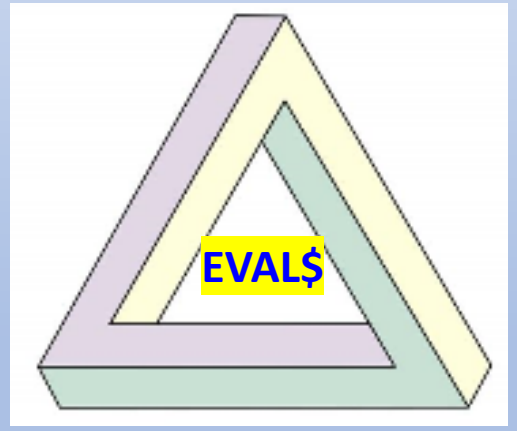
Towards HP-41 *Visual FOCAL++*

Teaching more New Tricks to an Ol'Dog

Ángel M. Martin – October 29, 2022

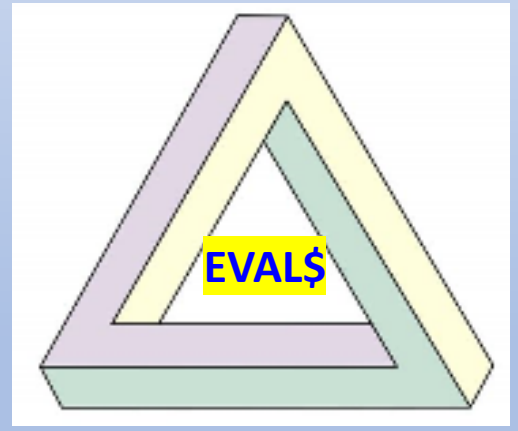
EVALΣ



EVALP

EVAL?

^FRMLA



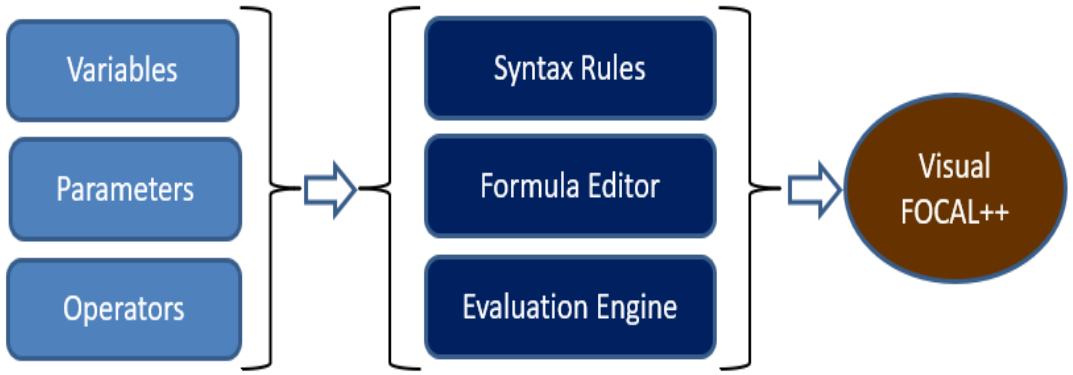
MVARS

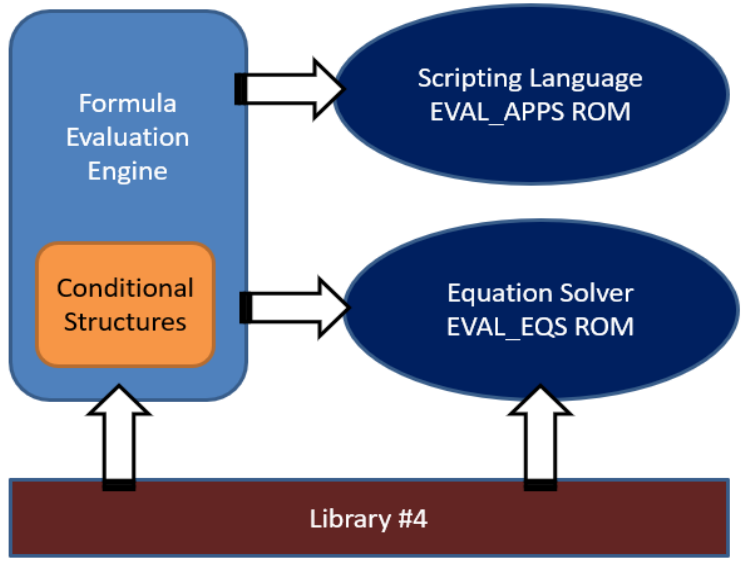
SOLVER

Presentation ~~Agenda~~ Sketches: Allschwil 2022

- Formula Evaluation Engine
- Conditional Structures
- Scripting Language
- From Solve to Solver: Equation Libraries

- CODA: Double-Down Module
- CODA II: X-Mem "Twin" Mods
- Acknowledgments





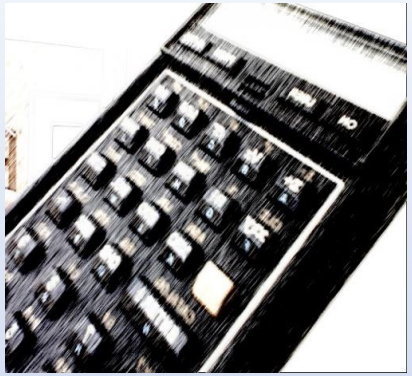
Acknowledgements (a.k.a. the usual suspects…)

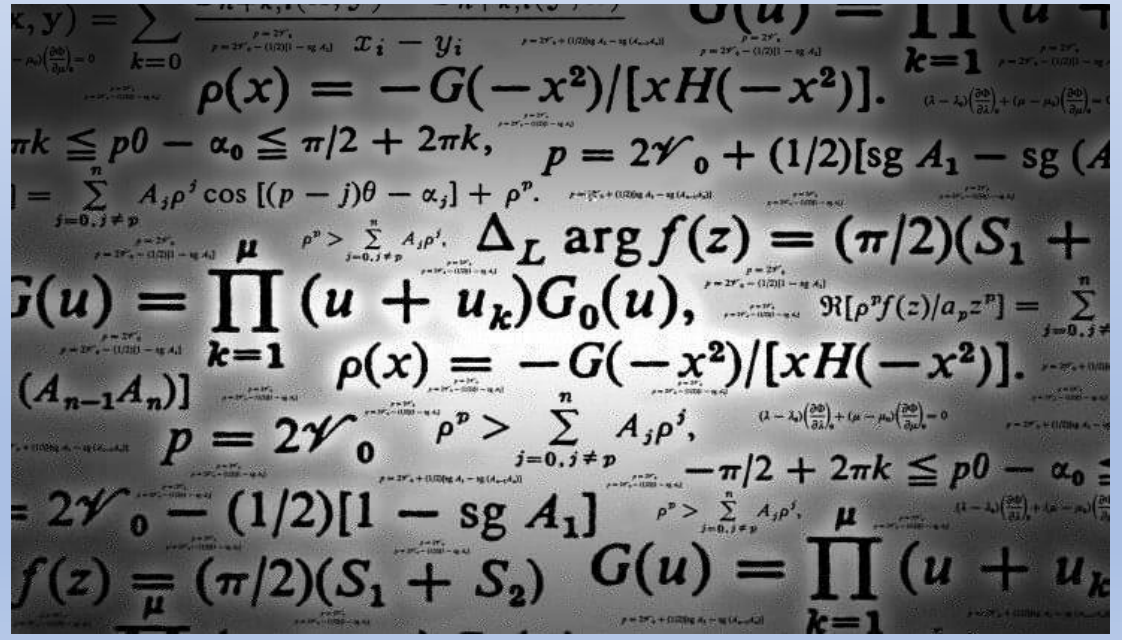
"Standing on the Shoulders of Giants"

