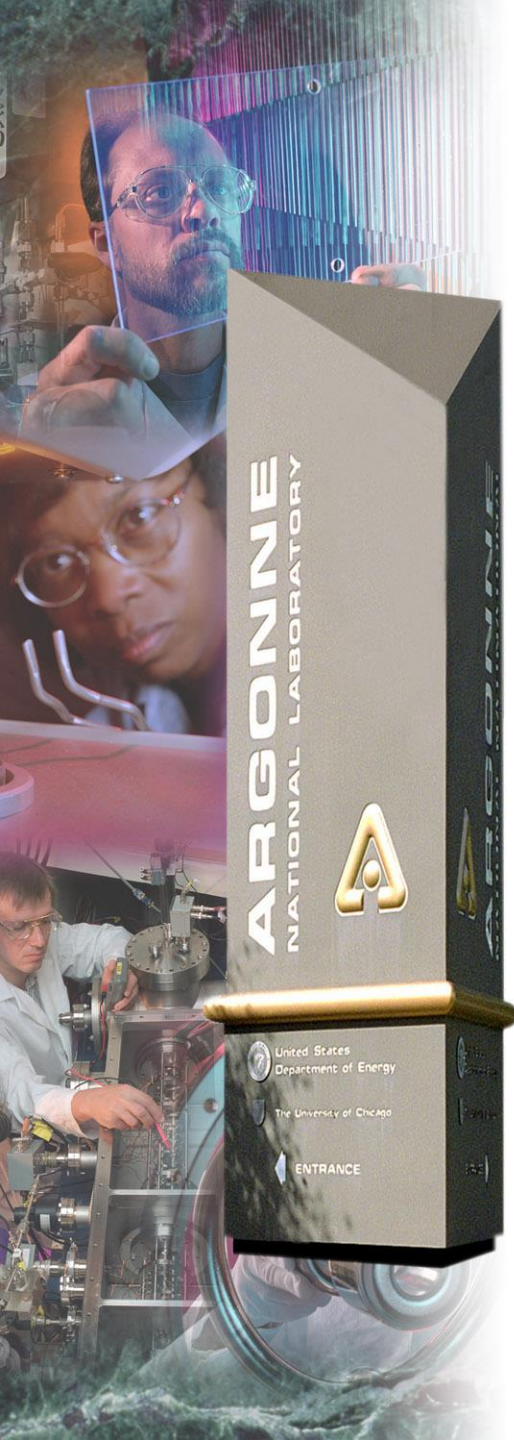


The synApps calc module

Tim Mooney



Argonne National Laboratory



*A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago*



calc overview

Support for **evaluating expressions** entered at run time

- **EPICS record types:**

- **acalcout** – like *calcout*, but also supports array expressions; user can specify wait-for-completion.
- **scalcout** – like *calcout*, but also supports string expressions; user can specify wait-for-completion.
- **swait** – like *calcout*, but uses recDynLink (no “PP MS” link attributes), and waits for completion.
- **transform** – like 16 *calcout* records that share a PV data pool
- **sseq** – like *seq*, in base, but can get and put strings; user can specify wait-for-completion.

- **Other code:**

- interpolation routines for *aSub* record
- averaging routines for *sub* record
- *sseq*-record editor

