr = "100";
roundErr(478, 1230, valStr, errStr);
valStr = "0", errStr = "1000";
roundErr(578, 1230, valStr, errStr);
valStr = "1000", errStr = "1000";

Output Functions

ProDeL offers a very easy way to display data on the screen, with two very similar functions that cover the needs of virtually any program.

print(anyType, anyType, ...);
printLn(anyType, anyType, ...);
printLn;

print the arguments on the screen, in the specified order. The printLn command prints a new line character after the last argument. If no arguments are specified, the parentheses are not required (and not accepted).

Note that the print commands accept any type as argument, converting it to the proper screen representation. Boolean values are output as the strings "true" or "false", pointer arguments are treated as pointers to character arrays.

Formatted Output

The print commands support arguments containing format instructions enclosed in < >. Such format instructions can contain the following directives, in this order:

"L" or "R	"Left or Right justified. By default, numeric output is right justified, while strings, character and booleans are left justified.
"+" or "	"By default, positive numeric values are output without a sign. This setting defines the character to be printed as the positive sign, i.e. either "+" or a space.
width	Specifies the minimum field width (in characters) in which to print the result. If the converted argument has fewer characters than the field width, it will be padded on the right (or left, if right justified) with spaces to make up the field width. For floating point values, width specifies the field width up to the decimal point (with the decimal point not included), thus allowing to easily align the decimal points.
"."	Specifies that the argument has to be displayed as floating point, and separates the width from the precision field.
precision	Specifies the number of digits to be printed after the decimal point.
"E"	Forces exponential representation of the floating point argument.

The numeric fields "width" and "precision" can be specified either by decimal numbers or by a corresponding sequence of hash characters ("#"). Uppercase and lowercase letters can be used for the justification and exponential directives.

Note that the sign visibility, decimal period, precision and exponential representation directives do not make sense when applied to strings or boolean values. Doing so may lead to a runtime error of the ProDeL interpreter.

Syntax:
<["L"|"R"]["+"|" "][width]["."[precision]]["E"]>

EXAMPLES	output
print("<8>", "hello");	"hello "
print("<########>", "hello");	"hello "
print("<R8>", "hello");	" hello"
print("<####>", 12);	" 12"
print("<L+####>", 12);	"+12 "
print("val = <L+4>", 12);	"val = +12 "
print("<###.###>", 12.3456);	" 12.346"
print("<###.###E>", 12.3456);	" 1.235e+02"
print("<3.4E>", 12.345621);	" 1.2346e+02"
print("<3E>", 12.345621);	" 1e+02"

This example shows the output of a sequence of three separate print statements.

EXAMPLE

```
printLn("<##> + <##> = <###>", 2, 7, 2 + 7);
printLn("<##> + <##> = <###>", 4, 25, 4 + 25);
printLn("<##> + <##> = <###>", 84, 22, 84 + 22);
```

OUTPUT

```
" 2 +  7 =   9"
" 4 + 25 =  29"
"84 + 22 = 106"
```

Note that the print command does not output the double quote characters. These are only used in these examples to show the space characters that are used to format the output.

ProDeL Xepr Library

Constants:

X_ABSC, Y_ABSC, REAL_ORD, IMAG_ORD

IDX, IGD, NTUP

AREA, POINT, POSITION, REGION, UNCHANGED

Dataset fetching:

pointer getCopyByHandle

pointer getCopyOfPrimary

pointer getCopyOfSecondary

pointer getCopyOfQualifier

pointer getCopyOfResult

int getHandleOfPrimary

int getHandleOfSecondary

int getHandleOfQualifier

pointer getCopyOfSlice (pointer dsetPtr, int index, X_ABSC or Y_ABSC)

Dataset setting:

copyDsetToPrimary (pointer dsetPtr)

copyDsetToSecondary (pointer dsetPtr)

copyDsetToQualifier (pointer dsetPtr, POINT or POSITION or REGION or AREA or UNCHANGED)

copyDsetToResult (pointer dsetPtr)

int storeCopyOfDset (pointer dsetPtr)

Dataset creation and destruction:

pointer createDset (boolean isCplx, int len)

pointer create2DDset (boolean isCplx, int lenAbsc1, int lenAbsc2)

destroyDset (pointer dsetPtr)

Dataset query:

boolean isComplex (pointer dsetP)

int getNrOfPoints (pointer dsetP, X_ABSC or Y_ABSC or REAL_ORD, IMAG_ORD)

int getDimension (pointer dsetP)

int getAbscType (pointer dsetP, X_ABSC or Y_ABSC)

setAbscType (pointer dsetP, X_ABSC or Y_ABSC, IDX or IGD or NTUP)

real getMin (pointer dsetP, X_ABSC or Y_ABSC or REAL_ORD or IMAG_ORD)

real getMax (pointer dsetP, X_ABSC or Y_ABSC or REAL_ORDor IMAG_ORD)

real getSPLReal (pointer dsetP, pointer keyword)

getTitle (pointer dsetP, pointer buffer)

setTitle (pointer dsetP, pointer title)

Dataset modification:

real getValue (pointer dsetP, int idx, X_ABSC or Y_ABSC or REAL_ORD or IMAG_ORD)

setValue (pointer dsetP, int idx, X_ABSC or Y_ABSC or REAL_ORD or IMAG_ORD, real val)

real get2DValue (pointer dsetP, int idxAbsc1, int idxAbsc2, REAL_ORD or IMAG_ORD)

set2DValue (pointer dsetP, int idxAbsc1, int idxAbsc2, REAL_ORD or IMAG_ORD, real val)

fillAbscissa (pointer dsetP, X_ABSC or Y_ABSC, real firstVal, lastVal)

appendPoint (pointer dsetP)

insertPoint (pointer dsetP, int idx)

removePoint (pointer dsetP, int idx)

Miscellaneous functions:

execCmd (string, anyType, anyType, ...)

workIndex (real percentage)

sleep (real seconds)

Error handling:

halt(int statusCode or LAST_ERROR_CODE)

onError(TERMINATE or SET_STATUS)

boolean fctFailed()

int getStatusCode()

Fetching Datasets

The getCopyOf* functions return a pointer to a copy of the specified dataset. This means that modifying a dataset in the ProDeL program does not affect the original dataset in the Xepr program.

pointer dsetP; # declare variable dsetP as pointer #

dsetP = getCopyOfPrimary;

dsetP = getCopyOfSecondary;

dsetP = getCopyOfQualifier;

dsetP = getCopyOfResult;

dsetP = getCopyByHandle(int handle);

Returns a copy of the managed dataset with the specified handle (handle is an integer value).

sliceP = getCopyOfSlice(dsetP, index, X_ABSC or Y_ABSC)

Returns the slice number <index> (integer value, slice numbers start at 0) of the dataset dsetP. The slice is extracted either along abscissa X_ABSC or Y_ABSC. The resulting sliceP is a normal one-dimensional dataset.

Datasets can be destroyed (freeing their memory) using the destroyDset function.

Query Dataset Handle Functions

The getHandleOf* functions return the handle of the specified dataset. If the dataset is not managed, i.e. if it is a local dataset, 0 is returned.

handle = getHandleOfPrimary;

handle = getHandleOfSecondary;

handle = getHandleOfQualifier;

Assume that handle is declared as int.

Creating Datasets

The following functions return a pointer to a local dataset.

dsetP = createDset(isComplex, len)

Creates a 1D dataset of <len> number of points. The data points are complex if the boolean flag <isComplex> is set to TRUE.

dsetP = create2DDset(isComplex, lenXAbsc, lenYAbsc)

Creates a 2D dataset of <lenXAbsc> number of points in the X abscissa, and <lenYAbsc> number of points in the Y abscissa. The data points are complex if the boolean flag <isComplex> is set to TRUE.

Assume the following declarations:

pointer dsetP;

int len, lenXAbsc, lenYAbsc;

boolean isComplex;

Setting Xepr Datasets

The following functions copy a local dataset to the Xepr application, and display it as the specified dataset.

copyDsetToPrimary(dsetP)

copyDsetToSecondary(dsetP)

copyDsetToResult(dsetP)

copyDsetToQualifier(dsetP, qualiType)

Works on 1D datasets only.

<qualiType> specifies the type of the qualifier, and thus how to interpret its data points. Valid values are:

POINT:	each dataset point is a qualifier point
POSITION:	each dataset point is a qualifier point, and only the abscissa value is meaningful.
REGION:	the dataset points are considered as pairs, and only their abscissa value is meaningful.
AREA:	the dataset points are considered as pairs, where the first point specifies the lower left corner and the second point the upper right corner of an area
UNCHANGED:	keeps the actual qualifier type

Storing Datasets

handle = storeCopyOfDset(dsetP)

Stores the local dataset and returns its handle. Note that this function does not save the dataset to disc. It merely makes it a managed dataset, adding an entry in the dataset selection menu of the main panel, and in the dataset directory panel.

Assume that dsetP is declared as pointer and handle is declared as int.

Destroying Datasets

The memory allocated by a local dataset is automatically released upon termination of the ProDeL program. However, it is advisable to destroy local datasets as soon as they are no longer needed.

destroyDset(dsetP)

Releases the memory used by the dataset.

Accessing Dataset Points

In the following functions and procedures, the parameter <axisCst> refers to the desired axis, and can be one of X_ABSC, Y_ABSC, REAL_ORD, IMAG_ORD.

Notice, however, that not all axis constants can be applied to every dataset. Y_ABSC is only valid on 2D datasets, IMAG_ORD only on complex datasets.

Point indices always start at zero.

r = getValue(dsetP, index, axisCst)

Returns the value of the point at <index> of the specified axis.

Returns 0 (zero) if the specified axis is not available for this dataset.

setValue(dsetP, index, axisCst, value)

Sets the point at <index> of the specified axis to <value>.

r = get2DValue(dsetP, xAbscIdx, yAbscIdx, ordCst)

set2DValue(dsetP, xAbscIdx, yAbscIdx, ordCst, value)

These are convenience functions to easily access ordinate values of 2D datasets.

<ordCst> can only be either REAL_ORD or IMAG_ORG.

Assume the following declarations:

pointer dsetP;

real r, value;

int index, xAbscIdx, yAbscIdx, axisCst, ordCst;

Fill Abscissa

The createDset or create2DDset functions create a dataset of the specified length, but leave the points uninitialized, i.e. their value is undefined. This is a convenience function to fill abscissas with equidistant values.

fillAbscissa(dsetP, abscissaCst, firstValue, lastValue)

Fills the abscissa with equidistant values between <firstValue> and <lastValue>. <abscissaCst> may be either X_ABSC or Y_ABSC, where Y_ABSC is only suitable for 2D datasets.

Assume the following declarations:

pointer dsetP;

real firstValue, lastValue;

int abscissaCst;

Dataset Manipulation

These functions change the size of a 1D dataset. They cannot be applied to 2D datasets.

appendPoint(dsetP)

Appends a point to the end of the dataset. The point is not initialized, so its value is undefined.

insertPoint(dsetP, index)

Inserts a point into the dataset at position <index>. Indexes start at 0.

The point is not initialized.

removePoint(dsetP, index)

Removes the point at position <index> from the dataset. Indexes start at 0.

Assume that dsetP is declared as pointer and index as int.

Dataset Attributes

b = isComplex(dsetP)

Returns TRUE if the dataset is complex.

dim = getDimension(dsetP)

Returns the dataset dimension (1 for 1D and 2 for 2D datasets).

n = getNrOfPoints(dsetP, axisCst)

Returns the number of points in the specified axis. <axisCst> can be one of X_ABSC, Y_ABSC, REAL_ORD, IMAG_ORD.

r = getMin(dsetP, axisCst)

r = getMax(dsetP, axisCst)

Return the smallest and largest value of the specified axis, respectively.
<axisCst> can be one of X_ABSC, Y_ABSC, REAL_ORD, IMAG_ORD.

r = getSPLReal(dsetP, keyword)

Returns the value for <keyword> from the Standard Parameter Layer as a real number.

