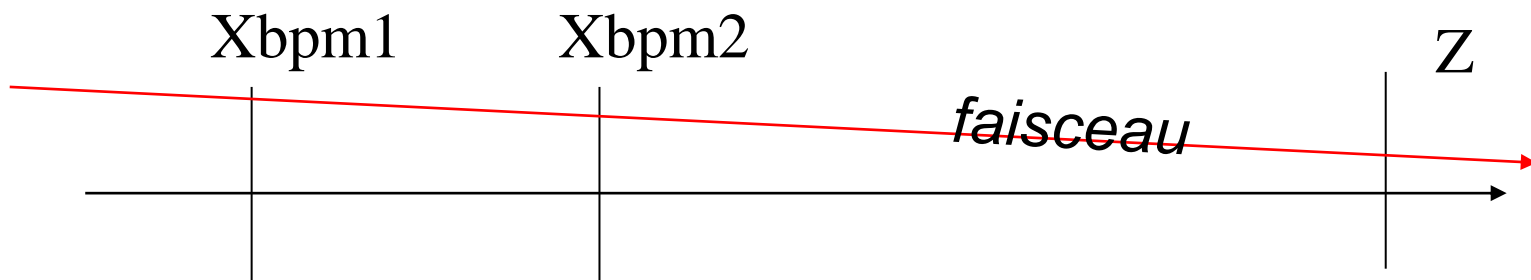


TangoParser

Ou

Device de calcul

- Fournir de **nouveaux attributs** TANGO qui soient le résultat de calculs à partir **d'attributs déjà existants**
- Exemple :
 - ➔ Je calcule la position du faisceau au Point Z en fonction des mesures effectuées sur les xbp1 et xbp2 (mesures déjà disponibles dans TANGO)



- Les attributs à lire dans le système Tango sont décrits par la propriété **AttributeNames**

➔ *Exemple :*

```
p1,TDL-D13-1/DG/XBPM.1/verticalPosition1  
p2,TDL-D13-1/DG/XBPM.1/verticalPosition2  
p3,TDL-D13-1/DG/XBPM.2/verticalPosition1  
p4,TDL-D13-1/DG/XBPM.2/verticalPosition2
```

➔ *Explication :*

- La « *variable* » *p1* va maintenant contenir le résultat de la lecture de l'attribut *TDL-D13-1/DG/XBPM.1/verticalPosition1*

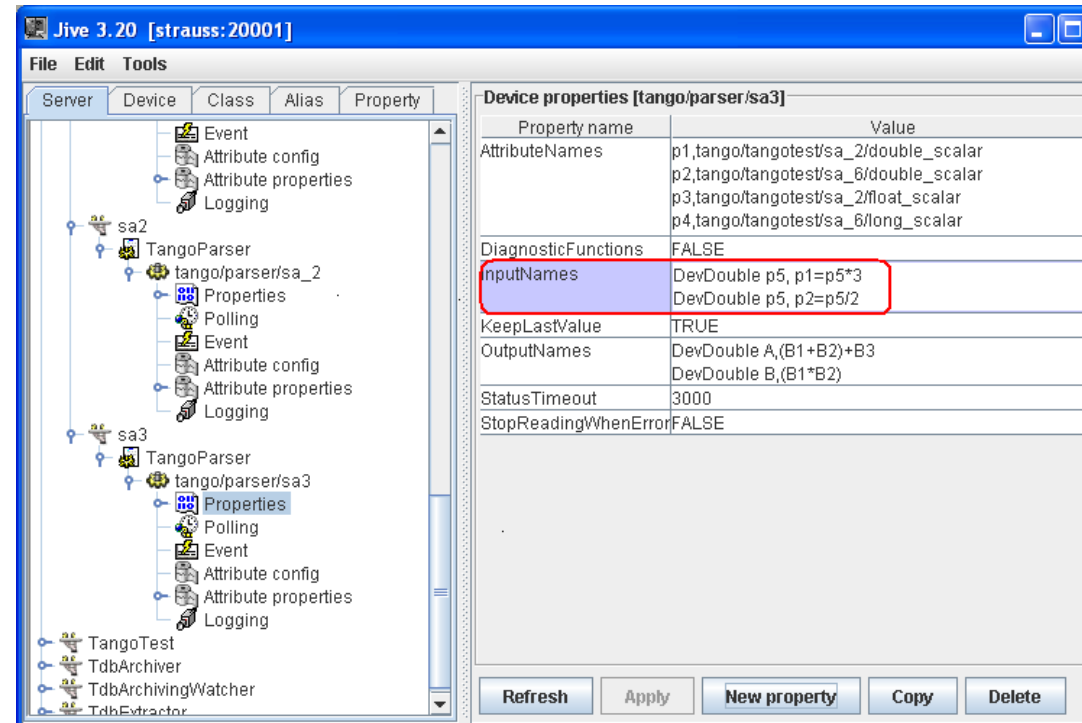
- Les nouveaux attributs dynamiques en écriture sont dans la propriété **InputNames**