calc databases and display files

- Databases, display files for run-time programming
 - userCalc, userCalcout
 - userStringCalc
 - userArrayCalc
 - userTransform
 - userStringSeq
 - userAve
 - interpolation
- Examples of ALL calc expressions can be found in synApps calc*Example displays

```

calcExamples.adl
-----
Algebraic functions/operators
ABS(a)      LOG(a)      SQR(a)      a+b
CEIL(a)     LOGE(a)     SQRT(a)     a/b
EXP(a)      MAX(a,b,...) a^b
FLOOR(a)    MIN(a,b,...) a+b
INT(a)      NINT(a)     a+b         a>?b
LN(a)       NOT(a)      a-b         a<?b

Trigonometric functions/operators
ACOS(a)     COS(a)      SINH(a)
ATAN(a)     COSH(a)     TAN(a)
ATAN2(a,b)  SIN(a)      TANH(a)

Relational operators
a>b         a<b         a!=b        a=b
a>?b       a<?b       a#b         a==b

Bitwise/Logical operators
a|b         a&b         a XOR b     a>>b
a OR b     a AND b     ~a          a<<b
a|?b       a&?b       a^b

Misc operators & symbols
(a)         a?b:c      UNTIL(a:=a-1;a<1)@a
RNDM       NRNDM      (a:=1;b:=2;c)  a:=b

String functions/operators
BYTE(aa)   PRINTF('%f',aa) aa+bb      aa-bb
DBL(aa)   SSCANF(aa,'%lf') aa>=bb     aa<=bb
INT(aa)   $S(aa,'%lf') aa>?bb     aa<?bb
MAX(aa,bb,...) READ(aa,'%ld') aa|=bb
MIN(aa,bb,...) WRITE('%ld',a) aa>bb      aa<bb
NINT(aa)  aa[bb,cc] aa==bb     aa!=bb
$( '%f',aa) aa[1,3] aa|-bb      aa|_bb
$(aa)     TR_ESC(aa) $(aa)     ESC(aa)
CRC16(aa) ADD_XOR3(aa) XOR3(aa) STR(aa)
MODBUS(aa) RMODBUS(aa) LRC(aa) 'abc'
$( '%lf',a) $(R(aa,'%ld')) LEN(aa)  @aa
aa>>a     aa<<a

NOTE: can use single or double quotes at runtime
but only single quotes in a static database.
'a', 'b', etc. are numeric arguments.
'aa', 'bb', etc. are string arguments.

Array functions/operators
AMAX(aa)  DERIV(aa)  NDERIV(aa,a) @aa
AMIN(aa)  FITPOLY(aa) NSMOD(aa,a) aa>>10
ARNDM    FITMPOLY(aa,bb) SMOD(aa) aa<<10
ARR(a)   FUHM(aa)  STD(aa)      CUM(AA)
AVG(aa)  IX        SUM(aa)      IXXMAX(AA)
IXMIN(AA) IXZ(AA)  IXNZ(AA)

Items in blue are useable in transform and sCalcout
records, and include string functions.
Items in red are useable in aCalcout records.
    
```

```

calcMiscExamples.adl
-----
TOP  ALG  TRIG  REL  BIT  MISC  STR
-----
Miscellaneous
-----
Grouping
(a)
CAN'T USE '[' or '{'
(A)ANDB
SAME AS A&B, BUT NOTE...
A&ANDB
SCALCOUT RECORD WILL PARSE 'AA'
AS A STRING VARIABLE

'?:' (if-else) operator
A?B:C
IF (A!=0) RESULT IS B, ELSE C

Array expressions
-----
@0
GET VARIABLE 0 (<@0=A, @1=B, ..)
@A+3
GET VARIABLE A; ADD 3 TO IT
@(A+3)
GET VARIABLE A+3
@@0
GET STRING VAR 0 (<@@0=AA, ..)

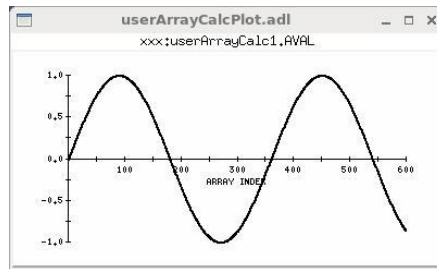
Random number
-----
RNDM
RANDOM VALUE IN RANGE [0,1]
    
```

Array-calc (acalcout) record

- **Calcout record plus:**
- **Array fields AA...LL**
- **Array result AVAL**
- **Additional output option “Never”**
- **Can wait for completion**
- **Array expressions:**