Assume the following declarations:

```
pointer dsetP, keyword;
boolean b;
int dim, n, axisCst;
real r;
```

Dataset Title

These functions query and set the dataset's title.

getTitle(dsetP, title)

setTitle(dsetP, title)

Note that setTitle accepts any kind of strings, i.e. either an array of characters or a constant string enclosed in double quotes.

For getTitle, the argument must be a character array at least 65 characters long to hold the read title.

Assume the following declarations:

pointer dsetP;

char title[65];

EXAMPLES

```
getTitle(dsetP, title);          # that's ok #
getTitle(dsetP, "hi there");     # WRONG #
setTitle(dsetP, title);          # that's ok #
setTitle(dsetP, "hi there");     # that's ok #
```

Dataset Abscissa Type

The abscissas of a dataset can be of different type.

These functions set and get the abscissa type.

abscTypeCst = getAbscType(dsetP, abscissaCst)

setAbscType(dsetP, abscissaCst, abscTypeCst)

<abscissaCst> can be either X_ABSC or Y_ABSC, where Y_ABSC only makes sense on 2D datasets.

<abscTypeCst> can be one of:

IDX: (Indexed) All points in the abscissa are equidistant and monotonous.

IGD: (Indexed-Gauged) Abscissa points need not be equidistant.

NTUP: (n-Tuple) Every data point is given as an n-Tuple of coordinate values, like (2, 7, 10). While there is no difference between index-gauged and n-Tuple in 1D-datasets, a 2D-dataset with index-gauged abscissas consists of one slice per abscissa value, while for an 2D-n-Tuple dataset, the term slice is not applicable since there is not necessarily more than one point at any abscissa position.

Note that converting a dataset to n-Tuple type is irreversible and increases the size of the dataset if it is 2D or of higher dimension.

Assume the following declarations:
pointer dsetP;
int abscTypeCst, abscissaCst;

Execution of an Xepr Command

A ProDeL program can execute all Xepr commands, just by passing the string to the command interpreter, using the same syntax as if the command were typed on the command line.

execCmd(cmdString, arg1, arg2, ...)

The first argument is supposed to contain the command name, and the following arguments can be of any type, and are converted to strings and added to the command string. Arguments are automatically separated by a space. The resulting command string is passed to the Xepr Command Interpreter for execution.

EXAMPLES
execCmd("prDeriv");
execCmd("ddLoad", fileName);

Work Index

This procedure sets the displayed percentage of the growing bar in the status line, to indicate the completion status of the program.

workIndex(percentage)

Assume that percentage is of type real.

Sleep

Suspend the execution of the program for a given amount of seconds.

sleep(seconds)

Assume that seconds is of type real.

Error Handling

By default, the ProDeL interpreter aborts a program as soon as an error is detected. These functions allow to have more control over the program, taking corrective actions or printing a detailed error message instead of simply aborting the program.

onError(TERMINATE)

Default behavior, aborts program when an error is detected.

onError(SET_STATUS)

When an error occurs, a status is set and the program continues normally.

boolean fctFailed()

Checks if the last called library function reported an error.

int getStatusCode()

Returns the status code of the last call to a library function.

See all status codes in the next pages.

halt(status code or LAST_ERROR_CODE)

Aborts the program with a message corresponding either to the status code passed as argument, or to the status code of the last called library function.

Error Handling Example

The following program wants to get either the primary or, if it does not exist, the secondary dataset. Asking for the primary dataset when there is none usually leads to an error (status code number 15, 'error getting dataset'), so we must handle this error condition.

```
program foo();
pointer dsetP;

onError(SET_STATUS);       # we handle error cases #
dsetP = getCopyOfPrimary;  # get primary dataset #
if(fctFailed())
if(getStatusCode != 15)                  # not the error we want #
    halt(LAST_ERROR_CODE);       # abort with error message from the system #
```

```
endif;

dsetP = getCopyOfSecondary;              # try getting secondary #
endif;

if(dsetP == NIL)                         # still no dataset #
        printLn("Could not get dataset"); # error message #
        return(FALSE);                   # stop program #
endif;
onError(TERMINATE);                      # continue with normal error handling #
....
....
```

ProDeL Status Codes

code	meaning
0	OK
1	Cmd interpreter failed
2	Invalid handle
3	Invalid viewport number
4	Invalid dimension
5	Invalid index
6	Invalid slice number
7	Invalid length
8	Invalid axis
9	Invalid axis type
10	Invalid title
11	Invalid qualifier type
12	Invalid dataset (null pointer)
13	Invalid command (null pointer)
14	Error creating dataset
15	Error getting dataset
16	Error copying dataset
17	Error destroying dataset
18	Error setting dataset
19	Error getting dataset slice
20	Error storing dataset
21	Error filling abscissa
22	Error appending point
23	Error inserting point
24	Error removing point
25	Error setting axis type
26	Error getting title
27	Error setting title
28	SPL keyword not found

ProDeL Examples

proc1Dskeleton
Skeleton of a ProDeL program that processes a 1D dataset.

proc2Dskeleton
Skeleton of a ProDeL program that processes a 2D dataset.

fitPeaks
Creates a dataset containing the sum of the fits of derived lorentz curves through the peaks of the (qualified) primary. (Works best with the vo_100 example dataset).

mexHat
Creates a 2D dataset with a "mexican hat".

norm01
Transforms the primary dataset to a range of 0 - 1.

norm11
Transforms the primary dataset to a range of 0 - 1 if the base line is in the lower part of the trace (like a gauss curve), and to a range of -1 - 1 if there are troughs in the lower part (like a derived gauss).

quick2D
Reads in a sequence of 1D files by Number and builds a 2D dataset out them. The filename of the 1D files must look like 'name?', where ? stands for the number.

shift2D
Skews a 2D dataset by shifting its slices proportional to the slice number.

acqDemo
This program demonstrates the acquisition support from within a ProDeL program. The experiment selected for the current viewport will be used.

acqManipDemo
This ProDeL program demonstrates the usage of the acquisition and manipulation support. The program will perform a field sweep experiment in viewport 1, extract the peaks and perform a time sweep experiment for each peak in viewport 2. Each acquisition result is stored in a dataset.

acqStepParDemo

This ProDeL program demonstrates the usage of the acquisition support. The program is running in the background, i.e. is triggered by the end of an acquisition. It will step a predefined parameter value and restart the acquisition. To work properly, this program must be compiled and defined as acquisition post process command to be executed.

This example can be used to perform 3D, 4D, etc. acquisitions using the basic 1D and 2D acquisitions we currently support. The result will be a series of datasets for the additionally swept parameter.

proc1Dskeleton

```
# Skeleton of a ProDeL program that processes a 1D dataset. #
# Important: set the program's name. #

program _Program_Name_ ();

pointer     dsetP;
int         idx, nrOfPoints;
real        x, y;

dsetP = getCopyOfPrimary;
nrOfPoints = getNrOfPoints(dsetP, X_ABSC);

idx = 0;
while(idx < nrOfPoints)        # loop over each point #
    x = getValue(dsetP, idx, X_ABSC);
    y = getValue(dsetP, idx, REAL_ORD);
    # ------------------------------------------------------ #
    # COMPUTE NEW VALUE: y = f(x, y); #
    y = y;
    # ------------------------------------------------------ #
    setValue(dsetP, idx, REAL_ORD, y);
    idx = idx + 1;
endwhile;

copyDsetToResult(dsetP);
```

proc2Dskeleton

```
# Skeleton of a ProDeL program that processes a 2D dataset. #
# Important: set the program's name. #

program _Program_Name_ ();

pointer     dsetP;
int         xIdx, yIdx, nrOfSlices, nrOfPoints;
real        x, y, z;

dsetP = getCopyOfPrimary;
nrOfPoints = getNrOfPoints(dsetP, X_ABSC);
nrOfSlices = getNrOfPoints(dsetP, Y_ABSC);
yIdx = 0;
while(yIdx < nrOfSlices)# loop over each slice #
        y = getValue(dsetP, yIdx, Y_ABSC);
        xIdx = 0;
        while(xIdx < nrOfPoints)# loop over each point in slice #
                x = getValue(dsetP, xIdx, X_ABSC);
                z = get2DValue(dsetP, xIdx, yIdx, REAL_ORD);
                # ------------------------------------------------- #
                # COMPUTE NEW VALUE: z = f(x, y, z); #
                z = z;
                # ------------------------------------------------- #
                set2DValue(dsetP, xIdx, yIdx, REAL_ORD, z);
                xIdx = xIdx + 1;
        endwhile;
        yIdx = yIdx + 1;
        workIndex(100 * yIdx / nrOfSlices);
endwhile;
copyDsetToResult(dsetP);
```

fitPeaks

```
# Creates a dataset containing the sum of the fits of derived lorentz curves through the peaks of the #
# (qualified) primary. #

program fitPeaks();

pointer     dsetP, peakP, sumP;
int         len, i;
real        x1,x2, y1, y2;

# Get copy of primary dataset and check that it is a 1D dataset #
dsetP = getCopyOfPrimary;
if(getDimension(dsetP) <> 1)
    printLn("Can process 1D datasets only");
    return(FALSE);
endif;

# Pick both peaks and troughs of the primary. #
# Picks peaks only on the qualified points of the primary dset. #
execCmd("prPeakPick -1 Primary both win");
peakP = getCopyOfResult;

# Create a dataset with same abscissa as the primary, but all ordinates set to zero, used to sum up #
# the single fit curves. Turn off any active qualifier. #
execCmd("vpCurrent -1 Qualifier 0");
execCmd("prCstOperation -1 Primary -1 * 0.0");
sumP = getCopyOfResult;

# Loop over all peaks. Suppose they are arranged in pairs #
# (peak/trough of each derived lorentz curve). #
len = getNrOfPoints(peakP, X_ABSC) div 2;
i = 0;
while(i < len)
```

```
        # Get the first two points (peak and trough) and remove them from the list of peaks #
        # (i.e. the <peakP> dataset). #
        x1 = getValue(peakP, 0, X_ABSC);
        y1 = getValue(peakP, 0, REAL_ORD);
        x2 = getValue(peakP, 1, X_ABSC);
        y2 = getValue(peakP, 1, REAL_ORD);
        removePoint(peakP, 1);      # remove in reverse order #
        removePoint(peakP, 0);
        # Fit a derived lorentz curve through these two points. #
        execCmd("prFitDLorentz -1 Primary -1",
                    abs(y1 - y2), "yes",            # amplitude #
                    x1 + (x2 - x1) / 2.0, "yes",    # x offset #
                    abs(x2 - x1), "yes");           # width #
        # Add the fitted curve to the <sumP> dataset. #
        execCmd("vpRToSec -1 -1");
        copyDsetToPrimary(sumP);
        execCmd("prSum -1 Primary -1 1.0 0.0 1.0");
        destroyDset(sumP);
        sumP = getCopyOfResult;
        # Prepare for next peak/trough pair. #
        copyDsetToPrimary(dsetP);
        i = i + 1;
        workIndex(100.0 * i / len);
endwhile;
# Display the <sumP> dataset as the result trace, and remove the secondary trace. #
execCmd("vpCurrent -1 Secondary 0");
copyDsetToResult(sumP);

# Free allocated memory and inform the interpreter that everything went right. This part is optional. #
destroyDset(sumP);
destroyDset(peakP);
destroyDset(dsetP);
return(TRUE);
```