— Exemple :

DevDouble p5, $p1=p5*3$
DevDouble p5, $p2=p5/2$

— Explication :

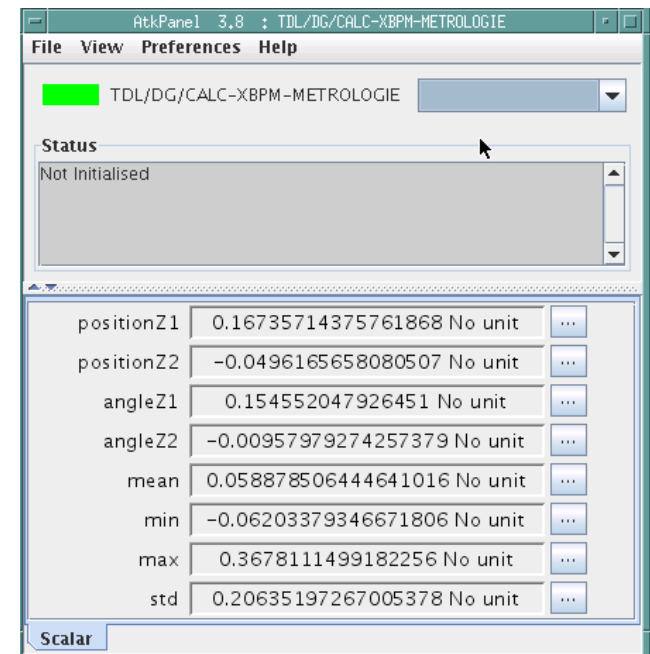
- Créer un nouveau attribut p5,
- A chaque fois qu'un client écrit dans l'attribut p5, le Parser écrit dans les attributs p1 et p2 les nouvelles valeurs calculées en fonction de p5 et selon les expressions définies dans InputNames.



- Les nouveaux attributs en lecture sont décrits par la propriété **OutputNames**

– *Exemple :*

```
DevDouble positionZ1, (p1+p3) / 2
DevDouble positionZ2, (p2+p4) / 2
DevDouble angleZ1, (p3-p1) / 3.030
DevDouble angleZ2, (p4-p2) / 3.030
```



– *Explication :*

- 4 nouveaux attributs de type *DevDouble* sont maintenant disponibles dans TANGO

- Les nouveaux attributs en écriture et en lecture sont décrits par la propriété **IONames**

– *Exemple :*

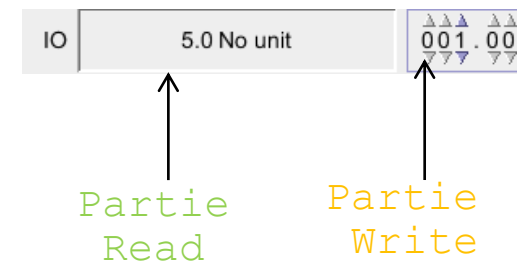
```
DevDouble IO;p1+p2;p1=IO+1;p2=IO+2
```



« ; » est le séparateur

– *Explication :*

- Un nouvel attribut est disponible. Sa partie read contiendra le résultat de « *p1+p2* » et lorsque l'on écrira dessus, le Parser écrira sur les attributs *p1* et *p2*.
- On ne peut définir qu'une seule expression (la première) pour la lecture. Et autant que souhaitées pour l'écriture.