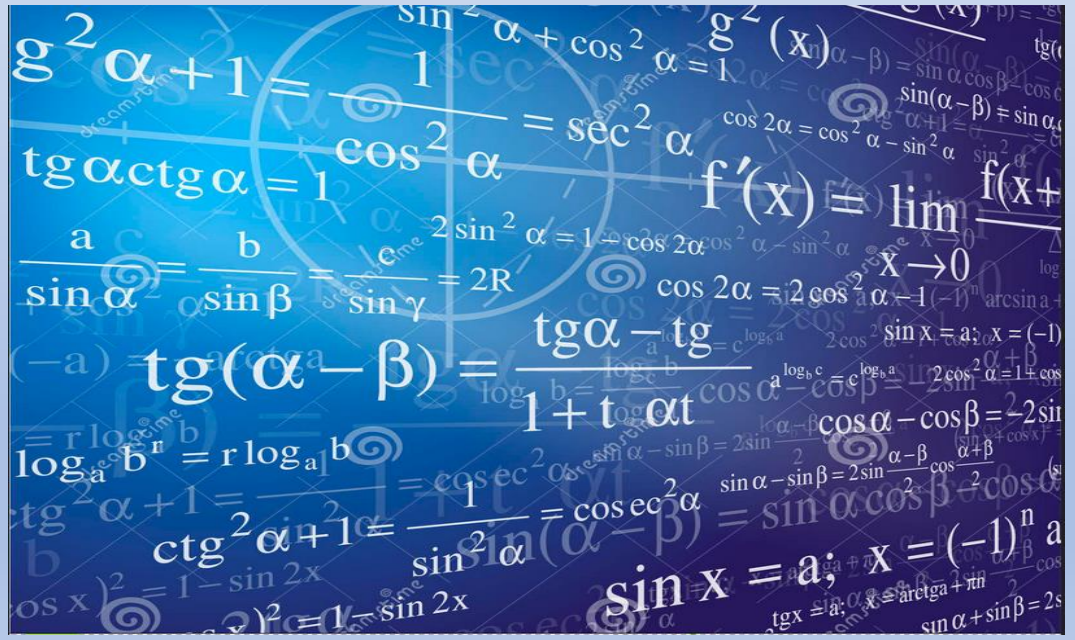
- Obviously to HP "of yore", for the HP-41 System and its *phenomenal* OS MCODE
- W&W GmbH, VM Electronics, Redshift s/w – publishers of CCD Module, HEPAX & AECROM
- Wilson "Bill" Holes and Nelson F. Crowle, head & heart of the AECROM – *"Viva Las Vegas!"*
- Doug Wilder, author of ROMED, Jump Distances and GOSUB/GOTO Decoders
- Håkan Thörngren, author of RAMED & several XM-Utils. *Takk!*
- Greg McClure co-authored the 16C Emulator and Formula Evaluation Modules, true "*partner-in-crime*" for numerous MCODE projects and explorations

- David Ingebretsen, program Sponsor and advisor
- Jean-Marc Baillard provided inextinguishable Math expertise and co-authored Math ROMs
- Mark Fleming authored the Equation Library (FOCAL version)
- Monte Dalrymple for creating the fabulous 41CL board
- Diego Díaz, Meindert Kuipers, Jean-François Garnier – *guys, your gizmos rock!*
- Sylvain Côté, always glad to share his encyclopedic knowledge
- Poul Kaarup, author of the PK-Collection w/ innovative apps
- *… and many, many others that would take too long to include!*

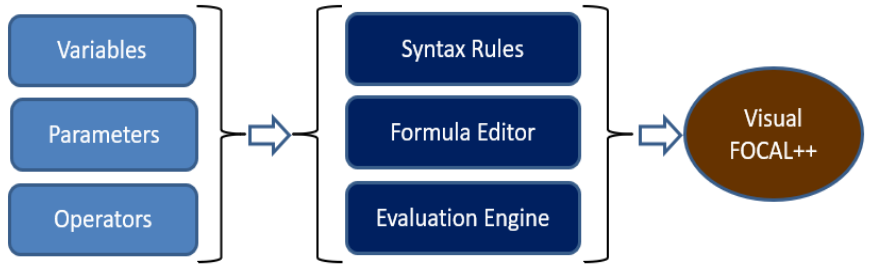
Formula Evaluation & Equation Solver ROMs