mexHat

```
# creates a 2D dataset with a "mexican hat". #
# The dset is <height> slices, each containing <width> points. <waves> specifies how many sine periods #
# are computed. The new dataset is displayed as the RESULT trace. #

program mexHat(int width, height, waves);

pointer     dsetP;
int         xIdx, yIdx;
real        x, y, r, percentDone;

if(waves <= 0)
    printLn("Warning: Invalid number of waves (<#>). Changed to 1", waves);
    waves = 1;
endif;

# create dataset and fill abscissa with equidistant values #
dsetP = create2DDset(FALSE, width, height);
fillAbscissa(dsetP, X_ABSC, -PI * waves, PI * waves);
fillAbscissa(dsetP, Y_ABSC, -PI * waves, PI * waves);

yIdx = 0;                   # loop for each slice #
while (yIdx < height)
    y = getValue(dsetP, yIdx, Y_ABSC);
    xIdx = 0;
    while(xIdx < width)     # loop for each point in slice #
        x = getValue(dsetP, xIdx, X_ABSC);
        r = sqrt(x * x + y * y);
        if(r <> 0)
            r = sin(r) / r;
        endif;
        set2DValue(dsetP, xIdx, yIdx, REAL_ORD, r);
        xIdx = xIdx + 1;
```

```
        endwhile;
        yIdx = yIdx + 1;
        percentDone = 100 * yIdx / height;
        workIndex(percentDone);      # display amount of work done #
endwhile;

setTitle(dsetP, "Mexican Hat");
copyDsetToResult(dsetP);     # set new dataset into result #
destroyDset(dsetP);          # destroy local dataset #

return(TRUE);            # END #
```

norm01

```
# Transforms the primary dataset to a range of 0 - 1 #

program norm01();

pointer     primP, resP;
int         primHdl;
real        minX, minY, width, height, x0, y0;

# Get the primary dataset and check that it is a 1D dataset. #
primHdl = getHandleOfPrimary;
primP = getCopyOfPrimary;
if(getDimension(primP) <> 1)
    printLn("Can normalize 1D datasets only");
    destroyDset(primP);
    return(FALSE);
endif;

# Get the range of the primary dataset and its first point. #
minX = getMin(primP, X_ABSC);
minY = getMin(primP, REAL_ORD);
width = getMax(primP, X_ABSC) - minX;
height = getMax(primP, REAL_ORD) - minY;
x0 = getValue(primP, 0, X_ABSC);
y0 = getValue(primP, 0, REAL_ORD);

# Transform the primary dataset abscissa and ordinate range to (0,1) and copy it to the result. #
execCmd("vpTransf -1 Primary first",
    -(x0 + (minX - x0) / width),  # x shift #
    1.0 / width,                  # x factor #
    -(y0 + (minY - y0) / height), # offset #
    1.0 / height);                # gain #
resP = getCopyOfPrimary;
```

```
copyDsetToResult(resP);

# Restore Primary dataset. #
if(primHdl == 0)
        copyDsetToPrimary(primP);
else
        execCmd("vpCurrent -1 Primary", primHdl);
endif;

# Free the allocated memory and return successfully. #
destroyDset(primP);
destroyDset(resP);
return(TRUE);
```

norm11

```
# Transforms the primary dataset to a range of 0 - 1 if the base line is in the lower part of the trace (like #
# a gauss curve), and to a range of -1 - 1 if there are troughs in the lower part (like a derived gauss). #

program norm11();

pointer     primP, resP;
int         primHdl, idx, len, lowCounter, midCounter;
real        minX, minY, width, height, x0, y0;
real        highLimit, lowLimit, y, lowRange;

# Get the primary dataset and check that it is a 1D dataset. #
primHdl = getHandleOfPrimary;
primP = getCopyOfPrimary;
if(getDimension(primP) <> 1)
    printLn("Can normalize 1D datasets only");
    destroyDset(primP);
    return(FALSE);
endif;

# Get the range of the primary dataset and its first point. #
minX = getMin(primP, X_ABSC);
minY = getMin(primP, REAL_ORD);
width = getMax(primP, X_ABSC) - minX;
height = getMax(primP, REAL_ORD) - minY;
x0 = getValue(primP, 0, X_ABSC);
y0 = getValue(primP, 0, REAL_ORD);

# Count how many points are in the lower third and how many are in the middle third of the dataset. #
lowLimit = minY + height / 3.0;
highLimit = lowLimit + height / 3.0;

idx = 0;
```

```
lowCounter = 0;
midCounter = 0;
len = getNrOfPoints(primP, REAL_ORD);
while(idx < len)
        y = getValue(primP, idx, REAL_ORD);
        if(y < lowLimit)
                lowCounter = lowCounter + 1;
        else
                if(y < highLimit)
                        midCounter = midCounter + 1;
                endif;
        endif;
        idx = idx + 1;
endwhile;

# Transform the abscissa range of the primary dataset to (0,1). Set the ordinate range to (-1, 1) if there #
# are more points in the middle third than in the lower third (this means that in the lower third there are #
# troughs, like in a derived gauss). Set the range to (0, 1) otherwise. Copy the transformed dataset to the #
# result. #
if(lowCounter < midCounter)
        height = height / 2.0;
        lowRange = -1.0;
else
        lowRange = 0.0;
endif;

execCmd("vpTransf -1 Primary first",
        -(x0 + (minX - x0) / width),   # x shift #
        1.0 / width,                   # x factor #
        lowRange - (y0 + (minY - y0) / height), # offset #
        1.0 / height);                 # gain #
resP = getCopyOfPrimary;
copyDsetToResult(resP);
```

```
# Restore Primary dataset. #
if(primHdl == 0)
        copyDsetToPrimary(primP);
else
        execCmd("vpCurrent -1 Primary", primHdl);
endif;

# Free the allocated memory and return successfully. #
destroyDset(primP);
return(TRUE);
```

quick2D

```
# Reads in a sequence of 1D files by Number and builds a 2D dataset out them. The filename of #
# the 1D files must look like 'name?', where ? stands for the number. #

program Quick2D(pointer path_filename; int firstNr, lastNr; boolean ESP_format);

char    cmdString[220];
char    extension[8];
int     i, firstHandle;

onError(TERMINATE);     # abort program on error (that's the default) #

# Set appropriate extension for datasets to be loaded #
if(ESP_format)
        strcpy(extension, ".par");
else
        strcpy(extension, ".dsc");
endif;

# Ensure that we are allowed to remove datasets #
execCmd("ddSecurity all attribute confirm no no");

# Load the first file and make it the Primary dataset #
strcpy(cmdString, "ddLoad ");
strcat(cmdString, path_filename);
strcat(cmdString, firstNr);
strcat(cmdString, extension);
execCmd(cmdString);                     # Load first dataset #
execCmd("vpCurrent -1 Primary -1");     # Display it as Primary #
firstHandle = getHandleOfPrimary;       # Remember its handle #

# Load each slice, display it as Secondary, attach it to the previous #
# slices, and make the result the primary dataset. #
```

```
i = firstNr+1;
while(i <= lastNr)
        strcpy(cmdString, "ddLoad ");
        strcat(cmdString, path_filename);
        strcat(cmdString, i);
        strcat(cmdString, extension);
        execCmd(cmdString);                         # Load dataset #
        execCmd("vpCurrent -1 Secondary -1");       # Current -> Secondary #
        execCmd("prBuild2D -1 angle");              # Combine to 2D #
        execCmd("vpRToPrim -1 -1");                 # Result -> Primary #
        execCmd("ddRemove -1");                     # Remove used dataset #
                                                    # Display work done #
        workIndex(100 * (i - firstNr) / (lastNr - firstNr));
        i = i+1;
endwhile;

execCmd("ddRemove", firstHandle);      # Remove the first one #
return(TRUE);
```

shift2D

```
# Skews a 2D dataset by shifting its slices proportional to the slice number #

program Shift2D(int inc, lastNo);

int     sliceNo, shiftCount, handle;

onError(TERMINATE);     # abort program on error (that's the default) #
# Initialize Variables #
sliceNo = 2;
shiftCount = inc;
handle = getHandleOfPrimary;
# Extract first slice, don't process, just put into primary #
execCmd("prSlice -1 prim 1");
# Loop over all slices #
while(sliceNo <= lastNo)
        # Get input into secondary, extract next slice, result to secondary #
        execCmd("vpCurrent 1 sec", handle);
        execCmd("prSlice -1 sec", sliceNo);
        # Shift slice, recombine with primary #
        execCmd("prShift -1 sec 0", shiftCount, "shift");
        execCmd("vpRToSec -1 -1");
        execCmd("prBuild2D 'f2'");
        execCmd("vpRToPrim -1 -1");
        # Increment counters, loop until done #
        sliceNo = sliceNo + 1;
        shiftCount = shiftCount + inc;
endwhile;
return(TRUE);
```

acqDemo

```
# This program demonstrates the acquisition support from within a ProDeL program. The experiment #
# selected for the current viewport will be used. #

program acqDemo();
pointer      experimentP, datasetP, nameP;
boolean      ret;
int          i, dimension[8];
real         value, min, max;
char         buffer[80], units[16];

printLn("------------------------------------------------------------");
printLn("This is a basic ProDeL program using the acquisition library");
printLn("");

# Start by initializing some variables #
i = 0;
while (i < 8)
    dimension[i] = 0;
    i = i + 1;
endwhile;

# get a reference to the experiment selected in the current viewport #
experimentP = aqGetSelectedExp(-1);

# ensure we found a correct experiment #
if (experimentP == NIL)
    printLn("No experiment has been selected for current viewport");
    printLn("------------------------------------------------------------");
    return(FALSE);
endif;

# display a description for the selected experiment #
```

```
# The output generated by the command aqExpDesc is very useful when writing a ProDeL program #
# involving the acquisition. It lists the parameters used in the experiment and the parameter properties. #
nameP = aqGetExpName(experimentP);

strcpy(buffer, "aqExpDesc ");
strcat(buffer, nameP);

printLn("Displaying a description and information for experiment ", nameP);
execCmd(buffer);

# change some general settings #
aqSetComment("modified by ProDeL program");
aqSetSplName("unspecified");
aqSetSplFormula("-");
aqSetSplAggrState(AQ_SPL_POWDER);

aqGetComment(buffer, 80);
printLn(" Comment:                ", buffer);
aqGetSplName(buffer, 80);
printLn(" Sample name:            ", buffer);
aqGetSplFormula(buffer, 80);
printLn(" Sample formula:         ", buffer);
printLn(" Sample aggregation state: ", aqGetSplAggrState);

# and now deal with parameters #
# Remark: parameters values can be set by value or by defining the value in a string. For numeric #
# parameters, one can use, aqSetIntParValue, aqSetRealParValue, or aqSetStrParValue. It does not matter, #
# except that you might loose flexibility or resolution. The same is true while retrieving a parameter value. #
# You can retrieve it as a numeric value, or as a string. If retrieved as a string the correct formatting will be #
# performed according to the resolution of the parameter. #

aqMbcOperate(experimentP);
aqExpActivate(experimentP);
```

```
ret   = aqSetIntParValue(experimentP, "NbScansToDo", 8, dimension, 4);
i     = aqGetIntParValue(experimentP, "NbScansToDo", 8, dimension);

printLn("");
printLn("Acquisition is going to perform ", i, " sweeps");

# remark: both will be turn On. The first one using a string argument, the second one using the numeric #
# index in the discrete list of values. The first index is always 0. However, if a numeric parameter is set #
# a numeric argument gives the desired value, not the index. The value will be rounded to one of the values #
# in the discrete list. #
ret   = aqSetStrParValue(experimentP, "BaselineCorr", 8, dimension, "On");
ret   = aqSetIntParValue(experimentP, "ReplaceMode",  8, dimension, 0);

ret   = aqSetIntParValue(experimentP,  "TimeConst", 8, dimension, 5);
ret   = aqSetRealParValue(experimentP, "ConvTime", 8, dimension, 165.25);
ret   = aqSetStrParValue(experimentP,  "ConvTime", 8, dimension, "10.24");

# another example on how to set discrete values using indices or by name #
ret   = aqSetIntParValue(experimentP, "ModOutput", 8, dimension, 1);
ret   = aqSetStrParValue(experimentP, "ModOutput", 8, dimension, "Internal");

ret   = aqGetBoolParValue(experimentP, "AFCTrap", 8, dimension);
if (ret == FALSE)
        ret = aqSetBoolParValue(experimentP, "AFCTrap", 8, dimension, TRUE);
endif;

# let's retrieve information about a specific parameter #
ret = aqGetParUnits(experimentP, "CenterField", units,  16);
ret = aqGetParLabel(experimentP, "CenterField", buffer, 80);
if (ret == FALSE)
        printLn("This experiment does not use the CenterField parameter");
        printLn("-------------------------------------------------------");
```

```
        return(FALSE);
endif;

value = aqGetRealParValue(experimentP, "CenterField", 8, dimension);
min   = aqGetParMinValue(experimentP,  "CenterField");
max   = aqGetParMaxValue(experimentP,  "CenterField");

printLn("");
printLn("Parameter CenterField information");
printLn(" prompt label is: ", buffer);
printLn(" min: ", min, ", max: ", max, ", current value: ", value, " ", units);
min = aqGetParFineSteps(experimentP, "CenterField");
max = aqGetParCoarseSteps(experimentP, "CenterField");
printLn(" fine stepping factor is ", min, " and coarse stepping factor is ", max);
printLn(" parameter type is ", aqGetParType(experimentP, "CenterField"));
printLn(" number of dimensions: ", aqGetParNbDim(experimentP, "CenterField"));
print(" and dimension sizes:  ");
i = 0;
while (i < 8)
        print(aqGetParDimSize(experimentP, "CenterField", i), ", ");
        i = i + 1;
endwhile;
printLn("");

ret = aqSetRealParValue(experimentP, "CenterField", 8, dimension, 3400.00);
ret = aqSetRealParValue(experimentP, "SweepWidth",  8, dimension, 1000.00);
ret = aqSetIntParValue(experimentP,  "Resolution",  8, dimension, 512);

# perform an auto-tuning and start the acquisition #
printLn("");
printLn("Current state of experiment is ", aqGetExpState(experimentP));

printLn("Performing a fine tune...");
```

```
aqMbcFineTune(experimentP);

printLn("Running the experiment and waiting...");
aqExpRunAndWait(experimentP);

# Depending on your xepr configuration, a dataset might not automatically be generated at the end of each #
# acquisition. Therefore we create one here. #
datasetP = getCopyOfPrimary;

if (datasetP <> NIL)
        i = storeCopyOfDset(datasetP);
        printLn("New dataset has been saved under handle ", i);
endif;

aqMbcStandby(experimentP);

printLn("Acquisition done");
printLn("------------------------------------------------------------");
return(TRUE);
```