- On peut créer de nouvelles variables (non définies dans AttributeNames) dans **InputNames** ou **IONames**:


- Exemple en créant les variables « aux1 » et « coef »:*

```
nbbin;aux1;aux1=nbbin
```

```
detcRW;10*(coef^(100/aux1));coef=detcRW
```

nbbin	10.0 No unit	010.00
detcRW	9.765625E7 No unit	005.00

- Explication :*
 - 2 attributs en Read/Write
 - Chaque écriture sur detcRW ou nbbin, modifie la partie read de detcRW.
- On peut réutiliser les variables créées dans **OuputNames**.

- On peut automatiser l'écriture des attributs définis dans **IONames** par la propriété **AutoInputProperties** :
 - Chaque ligne paramètre un attribut en précisant :
 - Son nom,
 - Son activation au démarrage du TangoParser
 - Sa fréquence d'actualisation, 
 - La précision, elle est optionnelle et ignorée dans le cas des types non numériques (booléen, chaînes de caractères).
Dans le cas des vecteurs, elle s'applique à chaque élément.
 - Pour chaque attribut IOName présent dans la liste des valeurs de **AutoInputProperties**, un attribut booléen en Read/Write est créé pour permettre son activation ou désactivation. Il n'est visible qu'en mode expert.

- Exemple :

```
valIO;false;0.5;0.01
```

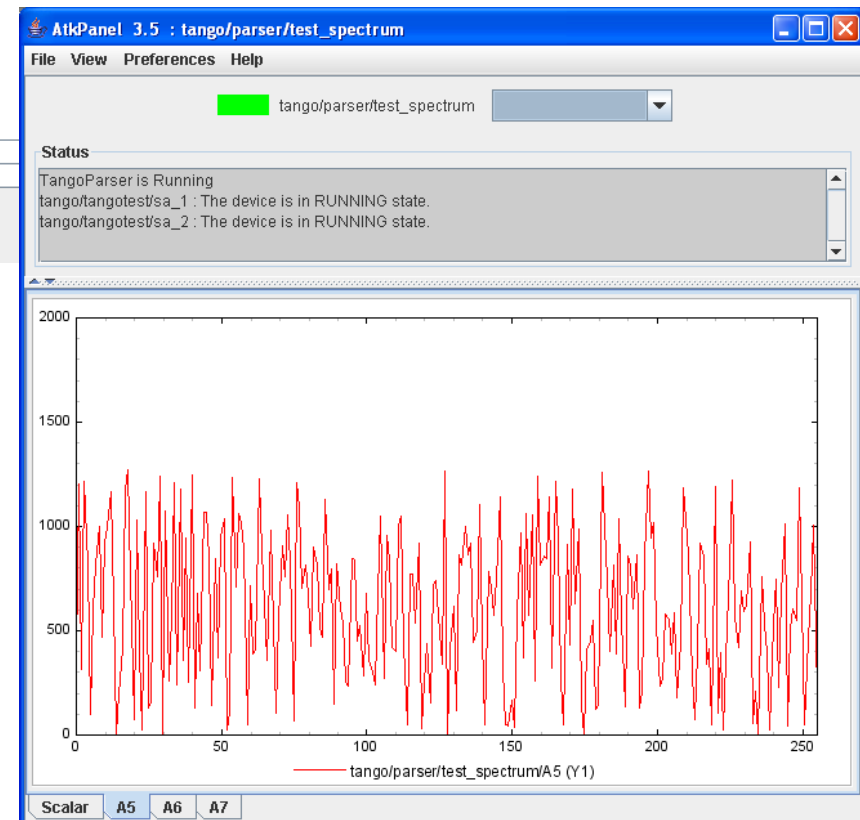
```
valIO2>true;1
```

- Explications

- La valeur présente dans la partie « output » de valIO, sera vérifiée toutes les 0,5 secondes et, si elle change de plus de 0,01 en valeur absolue, elle est écrite dans la partie « input ». Dès le démarrage du TangoParser, le processus de recopie est activé et une première recopie à lieu. Attention, ce mécanisme est dangereux.
- Pour valIO2, la recopie devra être activée via l'attribut booléen « valIO2AutoInputActivation ». Il sera vérifié toutes les secondes et recopié en cas de changement strict.