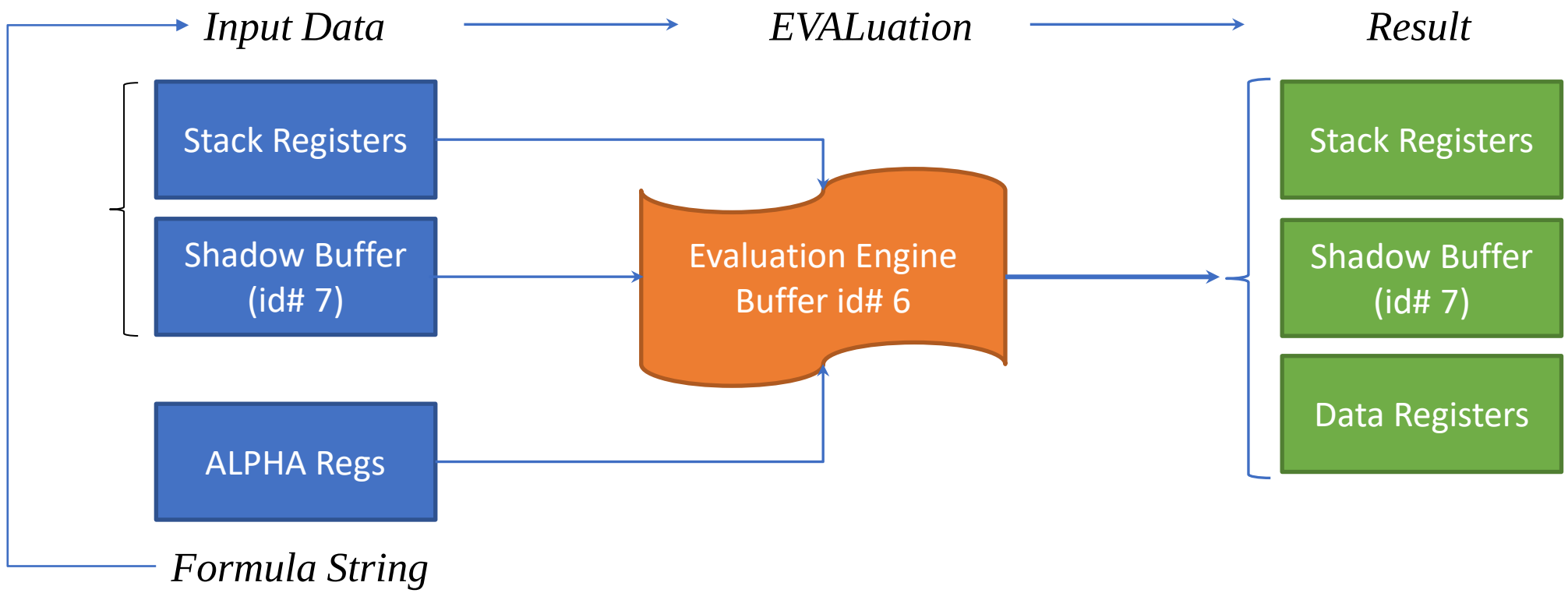






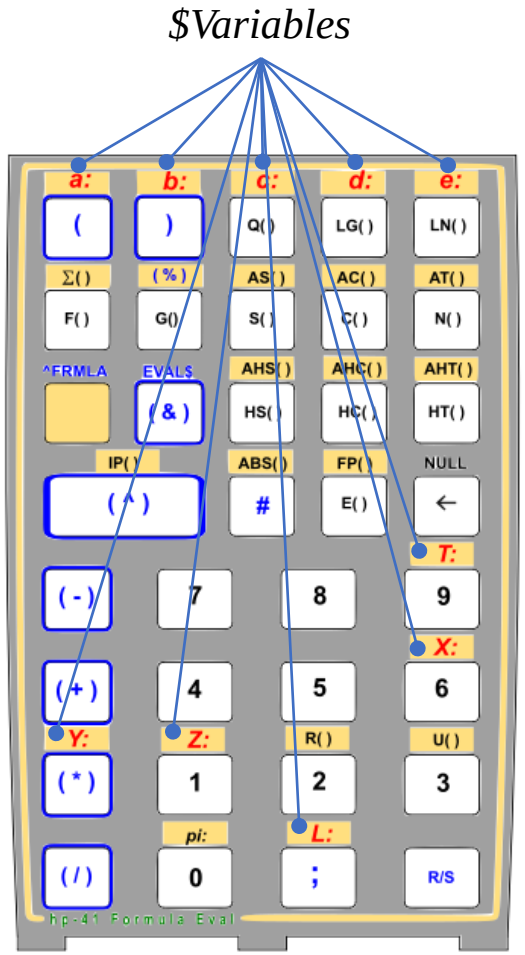
Formula Evaluation Operation
Block Diagram



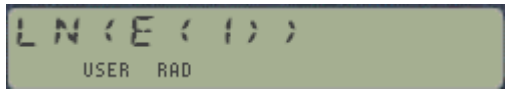


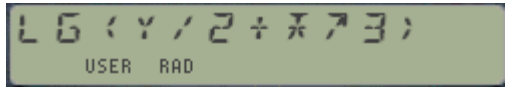
Formula Evaluation Engine [FRML]

a.k.a "Giving TI a run for its money"



- Formula Editor in ALPHA (24-char limit)
  - Uses Buffer #6 (LIFO, 16-Registers in size)
- Extended via Chained Evaluation
- Can also use ASCII Files in Script Language
  - GTO$, XEQ$, DO/WHILE$, FOR/NEXT$
- Rich, 13-digit Math function library
- Direct Stack-Registers, π, constants, and Custom Variables {a-f}
- Utility functions for data register transfer:
  - **LET=, GET=, SWAP= ; STO$/RCL$; ST<>RG ; A-PM**
- Full-expression Comparison Tests
- Sums, Series & Products Evaluation

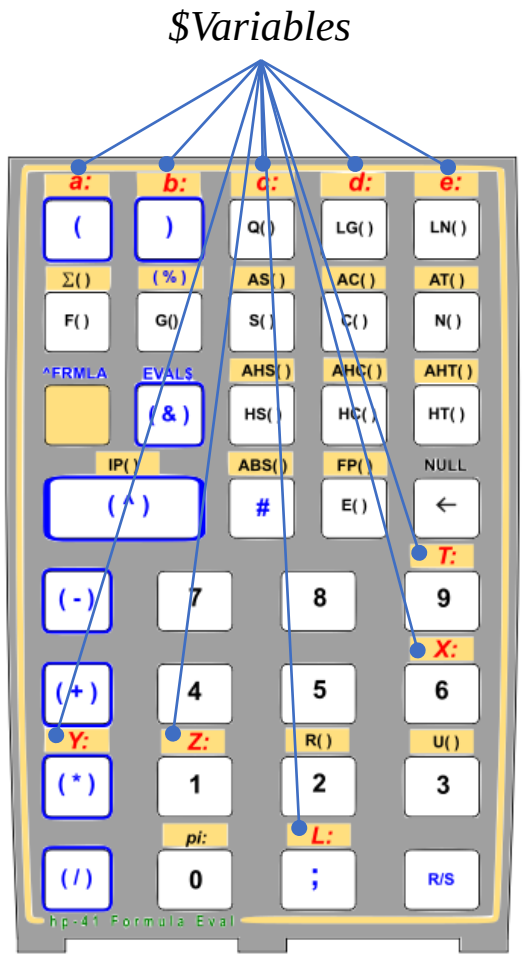


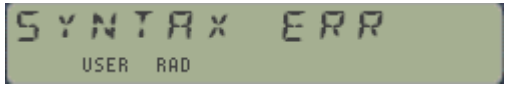


Buffer id#	Buffer Reg	Type	Used for:
06	b10	….	….
	b9	<other>	<available>
	b8	<other>	<available>
	b7	Hex code	Operator-3
	b6	BCD value	4th argument
	b5	Hex code	Operator-2
	b4	BCD value	3rd argument
	b3	Hex code	Operator-1
	b2	BCD value	2nd argument
	b1	BCD value	1st argument
	b0	admin	Header

Formula Evaluation Engine [FRML]