acqManipDemo

```
# This ProDeL program demonstrates the usage of the acquisition and manipulation support. #
# The program will perform a field sweep experiment in viewport 1 (argument viewport_1), extract #
# the peaks and perform a time sweep experiment for each peak in viewport 2 (argument viewport_2). #
# Each acquisition result is stored in a dataset. #

program acqManipDemo(int viewport_1, viewport_2);

real        value;
int         i, j, k;
pointer     exp1P, exp2P, datasetP, peakDsetP;
pointer     nameP;
boolean     ret;
int         dimension[8];
char        parName[20];


# Below are the parameters that can be customized #
strcpy(parName, "StaticField");

exp1P = aqExpBuild("FieldSweep", "CW", "B0VL", "NONE", "IADC", FALSE, FALSE, FALSE);
exp2P = aqExpBuild("TimeSweep" , "CW", "ETIM", "NONE", "IADC", FALSE, FALSE, FALSE);


# Build experiments and perform other variable init. #
if (exp1P == NIL or exp2P == NIL)
        printLn("Cannot built experiments. Aborting ProDeL program.");
        return(FALSE);
endif;

i = 0;
while (i < 8)
        dimension[i] = 0;
```

```
        i = i + 1;
endwhile;

# Output messages #
printLn("Created fieldsweep and time sweep experiments");

# Assign field sweep to viewport and start acquisition #
nameP = aqGetExpName( exp1P );
aqSetSelectedExp( viewport_1, nameP );

execCmd("vpCurrViewp ", viewport_1);

aqMbcFineTune( exp1P );
aqExpRunAndWait( exp1P );

# Store results of experiment into dataset #
datasetP = getCopyOfPrimary;

if (datasetP == NIL)
        printLn("Could not access fieldsweep dataset. Aborting ProDeL program.");
        return(FALSE);
endif;

i = storeCopyOfDset(datasetP);
printLn("Stored field sweep dataset under handle", i);

# Perform manipulation #
execCmd("prPeakPick -1 Primary Current Current peak win 10 10");

# Get a copy of the result dataset and perform a time sweep for each peak. #
peakDsetP = getCopyOfResult;

if (peakDsetP == NIL)
```

```
        printLn("Manipulation failed. Aborting ProDeL program.");
        return(FALSE);
endif;

# Assign time sweep experiment to viewport_2 and loop experiment for each static field = peak value #
nameP = aqGetExpName( exp2P );
aqSetSelectedExp( viewport_2, nameP );

execCmd("vpCurrViewp ", viewport_2);

# i is the number of peaks detected #
i = getNrOfPoints(peakDsetP, X_ABSC);
j = 0;

while (j < i)
        # Update the static field parameter value #
        value = getValue(peakDsetP, j, X_ABSC);
        ret   = aqSetRealParValue(exp2P, parName, 8, dimension, value);
        value = aqGetRealParValue(exp2P, parName, 8, dimension);
        printLn("Time sweep acquisition with static field set to ", value);
        # Perform a fine tuning and start the acquisition #
        aqMbcFineTune(exp2P);
        aqExpRunAndWait(exp2P);
        # Store results of experiment into dataset#
        datasetP = getCopyOfPrimary;
        if (datasetP <> NIL)
                k = storeCopyOfDset(datasetP);
                printLn("Stored Time sweep dataset under handle", k);
        endif;
        # Next peak value to use as static field #
        j = j + 1;
endwhile;
```

```
printLn("Acquisition/ProDeL program done.");

return(TRUE);
```

acqStepParDemo

```
# This ProDeL program demonstrates the usage of the acquisition support. The program is running in #
# the background, i.e. is triggered by the end of an acquisition. It will step a predefined parameter value #
# and restart the acquisition. #
# To work properly, this program must be compiled and defined as acquisition post process command to #
# be executed. Also do not forget to enable the execution of the acquisition post process commands (See #
# menu 'Acquisition->Post process"). Once the post processing has been setup and an experiment has been #
# assigned to the desired viewport, start the acquisition for the desired viewport. #
# This example can be used to perform 3D, 4D, etc. acquisitions using the basic 1D and 2D acquisitions we #
# currently support. #
# The result will be a series of datasets for the additionally swept parameter.
# The following variables can be customized: #
#       viewport        The viewport used to run the acq. #
#                       An experiment must be assigned to this viewport before starting the #
#                       acquisition #
#       parStep         Parameter value increment. Each time this program is run and if the end #
#                       has not been reached, this value is added to the current parameter value. #
#       endParValue     Parameter end value to reach #
#       parName         Parameter name to step #

program acqStepParDemo();

real        parStep, endParValue;
real        min, max, value, newValue;
int         viewport, i;
pointer      experimentP, datasetP;
boolean      ret;
int         dimension[8];
char        parName[20];

# Below are the parameters that can be customized #
viewport= 1;
parStep = 10;
```

```
endParValue     = 180;
strcpy(parName, "ModPhase");

# Get reference to experiment used in desired viewport #
experimentP = aqGetSelectedExp( viewport );

if (experimentP == NIL)
        printLn("No experiment has been assigned to viewport ", viewport);
        printLn("Aborting ProDeL program");
        return(FALSE);
endif;

# Store current experiment result into dataset #
datasetP = getCopyOfPrimary;

if (datasetP <> NIL)
        i = storeCopyOfDset(datasetP);
        printLn("New dataset has been stored under handle ", i);
endif;

# Retrieve parameter information #
i = 0;
while (i < 8)
        dimension[i] = 0;
        i = i + 1;
endwhile;

min   = aqGetParMinValue(experimentP, parName);
max   = aqGetParMaxValue(experimentP, parName);
value = aqGetRealParValue(experimentP, parName, 8, dimension);
newValue = value + parStep;

if (newValue > endParValue or newValue > max)
```

```
        aqMbcStandby(experimentP);
        printLn("Acquisition/ProDeL program done");
        printLn;
        return(TRUE);
endif;

ret = aqSetRealParValue(experimentP, parName, 8, dimension, newValue);
printLn("Running experiment. New parameter value: ", newValue);
aqMbcFineTune(experimentP);
aqExpRun(experimentP);
return(TRUE);
```

Experiment Devices

The following is a list of available devices for control in various experiment environments (i.e. CW Field Sweep, Pulse, etc.). The list of devices was generated using the following ProDeL program which displays the description of the current experiment and therefore allows control over the experiment via the commands in the ProDeL Acquisition Library. Examples of three different experiments are presented to provide an overview of the keywords typical for common experimental situations. The final section outlines the keywords available through the hidden acquisition experiment. This set of keywords are used for accessing devices not available through the standard experiments and allow reading and controlling of devices which may be hidden during specific experiment environments such as the SpecJet during CW acquisition.

Reading and setting of parameter values are achieved through the ProDeL Acquisition Library commands: aqGet... and aqSet...
For example to read the current field value:

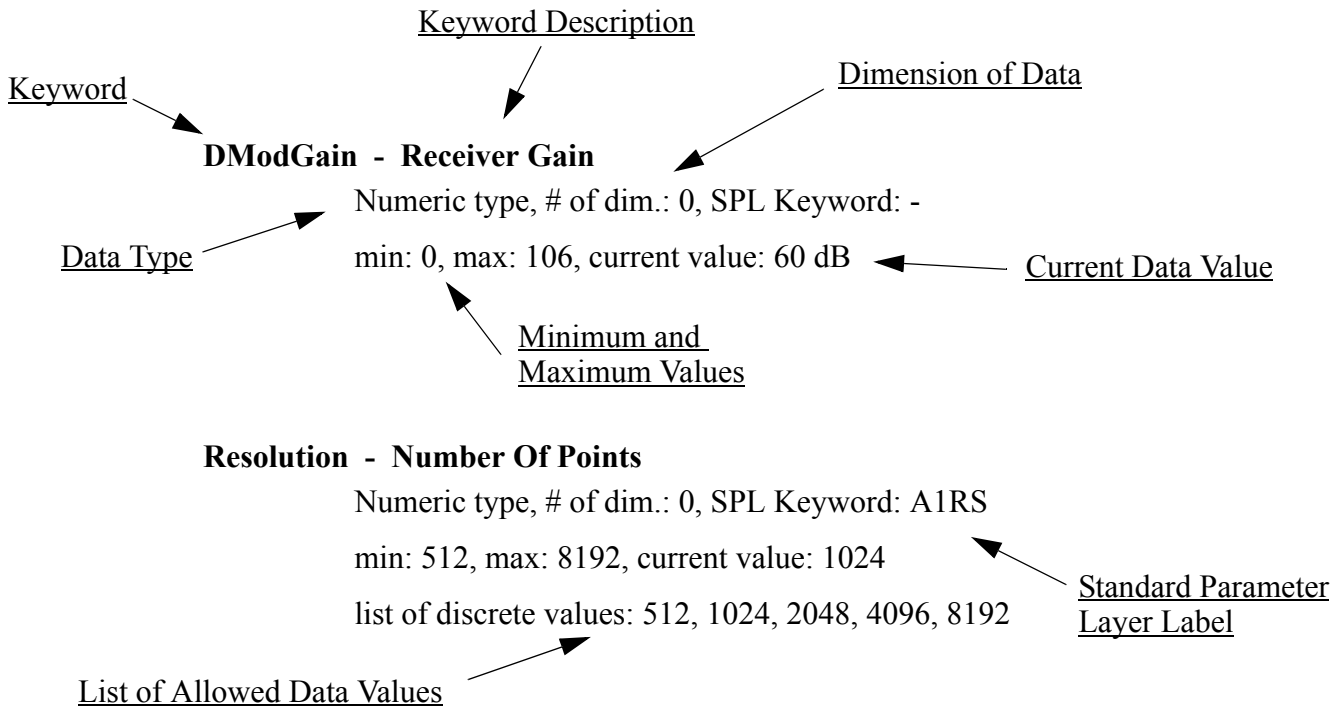
fieldReal = aqGetRealParValue(ExpPtr, “FieldValue”, 0, DimPtr)

and to set the current field position:

ansBool = aqSetRealParValue(ExpPtr, “FieldPosition”, 0, DimPtr, 3352.12)

where fieldReal is of type real, ansBool is of type boolean indicating if the set command succeeded, ExpPtr is a pointer to the current experiment, and DimPtr is a pointer to an array of type int.