Device properties [tango/parser/test_spectrum]	
Property name	Value
AttributeNames	B1,tango/tangotest/sa_1/State B2,tango/tangotest/sa_2/double_spectrum_ro
DiagnosticFunctions	FALSE
InputNames	
KeepLastValue	TRUE
OutputNames	DevBoolean A1,B1>5 B1<11 DevBoolean A2,B1>11 DevState A3,B1 DevDouble A4,ysum(B2) SPECTRUM DevDouble A5,B2*5 SPECTRUM DevDouble A6,B2+B2 SPECTRUM DevDouble A7,fit(B2)
StatusTimeout	3000
StopReadingWhenError	FALSE

- Un nouvel attribut A5 est crée et est le résultat de la multiplication de B2 par un scalaire
 - B2 est un attribut spectrum READ-ONLY relu sur un Device Tango
- Un nouvel attribut A6 est la somme de 2 attributs spectrum



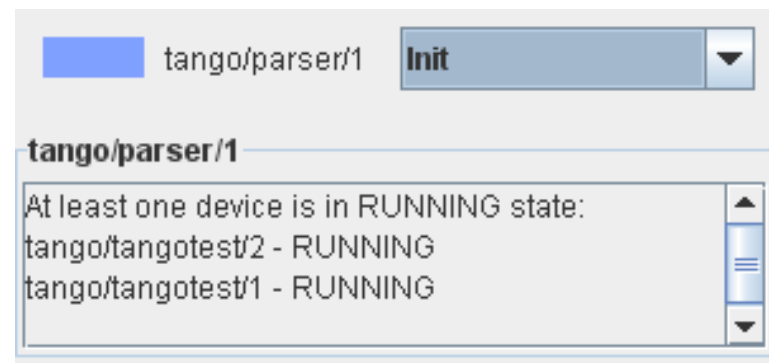
■ **Formats:** SCALAR, SPECTRUM.

➡ Optionnel. Valeur par défaut: SCALAR.

■ **Types:** DevBoolean, DevShort, DevLong, DevFloat, DevDouble, DevUShort, DevULong, DevString, State, DevUChar, DevLong64, DevULong64

➡ Optionnel. Valeur par défaut: DevDouble

- Le parser indique maintenant le State de chacun des Devices utilisés pour les calculs
- L'état du parser prend le state avec la priorité la plus élevée. (Voir propriété : PriorityList)



■ Dans l'attribut « log », donne la liste des erreurs.

➡ La 3eme colonne décrit l'erreur.

pichon/tools/test.parser

ERROR creating attributes, check 'log' attribute for more details

8	0	1	
9	22.03.29 - 15:23:34	INFO	/!\ building attribute "valD"
10	22.03.29 - 15:23:34	INFO	attribute "valD" is ready
11	22.03.29 - 15:23:34	ERROR	attribute "valD" not built. Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
12	22.03.29 - 15:23:34	ERROR	attribute "valC" not built. Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
13	22.03.29 - 15:23:34	ERROR	attribute "valB" not built. Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
14	22.03.29 - 15:23:34	ERROR	attribute "valA" not built. Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
15	22.03.29 - 15:23:34	ERROR	attribute "valIO" not built. Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
16	22.03.29 - 15:23:34	INFO	init of dynamic attributes done.
17	22.03.29 - 15:23:34	INFO	init device done
18	22.03.29 - 15:24:04	DEBUG	refresh value for attribute valIO
19	22.03.29 - 15:24:04	DEBUG	read 20.0 on valIO
20	22.03.29 - 15:24:04	DEBUG	refresh value for attribute valA
21	22.03.29 - 15:24:04	ERROR	exception message is: Encountered "<EOF>" at line 1, column 11. Was expecting one of: "+" ... "-" ... "!" ... <STRING_LITERAL> ... <INTEGER_LITERAL> ... <F
22	22.03.29 - 15:24:04	DEBUG	refresh value for attribute valB
23	22.03.29 - 15:24:04	ERROR	exception message is: Could not evaluate val3: variable not set
24	22.03.29 - 15:24:04	DEBUG	refresh value for attribute valC
25			

Scalar log

- Autres propriétés :