Assume $aa=(1,2,3,4,5)$

- $aa[2,4] \rightarrow (2,3,4)$
- $ix \rightarrow (0,1,2,3,\dots)$
- $\sin(ix*d2r) \rightarrow$



userArrayCalc.adl (xxx:userArrayCalc8)

Passive EVT 0 PROC #DIG 5 NUSE 102 ENABLE

DOUBLE VARIABLES	PV NAME	VALUE
A	xxx:traj1:StartPulses NPP NMS	0.00000
B	xxx:traj1:EndPulses NPP NMS	100.00000
C	xxx:m1.MRES NPP NMS	0.00100
D		0.00000

ARRAY VARIABLES	PV NAME	VALUE
AA	xxx:userArrayCalc5.AVAL NPP N	0.00000
BB		0.00000

HELP	CALC (CALCULATION)	RESULT
?	$(aa[a,b]-bb[a,b])/c$	0.00000
	ARRAY RESULT	0.00000

OCAL (OUTPUT CALCULATION)	RESULT	
?	0.00000	
	ARRAY RESULT	0.00000

DELAY 0.000 OUTPUT EVENT# 0 Every Time Use CALC

Continue normally IVOV 0.000 OUTPUT PV NAME

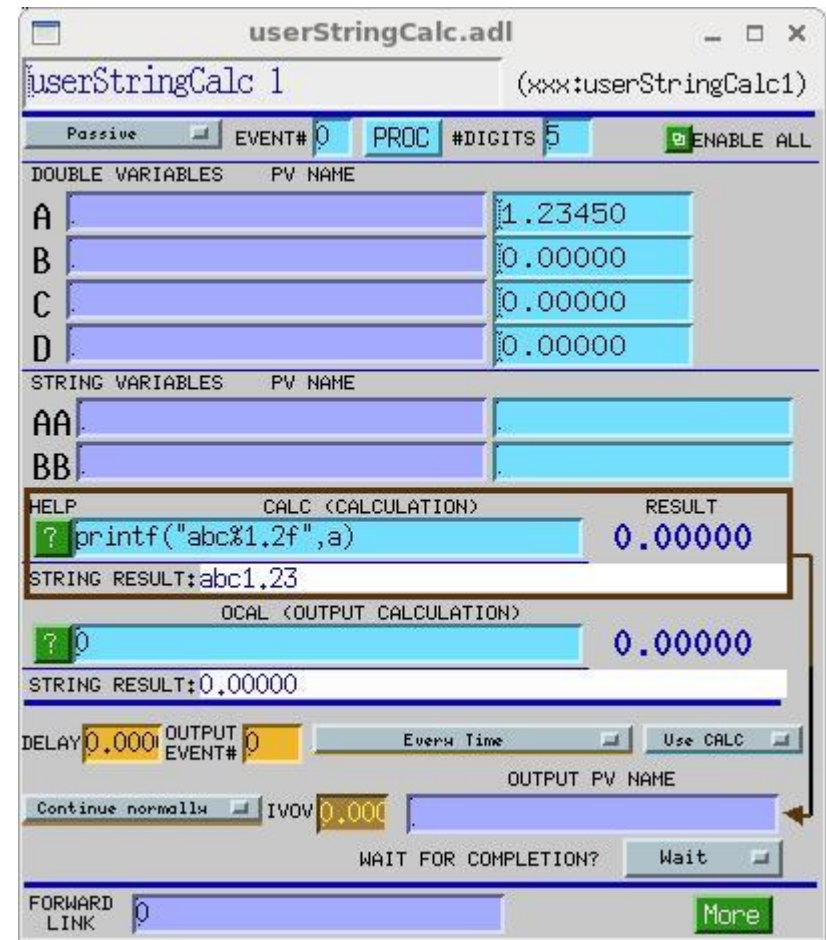
WAIT FOR COMPLETION? NoWait

FORWARD LINK Less More

String-calc (scalcout) record

- Calcout record plus:
- String fields AA...LL
- String result SVAL
- Additional output option “Never”
- Can wait for completion
- String expressions:

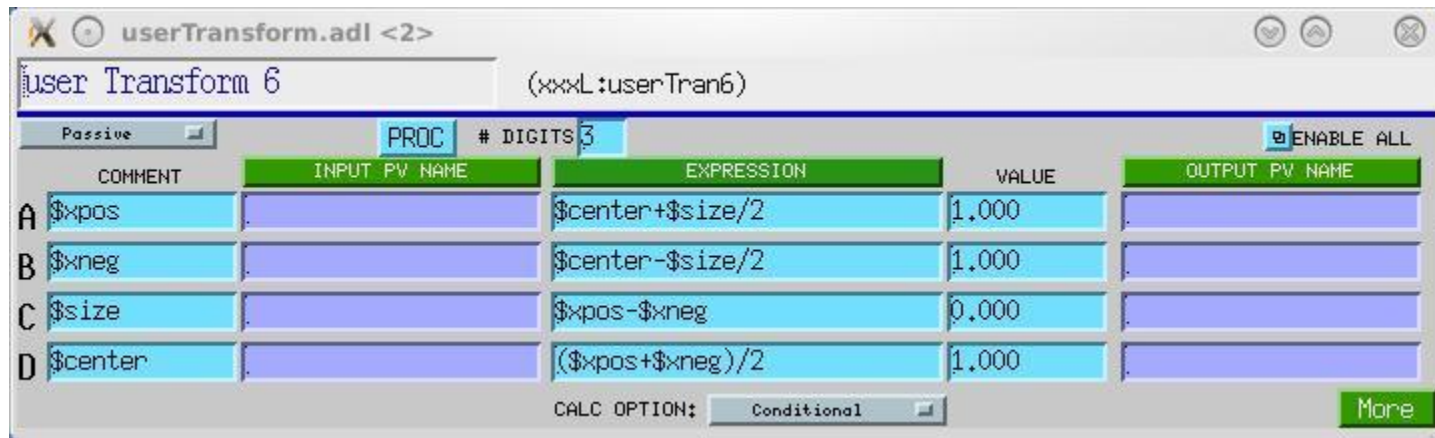
Expression	Result
<code>printf("abc%1.2f", a)</code>	“abc1.23”
<code>"abcdef"[2,4]</code>	“cde”
<code>“m1.VAL”{“.VAL”,“.EGU”}</code>	“m1.EGU”



The screenshot shows the control panel for a `userStringCalc` record. It includes fields for `EVENT#` (0), `PROC`, and `#DIGITS` (5). There are sections for `DOUBLE VARIABLES` (A, B, C, D) and `STRING VARIABLES` (AA, BB). A `HELP` section shows a calculation: `printf("abc%1.2f", a)` resulting in `0.00000` and a `STRING RESULT` of `abc1.23`. Below that, an `OCAL` section shows a calculation of `0` resulting in `0.00000` and a `STRING RESULT` of `0.00000`. At the bottom, there are controls for `DELAY` (0.000), `OUTPUT EVENT#` (0), `OUTPUT PV NAME`, and a `WAIT FOR COMPLETION?` checkbox (checked).

Transform record

- Like 16 calcout records, sharing a pool of input variables
- Original use was for coordinate transformations
- **Conditional expression evaluation:**
 - If field **A** was written to, don't execute expression CLCA
 - Each expression uses results of previous expression
- **Example: equations for slit**



COMMENT	INPUT PV NAME	EXPRESSION	VALUE	OUTPUT PV NAME
A \$xpos		\$center+\$size/2	1.000	
B \$xneg		\$center-\$size/2	1.000	
C \$size		\$xpos-\$xneg	0.000	
D \$center		(\$xpos+\$xneg)/2	1.000	