Summary of Keyword Entries



Keyword:	Name to use for accessing parameter data values.
Keyword Description:	Expanded name of parameter.
Data Type:	Type of data returned for read operations and expected data type for set operations (Numeric, Enumeration, or Boolean).
Dimensions of Data:	Number of dimensions for parameter type. If greater than 0, the size of each dimension is specified.
Minimum/Maximum Values:	Absolute minimum and maximum allowed values for parameter.
Current Data Value:	Current set value for parameter.
Standard Parameter Layer Label:	Identifying label for the Standard Parameter Layer in the descriptor file (DSC file) for this parameter.
List of Allowed Data Values:	Possible values for this parameter.

Experiment Description Reader

```
# display a description for the selected experiment. The output generated by the command aqExpDesc is #
# very useful when writing a ProDeL program involving the acquisition. It lists the parameters used in the #
# experiment and the parameter properties. #

# give program a name #
program DescriptionRead();

# define some pointers and a character buffer #
pointer     nameP, experimentP;
char        buffer[80];

# get the reference to the experiment in current viewport #
experimentP = aqGetSelectedExp(-1);

# get the name of the experimentP #
nameP = aqGetExpName(experimentP);

# get the description #
strcpy(buffer, "aqExpDesc ");
strcat(buffer, nameP);

# display the description #
execCmd(buffer);
```

CW - Field Sweep Experiment

DEVICE acqStart

DEVICE fieldCtrl

AtCenter - Center

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

AtLeftBorder - Left

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

AtRightBorder - Right

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

CenterField - Center Field

Numeric type, # of dim.: 0, SPL Keyword: A1CT

min: 0, max: 14800, current value: 12140.00 G

Delay - Settling Delay

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.0 s

FFMarkerField - FF-Lock Marker Field

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 3420, max: 3540, current value: 3480.0

FieldFlyback - Field Flyback

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: On, Off

FieldPosition - Field Position

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 12140.000 G

FieldValue - Field

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.000 G

FieldWait - Field Settling

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Wait LED off

list of discrete values: Do not wait, Given delay, Wait LED off, Wait stable

GFactor - Sample g Factor

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 10, current value: 2.000000

SetToSampleG - Set Field to g Factor

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

SweepDirection - Sweep Direction

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Up

list of discrete values: Up, Down

SweepPos - Sweep Position

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2048

SweepWidth - Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: A1SW

min: 0, max: 16000, current value: 200.0 G

DEVICE fieldSweep

DEVICE freqCounter

FrequencyMon - Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: MWFQ

min: 0, max: 500, current value: 0.0 GHz

QMonitBridge - Bridge Monitoring

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

DEVICE mwBridge

AcqFineTuning - Acq Fine Tuning

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Never

list of discrete values: Never, Each Slice Scan

EWSMBC - EWS MBC

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Power - Power

read-only

Numeric type, # of dim.: 0, SPL Keyword: MWPW

min: 0, max: 1000, current value: 0.000 mW

PowerAt0DB - Power at 0 dB

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.000 mW

PowerAtten - Attenuation

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 25 dB

TuneStateExpMon - Tune State Exp.

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values: Frequency, Phase, Bias, AFC, Iris, Upper Iris Limit, Lower Iris Limit

DEVICE recorder

Abs1Data - Abscissa 1 Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

Abs1Name - Abscissa 1 Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Field

Abs1Type - BES3T Abs1Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: IDX

list of discrete values: NODATA, IDX, IGD, NTUP

Abs2Data - Abscissa 2 Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

Abs2Name - Abscissa 2 Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

Abs2Type - BES3T Abs2Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: IGD

list of discrete values: NODATA, IDX, IGD, NTUP

AutoScale - Auto Scaling

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: On, Off

AutoScaleTrg - AutoScaleTrg

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

BaselineCorr - Auto Offset

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

DataRange - Data Range

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0, 0

ImagDataName - Imaginary Data Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

NbScansAcc - Accumulated Scans

read-only

Numeric type, # of dim.: 0, SPL Keyword: AVGS

min: 0, max: 4.29497e+09, current value: 0

NbScansDone - Scans Done

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

NbScansToDo - Number Of Scans

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 1

RealDataName - Real Data Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Intensity

ReplaceMode - Replace Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

DEVICE scanEnd

DEVICE signalChannel

AFCTrap - AFC Trap Filter

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Abs1Data - Abs1Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 512, SPL Keyword: -

min: 0, max: 512, current value: -

Abs1Name - Abs1Name

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

CalibName - Calibration Data Set

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: RESO

current value: Qp0502_Test

Calibrated - Calibrated

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

ConvTime - Conversion Time

Numeric type, # of dim.: 0, SPL Keyword: SPTP

min: 0.16, max: 5242.88, current value: 5.12 ms

list of discrete values: 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24, 20.48, 40.96, 81.92, 163.84, 327.68, 655.36, 1310.72, 2621.44, 5242.88

DModAFCTrap - AFC Trap Filter

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

DModAmp - Modulation Amplitude

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 1.00 G

DModCalibrated - Calibrated

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

DModDetectSCT - Detecting SCT

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: First

list of discrete values: First, Second

DModEliDelay - Ext. Lock In Delay

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.3, max: 10000, current value: 1.0 us

DModExtLockIn - External Lock In

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DModExtTrigger - External Trigger

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DModFieldMod - Field Modulation SCT

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: First

list of discrete values: First, Second

DModGain - Receiver Gain

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 106, current value: 60 dB

DModHighPass - High Pass Filter

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

DModIntegrator - Lock In Integrator

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

DModModOutput - Modulation Output

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Internal

list of discrete values: External, Internal

DModSignalInput - Signal Input

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Internal

list of discrete values: External, Internal

DModTimeConst - Time Constant

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.01, max: 5242.88, current value: 1.28 ms

list of discrete values: 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24, 20.48, 40.96, 81.92, 163.84, 327.68, 655.36, 1310.72, 2621.44, 5242.88

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 512, SPL Keyword: -

min: 0, max: 255, current value: -

Data1 - Data1

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 512, SPL Keyword: -

min: 0, max: 255, current value: -

DataRange - DataRange

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0, 0

DoubleModFreq - Modulation Frequency

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.1, max: 100, current value: 5.00 kHz

DoubleModPhase - Double Modulation Phase

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.0

DoubleMode - Double Modulation

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DualMode - Setup Scan SCT

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: A

list of discrete values: A, B, A&B

DualTrace - Dual Trace

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

EliDelay - Ext. Lock In Delay

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.3, max: 10000, current value: 1.0 us

ExtLockIn - External Lock In

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ExtTrigger - External Trigger

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ForbidNorm - Normalization Forbidden

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

Gain - Receiver Gain

Numeric type, # of dim.: 0, SPL Keyword: RCAG

min: 0, max: 106, current value: 60 dB

Harmonic - Harmonic

Numeric type, # of dim.: 0, SPL Keyword: RCHM

min: 1, max: 2, current value: 1

HighPass - High Pass Filter

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Integrator - Lock In Integrator

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ModAmp - Modulation Amplitude

Numeric type, # of dim.: 0, SPL Keyword: B0MA

min: 0, max: 100, current value: 1.00 G

ModFreq - Modulation Frequency

Numeric type, # of dim.: 0, SPL Keyword: B0MF

min: 0.1, max: 100, current value: 100.00 kHz

ModInput - Modulation Input

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Internal

list of discrete values: Internal, External

ModOutput - Modulation Output

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Internal

list of discrete values: External, Internal

ModPhase - Modulation Phase

Numeric type, # of dim.: 0, SPL Keyword: RCPH

min: -1.79769e+308, max: 1.79769e+308, current value: 0.0

Offset - Offset

Numeric type, # of dim.: 0, SPL Keyword: RCOF

min: -100, max: 100, current value: 0.0 %

QuadMode - Quadrature Detection

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

QuadPhase - Quad Detection Phase

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 90.0

RealDataName - RealDataName

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

RecvLevel - Receiver Level

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -100, max: 100, current value: 0 %

Resolution - Number Of Points

Numeric type, # of dim.: 0, SPL Keyword: A1RS

min: 512, max: 8192, current value: 1024

list of discrete values: 512, 1024, 2048, 4096, 8192

Resonator - Resonator

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 2, current value: 1

SctNorm - Normalize Acquisition

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

SctRevision - Signal Channel Revision

read-only

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: SCT

list of discrete values: SCT, DSC2

SelfTest - Self Test

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

SetupScanMode - Setup Scan

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

SetupScanWidth - Sweep Width

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100 %

SignalInput - Signal Input

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Internal

list of discrete values: External, Internal

SignalLevel - Signal Level

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -2.09715e+06, max: 2.09715e+06, current value: 0

SuscTc - Setup Scan TC

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.01, max: 5242.88, current value: 0.01 ms

list of discrete values: 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24, 20.48, 40.96, 81.92, 163.84, 327.68, 655.36, 1310.72, 2621.44, 5242.88

SweepTime - Sweep Time

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100000, current value: 5.24 s

list of discrete values: 0.08, 0.16, 0.33, 0.66, 1.31, 2.62, 5.24, 10.49, 20.97, 41.94, 83.89, 167.77, 335.54, 671.09, 1342.18, 2684.35, 5368.71, 10737.42, 21474.84, 42949.67

TimeConst - Time Constant

Numeric type, # of dim.: 0, SPL Keyword: RCTC

min: 0.01, max: 5242.88, current value: 1.28 ms

list of discrete values: 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24, 20.48, 40.96, 81.92, 163.84, 327.68, 655.36, 1310.72, 2621.44, 5242.88

TuneCaps - Tuning Caps

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 255, current value: 32

CW - Field vs Time Experiment

All of the parameters from the previous experiment are present as in the previous experiment with these additions and changes:

Addition

DEVICE delay

Delay - Delay

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1e+06, current value: 100 ms

DelayMon - Count down

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1e+06, current value: 0 ms

NbPoints - Number Of Points

Numeric type, # of dim.: 0, SPL Keyword: A2RS

min: 1, max: 131072, current value: 10

Time - Delay Since Acq. Start

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1.79769e+308, current value: 0 s

Change

Abs2Name - Abscissa 2 Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Time

Pulse - Advanced Experiment

DEVICE acqStart

DEVICE cwBridge

AcqFineTuning - Acq Fine Tuning

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Never

list of discrete values: Never, Each Slice Scan

EWSMBC - EWS MBC

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Power - Power

read-only

Numeric type, # of dim.: 0, SPL Keyword: MWPW

min: 0, max: 1000, current value: 0.000 mW

PowerAt0DB - Power at 0 dB

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.000 mW

PowerAtten - Attenuation

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 25 dB

TuneStateExpMon - Tune State Exp.