- ScanMode:
 - TRUE : lecture des attributs et de l'état de façon synchrone
 - FALSE : lecture de façon asynchrone
- PeriodRefresh
 - Délai de rafraîchissement des attributs et état en mode asynchrone en ms.
- DiagnosticFunctions
 - TRUE : le parser ajoute le calcul des fonctions MIN, MAX, MEAN, STD sur les AttributesNames
- MovingState
 - indique au ScanServer l'état auquel il doit s'attendre lorsque le device est en « moving »
- PriorityList
 - Liste des états selon leur priorité. Cela permet de composer les états des différents devices appelés et de remonter le plus important.
- AutoInputProperties
 - Définit les propriétés d'écriture automatique des attributs IONames

Property name	Value
AttributeNames	val1,tango/test/test.pub/val1 val2,tango/test/test.pub/val2 val3,tango/test/test.pub/val3 val4,tango/test/test.pub/val4 valSpect,tango/test/test.pub/valSpect
autoInputProperties	valIO,false;0.5;0.01
DiagnosticFunctions	FALSE
InputNames	SPECTRUM DevDouble valIn, valSpect=valIn
IONames	DevDouble valIO; val1*val2; val3=valIO/val2; val4=valIO/val1
KeepLastValue	
MovingState	
OutputNames	DevDouble valA,val1*val2 DevDouble valB,val2*val3 DevDouble valC,valA*valB DevDouble valD,val4*valC
PriorityList	FAULT,2 ALARM,1
ScanMode	FALSE
StatusTimeout	3000
StopReadingWhenError	FALSE

Affichage de
l'expression de
l'attribut dans le
champ description

Attribute property editor

Identification

Device: tango/parser/1

Attribute: testRW2

83.34602205026455 No 000.00

Properties

Label: testRW2

Minimum value: Not specified Maximum value: Not specified

Minimum alarm: Not specified Maximum alarm: Not specified

Format: Unit: No unit

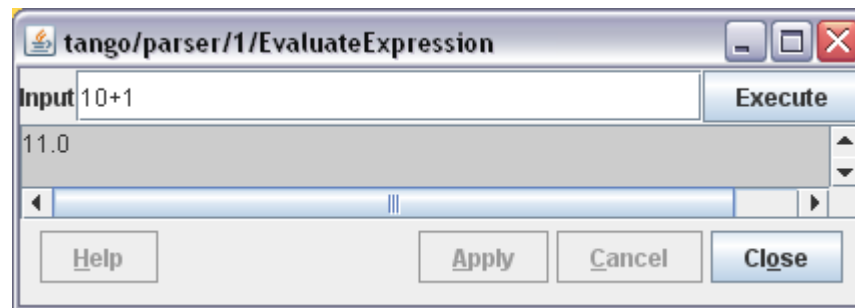
Min. warning: Not specified Max. warning: Not specified

Delta t(ms): Not specified Delta Val: Not specified

expression read: a+b
expressions write: [a=testRW2, b=testRW2*2]

Apply change Information Dismiss

- La commande EvaluateExpression retourne l'évaluation de l'expression passée en argument d'entrée.



Les fonctions mathématiques disponibles: Scalaires : Opérateurs

Power	\wedge
Boolean Not	!
Unary Plus, Unary Minus	+x, -x
Dot product, cross product	., ^^
Modulus	%
Division	/
Multiplication	*
Addition, Subtraction	+, -
Less or Equal, More or Equal	<=, >=
Less Than, Greater Than	<, >
Not Equal, Equal	!=, ==
Boolean And	&&
Boolean Or	
Assignment	=

mul sinh boolextract add asin stringlen mod cos
left floor cosh binom log mean toHex sin sum
tanh atan2 max sub arg asinh vcos ceil polar pow
fromDec abs atan im ln rand acosh vlog str if
substring fft vsin sqrt re cmod toBase tan std
conj right atanh round fromHex div acos toDec
min complex exp fromBase

Cf transparents suivants pour la documentation.

Constantes disponibles:

- ➡ i: (0.0, 1.0)
- ➡ e: 2.718281828459045
- ➡ pi: 3.141592653589793

Trigonometric Functions

All functions accept arguments of the `Double` and `Complex` only accepts `Double` arguments.