Operation Syntax





Buffer regs

Integer digits

Stack regs

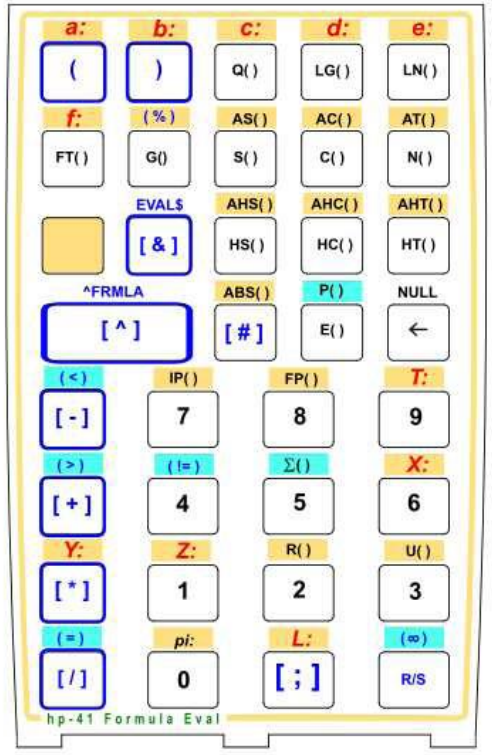
Control chars

Key	LCD Symbol	Function
[ ][a]	a	parameter
[ ][b]	b	parameter
[ ][c]	c	parameter
[ ][d]	d	parameter
[ ][e]	e	parameter
[ ][CLS]	F	parameter
[ ][π]	π	pi
[0]	0	integer
[1]	1	integer
[2]	2	integer
[3]	3	integer
[4]	4	integer
[5]	5	integer
[6]	6	integer
[7]	7	integer
[8]	8	integer
[9]	9	integer
[ ][X]	X	Variable
[ ][Y]	Y	Variable
[ ][Z]	Z	Variable
[ ][T]	T	Variable
[ ][LastX]	L	Variable
	Special Symbols	
[ ]P-R	Σ	SUM Eval
[ ]RTN	P	Product Eval
[,]	,	Semi-colon
[ ]VIEW	I	Infinite index
[ ][x=y?]	=	Comparison
[ ][x<=y?]	≤	Comparison
[ ][x=0?]	=	Comparison
[ ][BEEP]	! =	Comparison

Key	LCD Symbol	Function
[+]	+	Sum
[-]	-	Subtraction
[*]	*	Product
[/]	/	Division
[ENTER^]	↑	Power
[S+]	(	Open Parenthesis
[1/X]	)	Close Parenthesis
[CHS]	-	Negative value
[ ][ISG]	ABS	Absolute value
[ ][SF]	IP	Integer part
[ ][CF]	FP	Fractional part
[SQRT]	√	Square Root
[XEQ]	%	Modulus
[%]	%	Percentage
[ ][SCI]	^	Square Power
[ ][ENG]	^	Cube power
[EEX]	E	Exponential
[X<>Y]	FT	Factorial
[RDN]	G	Sign
[SIN]	S	Sine
[COS]	C	Cosine
[TAN]	N	Tangent
[ ][ASIN]	AS	Arc Sine
[ ][ACOS]	AC	Arc Cosine
[ ][ATAN]	AT	Arc Tangent
[STO]	HS	Hyperbolic SIN
[RCL]	HC	Hyperbolic COS
[SST]	HT	Hyperbolic TAH
[ ][LBL]	AHS	Hyperbolic ASIN
[ ][GTO]	AHC	Hyperbolic ACOS
[ ][BST]	AHT	Hyperbolic ATAN
[LN]	LN	Natural Log
[LOG]	LG	Decimal Log

Formula Evaluation Engine [FRML]

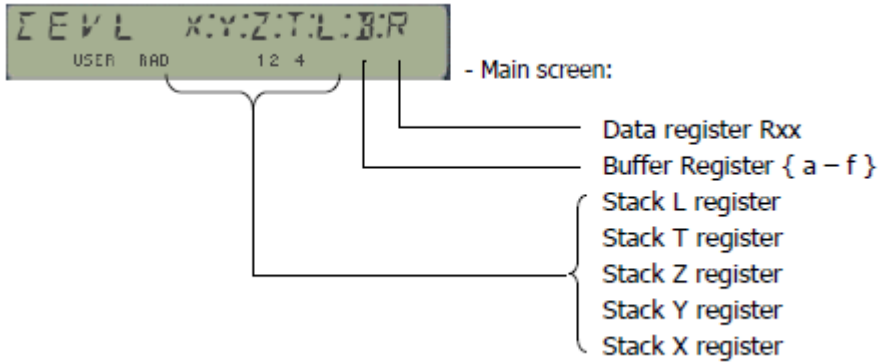
Creation → Storage → Execution



Builds Text in ALPHA
Autocompletes ")" at end

ALPHA text <-> Data Regs

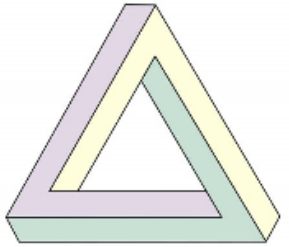
ALPHA text -> PRGM Step(s)



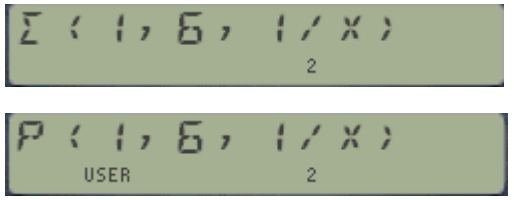
Can store the result in Stack regs, Buffer #7 regs, as well as any Data Register.

Very useful for Chain Calculations – either same variable or w/ new one.

Formula Evaluation Engine [FRML]
Sums, Products, & Comparison Tests



EVALΣ EVAL*P*

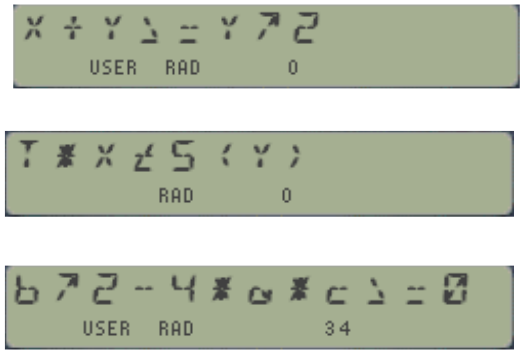


Special symbols [**Σ**], [**P**] for Sums and Products

[**I**] for *infinite* terms

Called separately via **EVALΣ**, **EVALP**, or Integrated in **EVAL$**

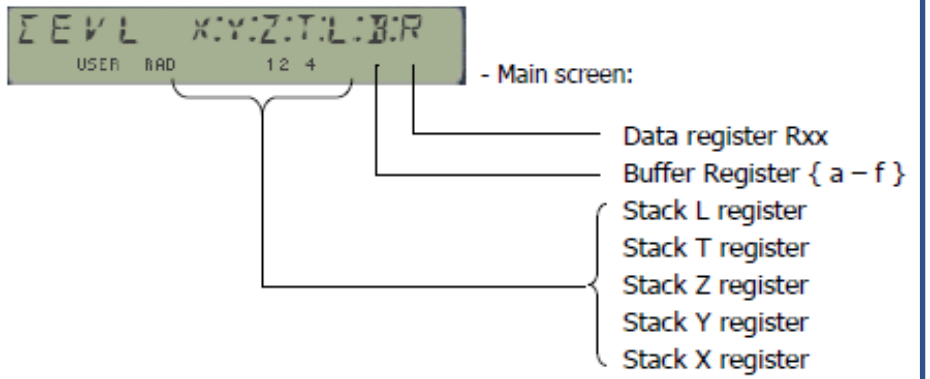
EVAL*?*



Comparison symbols divide formula in two sides.

No need for "(", ")" brackets or final "?"

EVAL$



Can store the result in Stack regs, Buffer #7 regs, as well as any Data Register.

Very useful for Chain Calculations – either same variable or w/ new one.