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values: Frequency, Phase, Bias, AFC, Iris, Upper Iris Limit

Lower Iris Limit

DEVICE endor

EIECenterField - EIE Centerfield

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 12140.000 G

EIEENDORFreq - ENDOR Freq.

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 14.9021800 MHz/3.5 kG

EIEFieldPos - EIE Current Pos.

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 12140.000 G

EIEIsotope - Isotope

read-only

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: H1

list of discrete values: H1, H2, C13, N14, N15, F19, P31

EIERFSweepDir - EIE RF Sweep Direction

read-only

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Same

list of discrete values: Same, Opposite

EIEStaticField - EIE Static Field

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 12140.000 G

EIEStaticRF - EIE Static RF

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 38.000 MHz

EIESweepWidth - EIE Sweep Width

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 200.000 G

FTXAxisQuant - FT X Axis Quantity

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Time

FTYAxisQuant - FT Y Axis Quantity

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Magnetic Field

RF1Atten - RF1 Attenuator

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 80, current value: 60.000 dB

RF1FreqPos - Current RF1

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 38.000 MHz

RF1StartFreq - RF1 Start

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 38.000 MHz

RF1SweepWidth - RF1 Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 30.000 MHz

RF2Atten - RF2 Attenuator

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 80, current value: 60.000 dB

RF2FreqPos - Current RF2

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 38.000 MHz

RF2StartFreq - RF2 Start

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 38.000 MHz

RF2SweepWidth - RF2 Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 30.000 MHz

RFSrcMixing - RF Sources Mixing

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Add

list of discrete values: Add, Multiply

SumAtten - Sum Attenuator

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 80, current value: 10.000 dB

SumAttenStart - Sum Attenuator Start

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 80, current value: 0.000 dB

SumAttenWidth - Sum Att. Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 80, current value: 80.000 dB

DEVICE fieldCtrl

AtCenter - Center

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

AtLeftBorder - Left

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

AtRightBorder - Right

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

CenterField - Center Field

Numeric type, # of dim.: 0, SPL Keyword: A1CT

min: 0, max: 14800, current value: 12140.00 G

Delay - Settling Delay

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.0 s

FFMarkerField - FF-Lock Marker Field

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 3420, max: 3540, current value: 3480.0

FieldFlyback - Field Flyback

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: On, Off

FieldPosition - Field Position

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: B0VL

min: -1.79769e+308, max: 1.79769e+308, current value: 12140.000 G

FieldValue - Field

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.000 G

FieldWait - Field Settling

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Wait LED off

list of discrete values: Do not wait, Given delay, Wait LED off, Wait stable

GFactor - Sample g Factor

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 10, current value: 2.000000

SetToSampleG - Set Field to g Factor

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

SweepDirection - Sweep Direction

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Up

list of discrete values: Up, Down

SweepPos - Sweep Position

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2048

SweepWidth - Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: A1SW

min: 0, max: 16000, current value: 200.0 G

DEVICE freqCounter

FrequencyMon - Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: MWFQ

min: 0, max: 500, current value: 0.0 GHz

QMonitBridge - Bridge Monitoring

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

DEVICE ftBridge

Attenuation - MW Attenuation

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 60.00 dB

DevOption - Device Option

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: QBandUPG_PBC

list of discrete values: None, EasyPulse, ELDOR, StandardPulse, EScan, QBand, QBandPBC, QBandUPG, QBandUPG_PBC, Transient, QBandUPG_STD

DevType - FT Brigde Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: QBandUPG_PBC

list of discrete values: None, EasyPulse, ELDOR, StandardPulse, EScan, QBand, QBandPBC, QBandUPG, QBandUPG_PBC, Transient, QBandUPG_STD

ELDORAtt - ELDOR Attenuation

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 30, current value: 30 dB

ELDORFreqHigh - Maximum ELDOR Frequency

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 10.000000 GHz

ELDORFreqLow - Minimum ELDOR Frequency

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 9.100000 GHz

ELDORFreqMon - ELDOR Frequency

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.000000 GHz

FreqAInc - Frequency 1 Increment

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1e-08, max: 10, current value: 1e-04 GHz

list of discrete values: 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 0.001, 0.01, 0.1, 1, 10

FrequencyA - Frequency 1

Numeric type, # of dim.: 0, SPL Keyword: -

min: 32.55, max: 33.35, current value: 32.550000 GHz

PowAttenMon - Attenuation

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 0 dB

VideoBW - Video Bandwidth

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: 50 MHz

list of discrete values: 25, 50, 100, 200

VideoGain - Video Gain

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 66, current value: 33 dB

DEVICE ftEpr

AllVisibleSlct - All visible

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

AveragesPerScan - Averages Per Scan

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

CalcPatterns - Calculate

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ChannelSlct - Channel Selection

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: +x

list of discrete values: Acquisition Trigger, +x, +<x>, -x, -<x>, +y, +<y>, -y, -<y>, ELDOR, RF1 Gate, RF2 Gate

ELDORFreqStart - ELDOR Freq. Start

Numeric type, # of dim.: 0, SPL Keyword: -

min: 32.55, max: 33.35, current value: 32.550000 GHz

ELDORFreqWidth - ELDOR Sweep Width

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 0.8, current value: 0.800000 GHz

FTAcqModeSlct - Acquisition Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Run from Tables

list of discrete values: Run from Tables, Run from PulseSPEL, Read Transient, Start Transient

IntgTimeBase - Integrator Time Base

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Single Point ns

list of discrete values: Single Point, 4.0, 6.0, 8.0, 10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0, 2000.0, 5000.0, 10000.0, 20000.0, 50000.0, 100000.0

PPExtTrg - External Trigger

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

PPExtTrgSlope - External Trigger Slope

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Rising

list of discrete values: Rising, Falling

PatternEdit - Pattern Editor

auxiliary

Numeric type, # of dim.: 2, sizes: 31, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: - ns

PlsSPELCmd - PlsSPELCmd

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

PlsSPELEXPSlct - Experiment

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: <none>

list of discrete values: <none>

PlsSPELGlbPaF - PlsSPELGlbPaF

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELGlbTxt - PlsSPELGlbTxt

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELLISTSlct - Phase Cycling

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: <none>

list of discrete values: <none>

PlsSPELMsg - PlsSPELMsg

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELPhPrgEx - Phase Program

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Normal

list of discrete values: Normal, Continuous, Next Cycle, Skip Program

PlsSPELPrg - PulseSPEL Program

read-only

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELPrgPaF - PlsSPELPrgPaF

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELPrgTxt - PlsSPELPrgTxt

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELSetVar - PulseSPEL Variable

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELVerbMsg - PlsSPELVerbMsg

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

PlsSPELVerbose - PlsSPELVerbose

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ProtectMode - Pattern Control

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Auto

list of discrete values: Auto, Manual, Setup

Psd1 - PSD 1

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd10 - PSD 10

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd11 - PSD 11

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd12 - PSD 12

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd13 - PSD 13

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd14 - PSD 14

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd15 - PSD 15

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd16 - PSD 16

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd17 - PSD 17

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd18 - PSD 18

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd19 - PSD 19

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd2 - PSD 2

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd20 - PSD 20

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd21 - PSD 21

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd22 - PSD 22

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd23 - PSD 23

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd24 - PSD 24

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd25 - PSD 25

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd26 - PSD 26

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd3 - PSD 3

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd4 - PSD 4

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd5 - PSD 5

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd6 - PSD 6

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd7 - PSD 7

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd8 - PSD 8

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Psd9 - PSD 9

Numeric type, # of dim.: 2, sizes: 33, 4, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

QuadDetect - Quadrature Detection

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

RF1Prg - RF1 Prg

Numeric type, # of dim.: 2, sizes: 33, 3, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: - MHz

RF2Prg - RF2 Prg

Numeric type, # of dim.: 2, sizes: 33, 3, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: - MHz

RFModeSlct - RF Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: <not available>

list of discrete values: <not available>

ReplaceMode - Replace Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

RestPatterns - Restore

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ShotRepTime - Shot Rep. Time

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1.02, max: 2.04e+06, current value: 999.60 us

ShotsPLoop - Shots Per Point

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 1024, current value: 1

StartPlsPrg - Start

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

StopPlsPrg - Stop

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

SweepsPExp - Number Of Scans

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 132000, current value: 1

TriggerTimeOut - Trigger Time Out

Numeric type, # of dim.: 0, SPL Keyword: -

min: 3, max: 360000, current value: 10 s

XAxisQuant - X-Axis Quantity

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Time

list of discrete values: Time, Magnetic Field, RF1, RF2, EIE with RF1, EIE with RF2, RF Sum Attenuation, ELDOR

XSpecRes - X-Axis Size

Numeric type, # of dim.: 0, SPL Keyword: -

min: 4, max: 65535, current value: 1024

YAxisQuant - Y-Axis Quantity

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Magnetic Field

list of discrete values: Magnetic Field, RF1, RF2, EIE with RF1, EIE with RF2, RF Sum Attenuation, ELDOR

YSpecRes - Y-Axis Size

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 4096, current value: 1

DEVICE recorder

Abs1Data - Abscissa 1 Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

Abs1Name - Abscissa 1 Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

Abs1Type - BES3T Abs1Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: IDX

list of discrete values: NODATA, IDX, IGD, NTUP

Abs2Data - Abscissa 2 Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

Abs2Name - Abscissa 2 Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

Abs2Type - BES3T Abs2Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: IDX

list of discrete values: NODATA, IDX, IGD, NTUP

AutoScale - Auto Scaling

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: On, Off

AutoScaleTrg - AutoScaleTrg

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

BaselineCorr - Auto Offset

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0

DataRange - Data Range

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0, 0

ImagDataName - Imaginary Data Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Intensity

NbScansAcc - Accumulated Scans

read-only

Numeric type, # of dim.: 0, SPL Keyword: AVGS

min: 0, max: 4.29497e+09, current value: 0

NbScansDone - Scans Done

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

NbScansToDo - Number Of Scans

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 1

RealDataName - Real Data Name

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Intensity

ReplaceMode - Replace Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

DEVICE routeTrg

DEVICE sigChanSmall

DEVICE transRec

AcqMode - Acquisition Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Start Single Trace

list of discrete values: Start Single Trace, Start Dual Trace, Read Single Trace, Read Dual Trace

ResetDevice - Reset

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

TrRecTrgTimeOut - Trigger Time Out

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 3, max: 360000, current value: 10 s

TransPerScan - Transients Per Scan

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

Hidden Acquisition Experiment

DEVICE cwBridge

- AFCState - AFC
- Abs1Data - Abs1Data
- Abs1Name - Abs1Name
- CWBridgeType - CW Bridge Type
- CalibState - Calibration
- Data - Data
- Data1 - Data1
- DataRange - DataRange
- DiodeCurrent Diode Current
- Dispersion - Dispersion
- DualTrace - DualTrace
- DualTraceEnab - Dual Trace
- EWSMBC - EWS MBC
- Frequency - Frequency
- IrisDown - Iris
- IrisPreadjust - Autotune Iris Preadjustment
- IrisState - Iris Limits
- IrisUp - Iris
- LevelState - Leveller
- LockOffset - Lock Offset
- LockPhase - Lock Phase
- LogScaleEnab - Log. Scale
- ModeZoom - Mode Zoom Factor
- OpMode - Operation Mode
- OpModeMon - Operation Mode
- Power - Power
- PowerAt0DB - Power at 0 dB
- PowerAtten - Attenuation
- PowerAttenMon - Attenuation
- PowerMon - Power
- QValue - Q-Value
- RealDataName RealDataName
- RefArm - Reference Arm
- RunIrisDown - Iris Run Down
- SignalBias - Bias
- SignalPhase - Signal Phase
- SignalPhaseMon - Signal Phase
- Tune - Auto Tuning
- TuneState - Tuning State