CALC OPTION: Conditional More

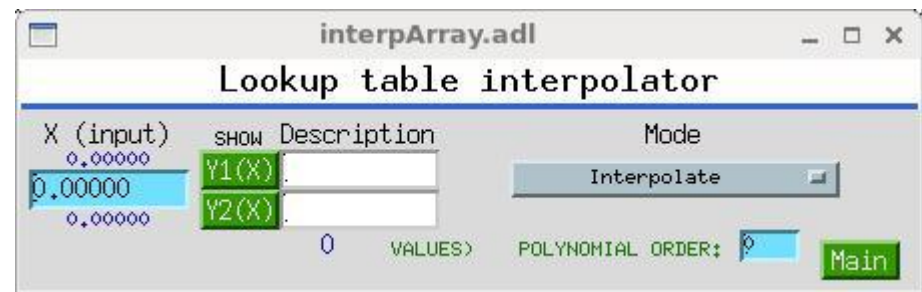
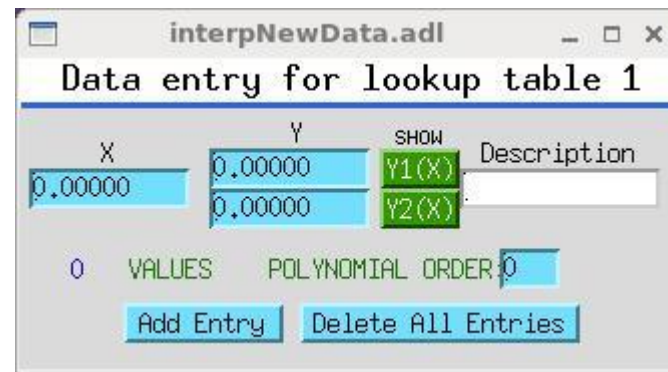
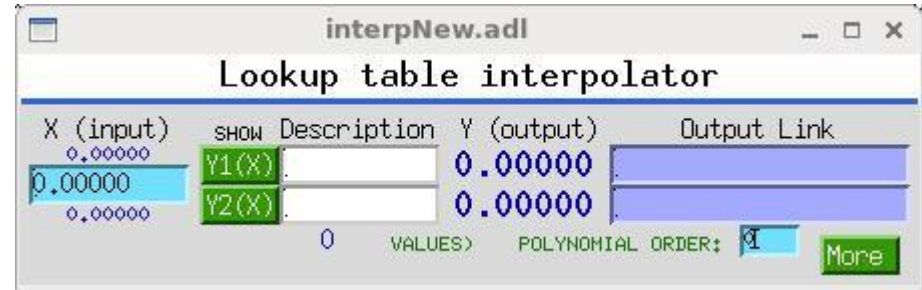
Sseq (String-Sequence) record

- Like seq record, but can read/write strings or numbers
- Can wait for completion (MUST use CA link)
- Example: driving caputRecorder playback from EPICS:

	INPUT PV NAME	LINK HELP	DELAY	STRING VALUE	NUMERIC VALUE	OUTPUT PV NAME	LINK HELP	WAIT FOR COMPLETION?
01			0.0000	motorscan	0.00000	xxx:caputRecorderMacr		Wait
02			0.0000	'm3'	0.00000	xxx:caputRecorderAng1		NoWait
03			0.0000		0.00000			NoWait
04			0.0000	Do	0.00000	xxx:caputRecorderExec		Wait
						FORWARD LINK		?

Interp database

- Lagrange n^{th} -order interpolation
- One input, two outputs
- Can load interpolation table, or build it point by point
- Array mode: accept input array, output interpolated array



userAve data averaging

- Continuous or one-shot
- “Acquire” calls back when specified average has completed
- Algorithms:
 - AVERAGE
 - *Straight average*
 - FIT-LINE
 - *Fit to a line, return estimate of current value from line fit*
 - AUTO
 - *Vary smoothly between AVERAGE and FIT-LINE, using correlation coefficient*