Description	Function Name
Sine	<code>sin(x)</code>
Cosine	<code>cos(x)</code>
Tangent	<code>tan(x)</code>
Arc Sine ²	<code>asin(x)</code>
Arc Cosine ²	<code>acos(x)</code>
Arc Tangent	<code>atan(x)</code>
Arc Tan with 2 parameters	<code>atan2(y, x)</code>
Secant	<code>sec(x)</code>
Cosecant	<code>cosec(x)</code>
Co-tangent	<code>cot(x)</code>
Hyperbolic Sine	<code>sinh(x)</code>
Hyperbolic Cosine	<code>cosh(x)</code>
Hyperbolic Tangent	<code>tanh(x)</code>
Inverse Hyperbolic Sine	<code>asinh(x)</code>
Inverse Hyperbolic Cosine ¹	<code>acosh(x)</code>
Inverse Hyperbolic Tangent ¹	<code>atanh(x)</code>

Les fonctions mathématiques disponibles: Scalaires: Fonctions

Log and Exponential Functions

All functions accept arguments of the Double and Complex

Description	Function Name
Natural Logarithm ¹	ln(x)
Logarithm base 10 ¹	log(x)
Logarithm base 2 ¹	lg(x)
Exponential (e ^x)	exp(x)
Power ¹	pow(x)

Statistical Functions

All functions accept either a vector (e.g. min([1,2,3])) or min(1,2,3)).

Description	Function Name
Average	avg(x1,x2,x3,...)
Minimum	min(x1,x2,x3,...)
Maximum	max(x1,x2,x3,...)

Rounding Functions

Description	Function Name
Round	round(x), round(x, p)
Floor	floor(x)
Ceiling	ceil(x)

Miscellaneous Functions

Description	Function Name
If	if(cond, trueval, falseval)
Str (convert number to string)	str(x)
Absolute Value / Magnitude	abs(x)
Random number (between 0 and 1)	rand()
Modulus	mod(x,y) = x % y
Square Root ¹	sqrt(x)
Sum	sum(x,y,...)
Binomial coefficients	binom(n, i)
Signum (-1,0,1 depending on sign of argument)	signum(x)

ToBase	Converts numbers to a string in a given base.
FromBase	Converts a string in a given base to numbers.

Complex Functions

Description	Function Name	C
Real Component	re(c)	R
Imaginary Component	im(c)	I
Complex Modulus (Absolute Value)	cmod(c)	A
Argument (Angle of complex value, in radians)	arg(c)	A
Complex conjugate	conj(c)	C
Complex, constructs a complex number from real and imaginary parts	complex(x, y)	C
Polar, constructs a complex number from modulus and argument	polar(r, theta)	P

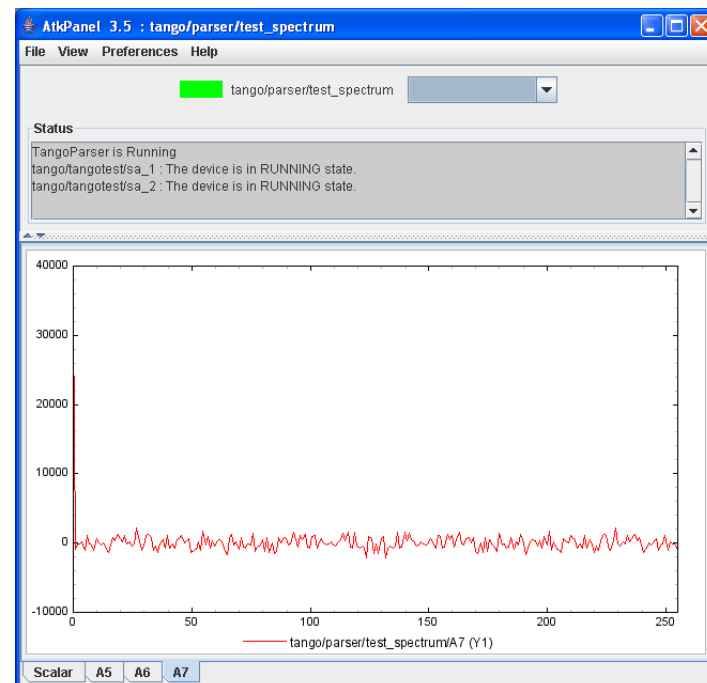
Name	Description	Examples
length	Finds the length of a vector or the number of elements in a matrix or tensor.	<code>length(5)==1, length([1,2,3])==3, length([[1,2], [3,4]])==4</code>
size	Finds the size of a vector, matrix or tensor.	<code>size(5)==1, size([1,2,3])==3, size([[1,2], [3,4], [5,6]])==[2,3]</code>
det	The determinant of a square matrix.	<code>det([[1,2], [3,4]])==-2</code>
trace	The trace of a square matrix.	<code>trace([[1,2], [3,4]])==5</code>
vsum	The sum of elements in a vector, matrix or tensor.	<code>vsum([1,2,3])==6, vsum([[1,2], [3,4]])==10</code>
trans	The transpose of a matrix.	<code>trans([[1,2], [3,4]])==[[1,3], [2,4]]</code>
getdiag	Extracts the diagonal from a square matrix.	<code>getdiag([[1,2], [3,4]])==[1,4]</code>
diag	Generates a square matrix with a given diagonal.	<code>diag([1,2,3])==[[1,0,0], [0,2,0], [0,0,3]]</code>