DEVICE ffLock

- Abs1Data - Abs1Data
- Abs1Name - Abs1Name
- Attenuator - Attenuator
- Data - Data
- Data1 - Data1
- DataRange - DataRange
- DialogPanelLock - DialogPanelLock
- DualTrace - DualTrace
- FFNbAccum - Number of Accumulations
- FFOffset - FF Lock Offset
- FFProp - Proportional Band
- FFTMPresent - FFLock / Teslameter Present
- GFactorMon - g Factor
- GmAcqTime - Acquisition Time
- GmAutoMode - Auto Mode
- GmAuxConf - Auxiliary Configuration
- GmContSpec - Get Continuous Spectrum
- GmDeadTime - Dead Time
- GmDefField - Default Field
- GmFFTLen - FFT Length
- GmFieldMon - Gaussmeter
- GmGain - Receiver Gain
- GmHomogenityMon - Homogeneity
- GmLockResol - Lock Resolution
- GmLowerLimit Lower Field Search Limit
- GmModulation - Modulation
- GmNbAccum - Number of Accumulations
- GmOpMode - Operation Mode
- GmPeakFilter - Peak Filtering
- GmPlsLen - HF Pulse Length
- GmProbeHead - Probe Head No.
- GmRecDelay - Recycle Delay
- GmSTLevel - Search Track Level
- GmSearchDir - Field Search Direction
- GmSpecScale - Ordinate Scale
- GmSpecType - Spectrum Type
- GmStepWidth - Step Width
- GmTestPeak - Test Peak
- GmType - Teslameter Type
- GmUpperLimit - Upper Field Search Limit
- GmUsed - Use Teslameter
- GmWaitTime - Field Settling Time
- LockMode - Lock In
- LockModeMon - Lock Status
- MarkerFieldMon - Marker Field
- MarkerMode - Marker Mode
- RealDataName - RealDataName
- Type - Type

DEVICE freqCounter

- FrequencyMon - Frequency
- QMonitBridge - Bridge Monitoring

RMPhase - RM Phase
RealDataName - RealDataName
RecPhase - Receiver Phase
SPFUOption - SPFU Option
STABMode - STAB Mode
TMLevel - Transmitter Level
TMLvlOption - TM Level Option
TMPhase - TM Phase
TuneStateMon Tuning State
VideoBW - Video Bandwidth
VideoBWMon - VideoBWMon
VideoGain - Video Gain
VideoGainMon - VideoGainMon
Void - Void
XAmp - +x Amplitude
XPhase - +x Phase
YAmp - +y Amplitude
YPhase - +y Phase

DEVICE gTempCtrl
AcqWaitTime - Temp Settling Time
AutoTune - PID Tuning
AutoTuneMon - PID Autotune
Derivative - Derivative Time
DerivativeMon - Dervative Time
EvapStatus - LN2 Evaporator Heater
EvapStatusMon - Evaporator Heater
GasCtrl - Gas Flow Control
GasFlow - Gas Flow [l/h]
GasFlowMon - Gas Flow
HeaterCtrl - Heater Control
HeaterPowerMon - Current Heater Power
HeaterStatus - Heater
HeaterstatusMon - Heater Status
ITCGasFlow - Gas Flow
Integral - Integral Time
IntegralMon - Integral Time
ItcGasFlowMon - Current Gas Flow
ItcHeaterPwr - Heater Power
LN2Power - LN2 Evaporator Heater Power
LN2PowerMon - Current LN2 Power
LN2TankMon - LN2 Tank Status
LocalRemote - ITC Operation Mode
OverHeatMon - Probe Head Heater
Proportional - Proportional Band
ProportionalMon - Proportional Band
PwrLimit - Heater Power Limit
TempUsage - Temperature taken from
Temperature - Temperature
TemperatureMon - Current Temperature
Tolerance - Tolerance
Type - Type

UnlockPID - Unlock PID Settings
VTUsed - Use VTU

DEVICE sctCalib

CalAFCTrap - AFC Trap Filter
CalHighPass - High Pass Filter
CalResonator - Resonator
CalibData - Results
CalibDbDelete - Delete Data Set
CalibName - Calibration Data Set
RFCalSetName - RF Calibration Data Set
RFCalibData - Results
RFCalibDbDelete - Delete Data Set

DEVICE specJet

Abs1Data - Abs1Data
Abs1Name - Abs1Name
AbscZoom - Zoom
AveragePause - Pause
AverageStart - Run
AveragesDoneMon - Averages Done
Ch1Select - Channel 1
Ch1SigOffset - Channel 1 Offset
Ch2Select - Channel 2
Ch2SigOffset - Channel 2 Offset
ClockSource - Clock Source
Data - Data
Data1 - Data1
DataRange - DataRange
DitherMode - Dither Mode
DualTrace - TrDualTrace
FullScale - FS
InterleaveMode - Interleave Mode
IntgDisplay - INTG
NoOfAverages - No. of Averages
NoOfPoints - No. of Points
RealDataName - RealDataName
RepetitiveMode - Repetitive Mode
ResetDevice - Reset
ScaleDivideBy2 - / 2
ScaleFactor - ScaleFactor
ScaleTimes2 - * 2
ScanTime - Scan Time
TimeBase - Time Base
TriggerLevel - Trigger Level
TriggerMode - Trigger Mode
TriggerSlope - Trigger Slope
TriggerSource - Trigger Source

DEVICE cwBridge

AFCState - AFC

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values: No AFC

Abs1Data - Abs1Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 256, SPL Keyword: -

min: 0, max: 255, current value: -

Abs1Name - Abs1Name

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

CWBridgeType - CW Bridge Type

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Standard

list of discrete values: Standard, Dispersion

CalibState - Calibration

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Uncalibrated

list of discrete values: Uncalibrated, Calibrated

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 256, SPL Keyword: -

min: 0, max: 255, current value: -

Data1 - Data1

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 256, SPL Keyword: -

min: 0, max: 255, current value: -

DataRange - DataRange

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.0000000000000000e+00, 0.0000000000000000e+00

DiodeCurrent Diode Current

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 400, current value: 3.51 uA

Dispersion - Dispersion

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DualTrace - DualTrace

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DualTraceEnab - Dual Trace

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

EWSMBC - EWS MBC

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Frequency - Frequency

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 0

IrisDown - Iris

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

IrisPreadjust - Autotune Iris Preadjustment

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

IrisState - Iris Limits

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

IrisUp - Iris

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

LevelState - Leveller

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Unlevelled

list of discrete values: Unlevelled, Levelled

LockOffset - Lock Offset

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -100, max: 100, current value: 0.28 %

LockPhase - Lock Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 0

LogScaleEnab - Log. Scale

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ModeZoom - Mode Zoom Factor

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 8, current value: 1

list of discrete values: 1, 2, 4, 8

OpMode - Operation Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Stand By

list of discrete values: Stand By, Tune, Operate

OpModeMon - Operation Mode

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Stand By

list of discrete values: Stand By, Tune, Operate

Power - Power

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: MWPW

min: 0, max: 1000, current value: 0.000 mW

PowerAt0DB - Power at 0 dB

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.000 mW

PowerAtten - Attenuation

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 60 dB

PowerAttenMon - Attenuation

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 60 dB

PowerMon - Power

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.000 mW

QValue - Q-Value

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

RealDataName RealDataName

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

RefArm - Reference Arm

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: On, Off

RunIrisDown - Iris Run Down

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

SignalBias - Bias

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 0

SignalPhase - Signal Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2017

SignalPhaseMon - Signal Phase

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.26

Tune - Auto Tuning

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Stop

list of discrete values: Up, Down, Fine, Stop

TuneState - Tuning State

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Q-Band

list of discrete values: Q-Band, Frequency, Phase, Bias, AFC, Iris, Upper Iris Limit, Lower Iris Limit

DEVICE ffLock

Abs1Data - Abs1Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 1024, SPL Keyword: -

min: 0, max: 65535, current value: -

Abs1Name - Abs1Name

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

Attenuator - Attenuator

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 255, current value: 26

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 1024, SPL Keyword: -

min: 0, max: 65535, current value: -

Data1 - Data1

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 65535, current value: 0

DataRange - DataRange

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.0000000000000000e+00, 0.0000000000000000e+00

DialogPanelLock - DialogPanelLock

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DualTrace - DualTrace

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

FFNbAccum - Number of Accumulations

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 200, current value: 3

FFOffset - FF Lock Offset

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -10000, max: 10000, current value: 0.000

FFProp - Proportional Band

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 250.00 1/1000

FFTMPresent - FFLock / Teslameter Present

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: fftmPresent

list of discrete values: None, ffPresent, tmPresent, fftmPresent

GFactorMon - g Factor

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

GmAcqTime - Acquisition Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.01, max: 30, current value: 1.00 ms

GmAutoMode - Auto Mode

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1, current value: 1

GmAuxConf - Auxiliary Configuration

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1000, current value: 0.00

GmContSpec - Get Continuous Spectrum

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

GmDeadTime - Dead Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.01, max: 100, current value: 10.00 ms

GmDefField - Default Field

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 450, max: 20000, current value: 3480.00 G

GmFFTLen - FFT Length

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 256, max: 32768, current value: 2048

list of discrete values: 256, 512, 1024, 2048, 4096, 8192, 16384, 32768

GmFieldMon - Gaussmeter

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: G

GmGain - Receiver Gain

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 40, max: 114, current value: 75

GmHomogenityMon - Homogeneity

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 40000, current value: 1

GmLockResol - Lock Resolution

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: 0.1 G

list of discrete values: 0.1, 0.01, 0.001

GmLowerLimit Lower Field Search Limit

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 450, max: 20000, current value: 3000.00 G

GmModulation - Modulation

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Positive

list of discrete values: Off, Last Direction, Positive, Negative

GmNbAccum - Number of Accumulations

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 10000, current value: 1

GmOpMode - Operation Mode

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Flash

list of discrete values: Flash, Search

GmPeakFilter - Peak Filtering

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 2, current value: 1

GmPlsLen - HF Pulse Length

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 100, current value: 5.00 us

GmProbeHead - Probe Head No.