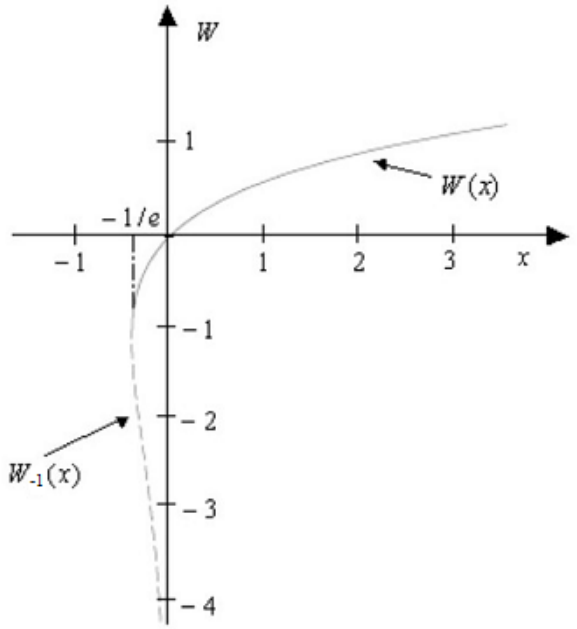
Formula Evaluation Applications [FRMX]

SYNTAX ERR
USER RAD

- ✓ Built-in Library of Applications (Upper 4k)
- ✓ Accuracy holds to 8th/9th decimal point
- ✓ Comparable runtimes to "RPN" approach

- **SV$, ITG$ - Solve & Integrate**
- AGM$ - Arithmetic-Geometric Mean
- 2D & 3D Distances; Dot-Products
- CL$, FL$ - Ceiling & Floor Functions
- QRT$ - Quadratic Equation Roots
- LINE$ - Line equation thru 2 points
- P4$ - Polynomial Evaluation (n<=4)
- NFD$ - Normal Density Function
- Rectangular <> Spherical
- Julian <> Calendar Dates
- Exponential function
- Erdós-Borwein constant

- GAM$ - Gamma Function
- PSI$ - Digamma Function
- LNG$ - LogGamma function
- WL$ - Lambert function
- Elliptic Integral 1st. Kind
- Combinations & Permutations
- Legendre Polynomials
- Hermite's Polynomials
- Chebyshev's Polyn. 1st & 2nd. Kind
- Sine & Cosine Integral
- Error Function
- Bessel J integer order
- Harmonic Number
- Generalized Faulhaber's Sum
- Ulam's Conjecture



$$E = \sum_{n=1}^{\infty} \frac{1}{2^{n^2}} \frac{2^n + 1}{2^n - 1}$$

$$E = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{1}{2^{mn}}$$

$$w_{j+1} = w_j - \frac{w_j e^{w_j} - z}{e^{w_j} + w_j e^{w_j}}.$$

$$z = W(z)e^{W(z)}.$$

$$\Gamma(z) = \frac{\sum_{n=0..N} q_n z^n}{\prod_{n=0..N} (z+n)} (z+5.5)^{z+0.5} e^{-(z+5.5)}$$

$$\Psi(x) = \log(x) - \frac{1}{2x} - \frac{1}{12x^2} + \frac{1}{120x^4} - \frac{1}{252x^6} + O\left(\frac{1}{x^8}\right)$$

Formula Evaluation Applications [FRMX]

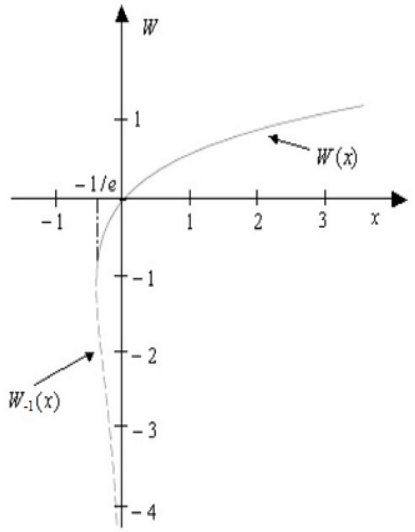
Two for the Road

$$\Gamma(z) = \frac{\sum_{n=0..N} q_n z^n}{\prod_{n=0..N} (z+n)} (z+5.5)^{z+0.5} e^{-(z+5.5)}$$

$$z = W(z)e^{W(z)}.$$

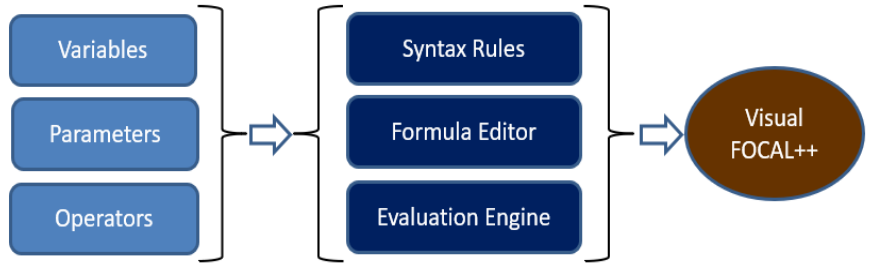
$$w_{j+1} = w_j - \frac{w_j e^{w_j} - z}{e^{w_j} + w_j e^{w_j}}.$$

1	LBL "WL$
2	FIX 9
3	STO Z
4	LN1+X
5	"X-(X*E(X)-Z)/(1"
6	"|-+X)/E(X)"
7	LBL 00
8	STO Y
9	EVAL$
10	FS? 10
11	VIEW X
12	X<>Y
13	RND
14	X<>Y
15	RND
16	X#Y?
17	GTO 00
18	END

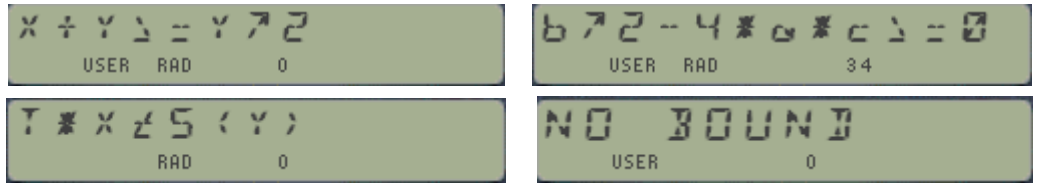


1	LBL "GAM$"	x>0		26	LET= c	
2	CF 04			27	8687.245297	q3
3	X<0?			28	LET= d	
4	X=0?			29	1168.926495	q4
5	GTO 04			30	LET= e	
6	RAD			31	83.86760434	q5
7	SF 04			32	ENTER^	
8	"1-X"	X=(1-X)		33	2.5066282	q6
9	EVAL$			34	LASTX	x
10	LBL 04			35	"c+X*(d+X*(e+X*("	polynomial term
11	"P(0;6;Y+X)"	denominator		36	>"Z+X*Y)))"	part-1
12	EVALP	saves x in R09		37	EVAL$	
13	STO 09	save for later		38	"a+b*L+X*L^2"	part-2
14	5.5			39	EVAL$	
15	LET= a			40	RCL 10	get partial result
16	LASTX	x		41	*	factor it in
17	"(X+a)^(X+1/2)/E"	trascendent term		42	RCL 09	get denominator
18	|-"(X+a)"			43	/	divi by it
19	EVAL$			44	FC? 04	negative argument?
20	STO 10	partial result		45	GTO 04	no, skip
21	75122.63315	q0		46	RCL 08	get (1-x)
22	LET= a			47	"π/Y/S(π*(1-X))"	reflection formula
23	80916.62789	q1		48	EVAL$	
24	LET= b			49	LBL 04	
25	36308.29514	q2		50	CLD	clear display
				51	END	done.