GenMat	Generates vectors and matrixes	<p>First argument specifies size of the vector (3) and matrixes ([2,2]). Second argument is formula for each element, constants (1), functions (rand()), Third argument (if present) is list of variables used in formula. For vectors a single variable is specified whos value runs from 1 to number of elements. For matrixes a two variable list ([ii,jj]) is specified whos value are the column and row indicies.</p> <p><code>GenMat(3,1) -> [1,1,1]</code> <code>GenMat(3,ii,ii) -> [1,2,3]</code> <code>GenMat(3,rand()) -> [0.343,0.974,0.567]</code> <code>GenMat([2,2],ii+jj,[ii,jj]) -> [[2,3],[3,4]]</code></p>
ele	Extracts an element from a vector, matrix or tensor.	<code>ele([1,2,3],2)==2, ele([[1,2], [3,4]], [1,2])==2</code>

The Map function applies a function to each element of a vector or matrix

Device properties [tango/parser/test_spectrum]		
Property name		Value
AttributeNames	B1,tango/tangotest/sa_1/State	
	B2,tango/tangotest/sa_2/double_spectrum_ro	
DiagnosticFunctions	FALSE	
InputNames		
KeepLastValue	TRUE	
OutputNames	DevBoolean A1,B1>5 B1<11	
	DevBoolean A2,B1>11	
	DevState A3,B1	
	DevDouble A4,ysum(B2)	
	SPECTRUM DevDouble A5,B2*5	
	SPECTRUM DevDouble A6,B2+B2	
StatusTimeout	3000	
StopReadingWhenError	FALSE	

- Un nouvel attribut A7 est crée et est le résultat de la **Transformée de Fourier** de l'attribut spectrum B2



■ AttributeNames:

➔ B2,tango/tangotest/sa_2/double_spectrum_ro

■ OutputNames:

➔ SPECTRUM DevDouble A1,vlog(B2)

▶ Chaque point du spectrum A1 contient le log du point de même indice du spectrum B2

➔ SPECTRUM DevDouble A2,vcos(B2)

▶ Chaque point du spectrum A2 contient le cos du point de même indice du spectrum B2

➔ SPECTRUM DevDouble A3,vsin(B2)

▶ Chaque point du spectrum A3 contient le sin du point de même indice du spectrum B2

➔ SPECTRUM DevDouble A4,add(B2,5)

▶ Chaque point du spectrum A4 contient la somme du scalaire de valeur 5 et du point de même indice du spectrum B2

➔ SPECTRUM DevDouble A5,sub(B2,3)

▶ Chaque point du spectrum A5 contient la différence du point de même indice du spectrum B2 et du scalaire de valeur 3

- Introduire un nouveau sensor pour les Scans tel que :
 - ➡ le rapport I/I_0
 - ➡ Le rapport $\log(I/I_0)$
 - ➡ etc ..
- Définir de nouvelles conditions d'alarme sur le système de contrôle

■ Scalaires :

➡ <http://www.singularsys.com/jep/doc/html/index.html>

■ Spectrum

➡ <http://www.singsurf.org/djep/html/djep/VectorJep.html>