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 1e+08, current value: 1.00

GmRecDelay - Recycle Delay

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 10000, current value: 1000.00 ms

GmSTLevel - Search Track Level

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.001, max: 10000, current value: 500.00

GmSearchDir - Field Search Direction

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Up

list of discrete values: Up, Stop Search, Down

GmSpecScale - Ordinate Scale

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.0001, max: 10000, current value: 1.00

GmSpecType - Spectrum Type

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: FID

list of discrete values: FID, FFT

GmStepWidth - Step Width

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 50, current value: 10 G

GmTestPeak - Test Peak

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 30000, current value: 0

GmType - Teslameter Type

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: ER036TM

list of discrete values: None, ER035M, ER036TM

GmUpperLimit - Upper Field Search Limit

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 450, max: 20000, current value: 4000.00 G

GmUsed - Use Teslameter

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

GmWaitTime - Field Settling Time

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.1, max: 20, current value: 1.00

LockMode - Lock In

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

LockModeMon - Lock Status

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: G

MarkerFieldMon - Marker Field

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: G

MarkerMode - Marker Mode

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

RealDataName - RealDataName

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

Type - Type

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: ER033S

list of discrete values: None, ER033M, ER033S

DEVICE freqCounter

FrequencyMon - Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: MWFQ

min: 0, max: 500, current value: 0.000000 GHz

QMonitBridge - Bridge Monitoring

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

DEVICE ftBridge

AFCGain - AFC Gain

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 10, current value: 2

AFCMode - AFC Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: AC

list of discrete values: Off, AC, DC

AFCTimeConst - AFC Time Constant

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: High

list of discrete values: Low, High

ALTMode - X/W Delay

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

Abs1Data - Abs1Data

read-only

Numeric type, # of dim.: 1, sizes: 256, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -

Abs1Name - Abs1Name

read-only

String type, # of dim.: 0, SPL Keyword: -

current value:

Attenuation - MW Attenuation

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 60.00 dB

AutoPhase - Phase Autotune

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

BrAdvMode - Advanced Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

BrCWMode - CW Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

BrMinXAmp - -<x> Amplitude

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

BrMinXPhase - -<x> Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 30.965 %

BrMinYAmp - -<y> Amplitude

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

BrMinYPhase - -<y> Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.000 %

BrPlsMode - Pulse Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

BrTransMode - Transient Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

BrXAmp - +<x> Amplitude

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

BrXPhase - +<x> Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 19.389 %

BrYAmp - +<y> Amplitude

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

BrYPhase - +<y> Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 %

CWMode - CW Mode

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

DCAFCPresent - DC AFC Present

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

DIGMode - DIG Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

Data - Data

read-only

Numeric type, # of dim.: 1, sizes: 256, SPL Keyword: -

min: 0, max: 65535, current value: -

Data1 - Data1

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 65535, current value: 0

DataRange - DataRange

read-only

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: 0.0000000000000000e+00, 0.0000000000000000e+00

Detection - Detection Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Signal

list of discrete values: Signal, RM, TM

DevOption - Device Option

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: ELDOR

list of discrete values: None, EasyPulse, ELDOR, StandardPulse, EScan, QBand, QBandPBC, QBandUPG, QBandUPG_PBC, Transient, QBandUPG_STD

DevType - FT Brigde Type

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: QBandUPG_PBC

list of discrete values: None, EasyPulse, ELDOR, StandardPulse, EScan, QBand, QBandPBC, QBandUPG, QBandUPG_PBC, Transient, QBandUPG_STD

DiodeCurrent - Matching Coarse

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 400, current value: 0

DualTrace - DualTrace

read-only

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ELDORAtt - ELDOR Attenuation

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 30, current value: 30 dB

ELDORFreqHigh - Maximum ELDOR Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 10.000000 GHz

ELDORFreqLow - Minimum ELDOR Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 9.200000 GHz

ELDORFreqMon - ELDOR Frequency

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 9.500000 GHz

ExtStabDown - External Stabilizer

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ExtStabUp - External Stabilizer

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

FreqAInc - Frequency 1 Increment

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1e-08, max: 10, current value: 1e-04 GHz

list of discrete values: 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 0.001, 0.01, 0.1, 1, 10

FreqBInc - Frequency 2 Increment

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1e-08, max: 10, current value: 1e-04 GHz

list of discrete values: 1e-08, 1e-07, 1e-06, 1e-05, 1e-04, 0.001, 0.01, 0.1, 1, 10

FreqIntv - Frequency Interval

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 200, current value: 100

FrequencyA - Frequency 1

Numeric type, # of dim.: 0, SPL Keyword: -

min: 33.55, max: 34.35, current value: 33.550000 GHz

FrequencyB - Frequency 2

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 10, current value: 9.500000 GHz

HPPMode - HPP Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

HoppingAmp - Phase Tuning

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 10000, current value: 100 kHz

IFOption - IF Option

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

INDMode - IF Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

LCWMode - LCW Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

LCWOption - LCW Option

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

LCWPhase - LCW Phase

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

LPPMode - LPP Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

LockOffset - Lock Offset

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: -100, max: 100, current value: 0 %

LockSearch - Lock Search

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

MWGain - MW Amplifier

read-only

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

MinXAmp - -x Amplitude

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

MinXPhase - -x Phase

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

MinYAmp - -y Amplitude

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

MinYPhase - -y Phase

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

MonPanel - Monitoring Panel

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: CW

list of discrete values: CW, DC, PH, DCPH

PhaseTuneMon - PhaseTuneMon

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

PhaseTuning - Phase Tuning

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

PowAttenMon - Attenuation

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 60, current value: 60 dB

PulseMode - Q Band Pulse Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

QMonitBridge - Bridge Monitoring

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: On

list of discrete values: Off, On

QuadMode - Quadrature Detection

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

RMPhase - RM Phase

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

RealDataName - RealDataName

read-only

String type, # of dim.: 0, SPL Keyword: -

current value:

RecPhase - Receiver Phase

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 4095

SPFUOption - SPFU Option

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

STABMode - STAB Mode

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

TMLevel - Transmitter Level

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

TMLvlOption - TM Level Option

read-only

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

TMPhase - TM Phase

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 100.000 %

TuneStateMon Tuning State

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values: Phase

VideoBW - Video Bandwidth

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: 200 MHz

list of discrete values: 20, 200

VideoBWMon - VideoBWMon

read-only

Enumeration type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values:

VideoGain - Video Gain

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 69, current value: 33 dB

VideoGainMon - VideoGainMon

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 33 dB

Void - Void

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

XAmp - +x Amplitude

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

XPhase - +x Phase

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

YAmp - +y Amplitude

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

YPhase - +y Phase

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 49.988 %

DEVICE gTempCtrl

AcqWaitTime - Temp Settling Time

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 3.6e+06, current value: 5.0 s

AutoTune - PID Tuning

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: No Tuning

list of discrete values: No Tuning, Tune

AutoTuneMon - PID Autotune

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: No Tuning

list of discrete values: No Tuning, Tune

Derivative - Derivative Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 min

DerivativeMon - Dervative Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 s

EvapStatus - LN2 Evaporator Heater

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

EvapStatusMon - Evaporator Heater

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

GasCtrl - Gas Flow Control

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Auto

list of discrete values: Manual, Auto

GasFlow - Gas Flow [l/h]

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: 0

list of discrete values: 0, 135, 270, 400, 535, 670, 800, 935, 1070, 1200, 1335, 1470, 1600, 1735, 1870, 2000

GasFlowMon - Gas Flow

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Failure

list of discrete values: Failure, OK

HeaterCtrl - Heater Control

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Auto

list of discrete values: Manual, Auto

HeaterPowerMon - Current Heater Power

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 %

HeaterStatus - Heater

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

HeaterstatusMon - Heater Status

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Off

list of discrete values: Off, On

ITCGasFlow - Gas Flow

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 99.9, current value: 0.00 arb. units

Integral - Integral Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 14.00 min

IntegralMon - Integral Time

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 s

ItcGasFlowMon - Current Gas Flow

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 arb. units

ItcHeaterPwr - Heater Power

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 % of limit

LN2Power - LN2 Evaporator Heater Power

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 %

LN2PowerMon - Current LN2 Power

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 %

LN2TankMon - LN2 Tank Status

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Empty

list of discrete values: Empty, Refill, Full, Not Connected

LocalRemote - ITC Operation Mode

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: LOCAL

list of discrete values: LOCAL, REMOTE

OverHeatMon - Probe Head Heater

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: OK

list of discrete values: OK, Overheating

Proportional - Proportional Band

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 36.00 K

ProportionalMon - Proportional Band

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 0.00 %

PwrLimit - Heater Power Limit

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 50.00 %

TempUsage - Temperature taken from

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Main Panel

list of discrete values: Main Panel, Experiment

Temperature - Temperature

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 1273, current value: 295.00 K

TemperatureMon - Current Temperature

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: STMP

current value: K

Tolerance - Tolerance

read-only

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 100, current value: 1.00 K

Type - Type

read-only, auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: ITC503

list of discrete values: None, ER4111VT, ER4121VT, ER4131VT, ITC502, ITC503

UnlockPID - Unlock PID Settings

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

VTUsed - Use VTU

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DEVICE sctCalib

CalAFCTrap - AFC Trap Filter

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

CalHighPass - High Pass Filter

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

CalResonator - Resonator

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 2, current value: 1

CalibData - Results

read-only, auxiliary

String type, # of dim.: 2, sizes: 6, 10, SPL Keyword: -

current value: -

CalibDbDelete - Delete Data Set

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

CalibName - Calibration Data Set

auxiliary

String type, # of dim.: 0, SPL Keyword: RESO

current value: Qp0502_Test

list of discrete values: Q_p0502demo, Qp0502_Test, p0503QE_0805, qte_demo, std_QT_1204, xen_newbuild

RFCalSetName - RF Calibration Data Set

auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

list of discrete values: <none>

RFCalibData - Results

read-only, auxiliary

String type, # of dim.: 2, sizes: 3, 7, SPL Keyword: -

current value: -

RFCalibDbDelete - Delete Data Set

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DEVICE specJet

Abs1Data - Abs1Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -3.40282e+38, max: 3.40282e+38, current value: - ns

Abs1Name - Abs1Name

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: Time

AbscZoom - Zoom

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 2, max: 4096, current value: 4096

AveragePause - Pause

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

AverageStart - Run

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

AveragesDoneMon - Averages Done

read-only, auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4.29497e+09, current value: 0

Ch1Select - Channel 1

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Ch1SigOffset - Channel 1 Offset

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2061

Ch2Select - Channel 2

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

Ch2SigOffset - Channel 2 Offset

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2060

ClockSource - Clock Source

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Ext

list of discrete values: Int, Ext

Data - Data

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -2.14748e+09, max: 2.14748e+09, current value: -

Data1 - Data1

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -2.14748e+09, max: 2.14748e+09, current value: -

DataRange - DataRange

read-only, auxiliary

Numeric type, # of dim.: 1, sizes: 2, SPL Keyword: -

min: -1.79769e+308, max: 1.79769e+308, current value: -1.4080000000000000e+03,
-1.4080000000000000e+03

DitherMode - Dither Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

DualTrace - TrDualTrace

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

FullScale - FS

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

InterleaveMode - Interleave Mode

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

IntgDisplay - INTG

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

NoOfAverages - No. of Averages

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 1, max: 1.67772e+07, current value: 11

NoOfPoints - No. of Points

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 32, max: 4096, current value: 512

list of discrete values: 32, 64, 128, 256, 512, 1024, 2048, 4096

RealDataName - RealDataName

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value:

RepetitiveMode - Repetitive Mode

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: True

list of discrete values: False, True

ResetDevice - Reset

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ScaleDivideBy2 - / 2

auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ScaleFactor - ScaleFactor

read-only, auxiliary

String type, # of dim.: 0, SPL Keyword: -

current value: * 1

ScaleTimes2 - * 2

read-only, auxiliary

Boolean type, # of dim.: 0, SPL Keyword: -

current value: False

list of discrete values: False, True

ScanTime - Scan Time

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0.128, max: 409600, current value: 2.048 us

TimeBase - Time Base

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 4, max: 100000, current value: 4.0 ns

list of discrete values: 4.0, 6.0, 8.0, 10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0, 2000.0, 5000.0, 10000.0, 20000.0, 50000.0, 100000.0

TriggerLevel - Trigger Level

auxiliary

Numeric type, # of dim.: 0, SPL Keyword: -

min: 0, max: 4095, current value: 2097

TriggerMode - Trigger Mode

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Auto

list of discrete values: Auto, Normal

TriggerSlope - Trigger Slope

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Pos

list of discrete values: Pos, Neg

TriggerSource - Trigger Source

auxiliary

Enumeration type, # of dim.: 0, SPL Keyword: -

current value: Ext ECL

list of discrete values: Int Ch1, Int Ch2, Ext ECL, Ext TTL