From Formula Eval to VF++



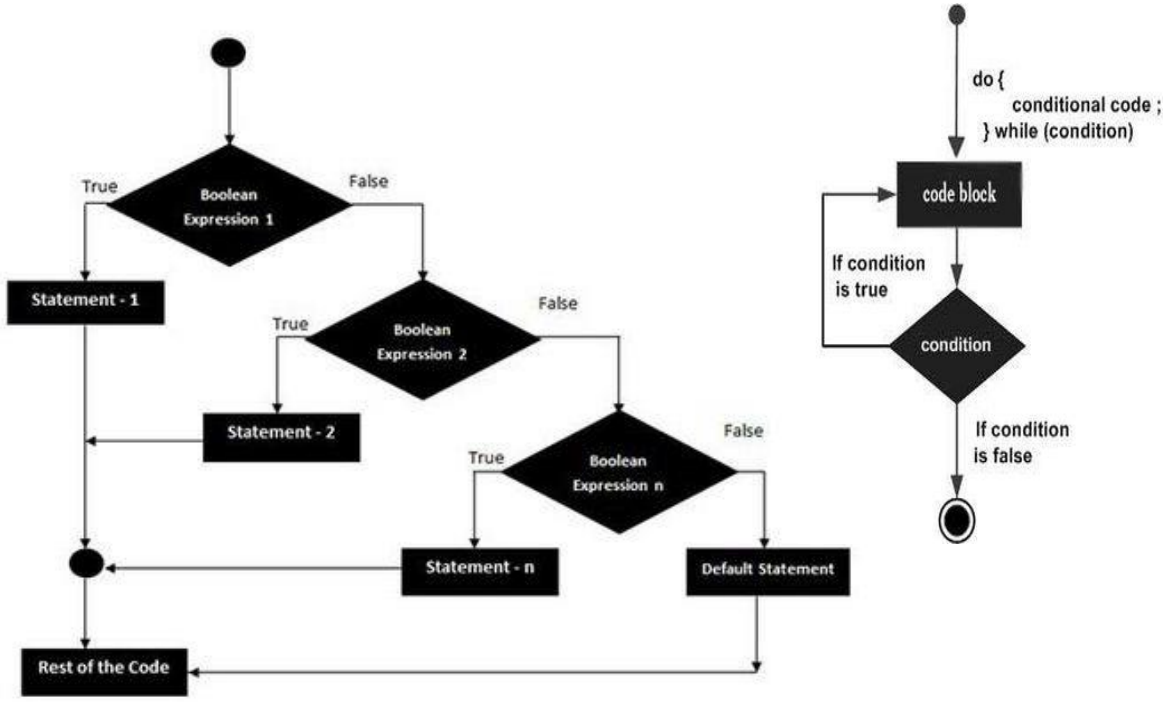
- **EVALΣ** and **EVALP** integrated into **EVAL$**
- Generalized Comparison Tests with **EVAL?**
  - ✓ {formula1} <,=,>,# {formula2}
  - ✓ Full MCODE implementation
  - ✓ LET= / GET= / SHOW= (buffer vars)



- Visual FOCAL++ Conditional Structures

EVAL*?*

- ✓ **SELECT / CASE / CASELSE / ENDSLCT**
- ✓ Nested structures partially supported
- ✓ **FOR…NEXT (STEP)**
- ✓ bbb.eee:ss control word
- ✓ Up to Six nested loops
- ✓ **IF / ELSE / THEN**
- ✓ Generalized formula comparison
- ✓ Up to Three nested structures (w/ caveats)
- ✓ **DO … WHILE**
- ✓ Generalized formula comparison
- ✓ Up to Three nested loops



VF++ Structures Under the hood

- **SELECT** (nn) (also IND, STK) — SELECTs Rnn in header; Activates Structure flag in header; Searches for [ENDSLCT] below and saves its address in header
- **CASE** (xxx) — if flag not active jump to [ENDSLCT]; checks whether Rnn=xxx?; if not, search for [CASE/ELSE] below and jump to it (or to ENDSLCT]; if yes, clears flag and runs clause
- **CASE** (yyy) — if flag not active jump to [ENDSLCT]; Checks whether Rnn=yyy?; if not, search for [CASE/ELSE] below and jump to it (or to [ENDSLCT]); if yes, clears flag and runs clause
- **CASELSE** — if flag not active jump to [ENDSLCT]; Deactivates structure flag; unconditional run of this clause
- **ENDSLCT** — Deactivates structure flag; Clears SELECTed reg in header

- **FOR** (nn) (also IND, STK) **bbb.eee:ss** in X — Selects Rnn and saves pointers in it; Saves its own address in header; Searches for [NEXT] below (Bound check)

  (**LOOPT** — returns kkk.eee:ss *current* pointers to X)
- **NEXT** (nn) — Increases/Decreases pointers; Returns to [FOR] if not done

- **DO** Boolean in ALPHA — Saves its own address in <RTN1>; Searches for [WHILE] below (Bound check)
- **WHILE** — Checks Boolean test in ALPHA; Returns to [DO] if True

- **IF** Boolean in ALPHA — Searches for [ENDIF] (or [ELSE]) below and saves its address in <RTN1>; Checks Boolean test in ALPHA; Jumps to [ELSE/ENDIF] if False
- **ELSE** — Jumps to [ENDIF] if Boolean was True
- **ENDIF** — Does nothing (but an audible blip)

VF++ Structures Under the hood (Con't)

Buffer #7 Header fields:

<13:12> id#	<11:10> Size	<9:6> RTNADR	<5:3> SELVAR#	<S&X> Flags
7:7:	0:8:	B:R:R:R:	R:E:G:	F:0:0

- RTNADR is the address to jump to at the bound if the condition is not met:
  [**FOR**] address at the [**NEXT**] step
  [**ENDSLCT**] address at the [**CASE**] step (after a previous CASE clause)
- SELVAR# is the pointer to the SELECTED register, declared at the SELECT step
- SELECTED register holds the FOR/NEXT Loop pointers (**bbb.eee:ss**)
- Active Structure Flag set by SELECT until a CASE clause is done.

FOCAL Return Stack:

b(12):

R	3	A	D	R	2	A	D	R	1	P	C	N	T	
13	12	11	10	9	8	7	6	5	4	3	2	1	0	

a(11):

A	D	R	6	A	D	R	5	A	D	R	4	A	D	
13	12	11	10	9	8	7	6	5	4	3	2	1	0	

- [**DO**] address is saved in <RTN1> field of status register b(12)
- [**ENDIF**] (or [**ELSE**]) address saved in <RTN1> field of status register b(12)
- These addresses are pushed to <RTN2> *when the execution moves to FOCAL for the Boolean test*:
  - If [**IF**] condition is met, <RTN2> is deleted so execution resumes after it.
    If it is not met <RTN1> is purged and <RTN2> is popped to <RTN1>
  - If [**WHILE**] condition is met, <RTN2> is purged so execution resumes after it (<RTN1>)
    if it isn't met then <RTN2> is popped to <RTN1> field for a repeat loop.

VF++ Structures Examples (Con't)

#1. Arithmetic-Geometric Mean

$$a_{n+1} = \frac{1}{2}(a_n + g_n)$$
$$g_{n+1} = \sqrt{a_n g_n}\,.$$

- *using a DO/WHILE structure*

```
01 LBL "AGM#"
02 "ABS(IP(Y))
03 EVALY
04 "ABS(IP(X))"      ; INT, ABS
05 EVAL$
06 DO
07 "(X+Y)/2"
08 EVAL$             ; an
09 "Q{L*Y)"          ; L, not X !
10 EVALY             ; gn
11 VIEW X            ; show value
12 X<>Y
13 RND
14 X<>Y
15 RND
16 "X#Y"
17 WHILE             ; in FIX 10
18 END
```

optional (lines 12–15)

#2. Collatz Conjecture

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \pmod 2 \\ 3n+1 & \text{if } n \equiv 1 \pmod 2. \end{cases}$$

- *IF/ELSE/ENDIF nested within a DO/WHILE*

```
01 LBL "ULAM#"
02 "ABS(IP(X))"
03 EVAL$             ; naturalize input
04 "X=0"
05 IF                ; x=0?
06 RTN               ; abort if x=0
07 ENDIF
08 "0"
09 EVALY             ; reset count
10 DO                ; loop begins
11 "(X&2)=0"         ; MOD (x,2)
12 IF                ; is x even?
13 "X/2"             ; yes, halve it
14 ELSE
15 "3*X+1"           ; no, increase it
16 ENDIF
17 EVAL$             ; evaluate new
18 VIEW X            ; show value
19 "Y+1"
20 EVALY             ; increase count
21 "X#1"
22 WHILE             ; repeat while
23 VIEW Y            ; show length
24 END
```

EVALXM+ Scripting Language

$$\Gamma(z) = \frac{\sum_{n=0..N} q_n z^n}{\prod_{n=0..N} (z+n)} (z+5.5)^{z+0.5} e^{-(z+5.5)}$$

- Uses X-Mem ASCII Files
- Each record limited to 24-chars max
- Variable assignment and store/recall syntax
- Adds its own flow control capability:
  DO/WHILE, GOTO, FOR/NEXT
- Supports SOLVE/INTEG commands

1	LBL "SV$"	*x0, x1 in Y,X*		
2	STO$ 07	*save integrand fnc,*		
3	LBL 00			
4	EVALZ	*f(x1)*		
5	X<>Y			
6	EVALT	*f(x0)*		
7	"Y-Z*(Y-X)/(Z-T)"			
8	EVAL$	*x2 = x1-f(x1).[x1-x0)/f(x1)-f(x0)]*		
9	FS? 10			
10	VIEW X(3)			
11	RCL$ 07	*recall integrand function*		
12	X#Y?	*is x1 = x'?*		
13	*GTO 00*	*no, do another loop*		
14	END	*yes, we're done.*		

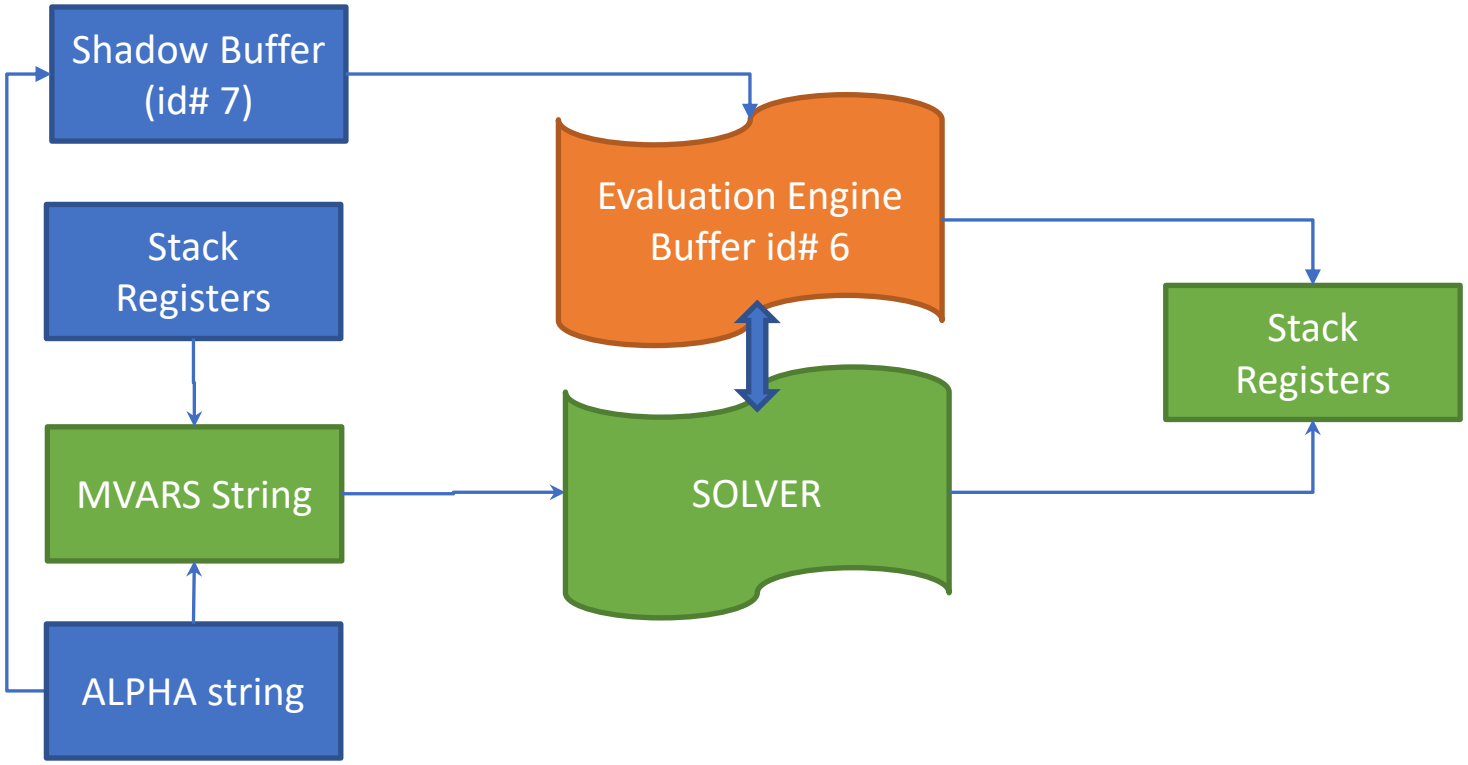
$$x_n = x_{n-1} - f(x_{n-1}) \frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}$$

Example: Lanczos Gamma

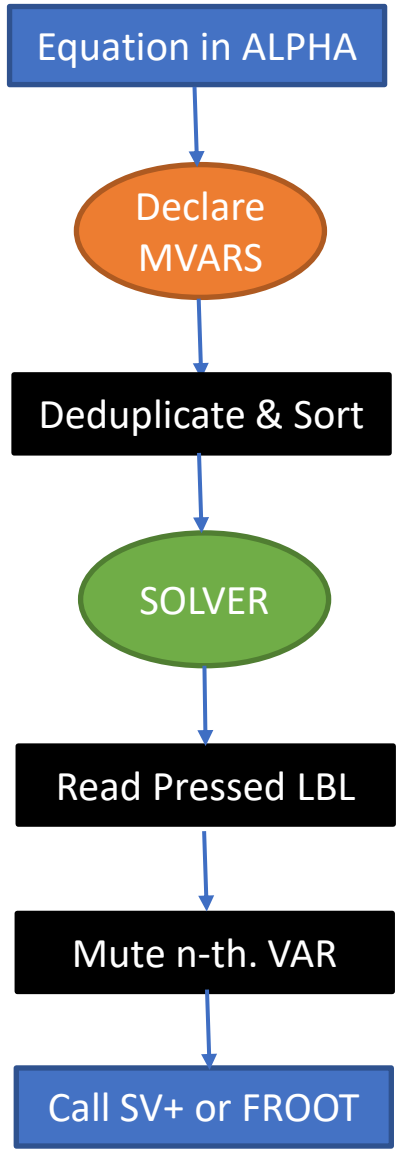
```
00  PP(0;6;Y+X)                ; product term
01  XS9                        ; result to R09
02  X=L                        ; argument back to X
03  a 5.5
04  X=(X+a)^(X+1/2)/E(X+a)     ; exponential term
05  XS10                       ; stored in R10
06  a 75122.63315              ; load coefficients
07  b 80916.62789              ; in buffer regs
08  c 36308.29514
09  d 8687.245297
10  e 1168.926495
11  Z 83.86760434
12  Y 2.5066282
13  X=L                        ; argument back to X
14  X=X*(d+X*(e+X*(Z+X*Y)))    ; had to leave off c due to length
15  Z=L                        ; use Z for L
16  X=X+c                      ; add it in here
17  X=a+b*Z+X*Z^2              ; changed this formula accordingly
18  YR10                       ; recall exponential result
19  X=X*Y
20  YR9                        ; recall product result
21  X=X/Y
```

Equation Solver Operation
Block Diagrams

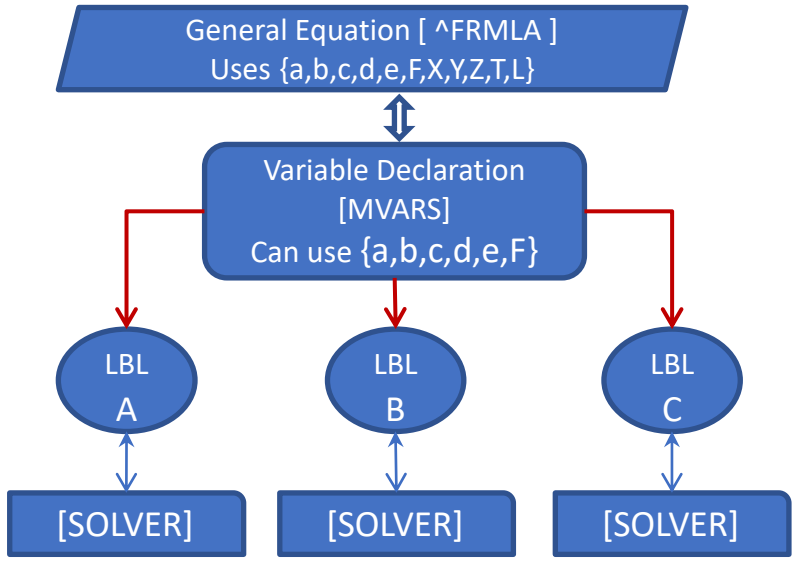
Input Data → *EVALuation* → *Result*



Multi-component complete solution:



Equation Solver Module
[SLVF]

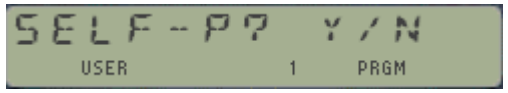


- ✓ Equations are interpreted using the Formula Evaluation Module.
- ✓ Long equations may be chained in two parts (using SLV+)

- ✓ Variables are declared with **MVARS**; resolved via **SOLVER**

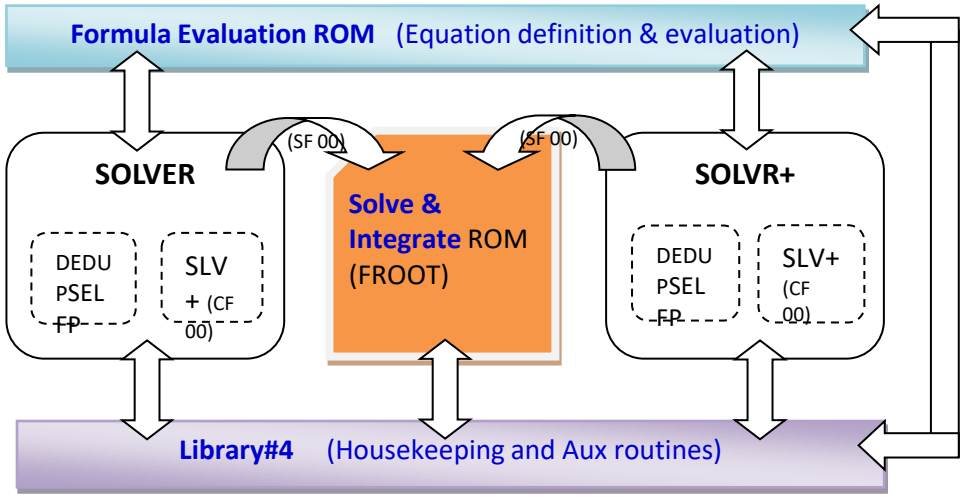
- ✓ Two SOLVER sets are available: 5-Vars (Custom) or 6-Vars (A-F)

- ✓ *Self-programming* feature completes the FOCAL Solver code

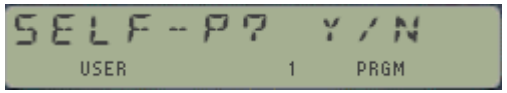


- ✓ Mini-Library with 15 example equations is included

- ✓ Two options of SOLVE engines: built-in "**SLV+**" (CF 00) or **FROOT** (SF 00) in Solve & Integrate ROM.



Equation Solver Module
Variable Usage

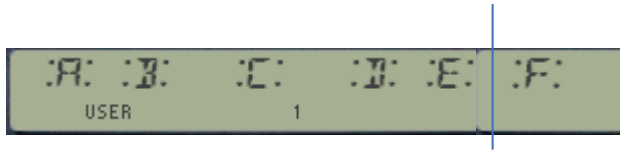


- Variables are declared with **MVARS** / **MVRS+**; and resolved via **SOLVER** / **SLVR+**

- up to 5 MenuVars (custom letter)
- Mapped to local LBL [A]-[F] by position in LCD
- Mapped to FRMLA vars (buffer regs) *by position in vars string*
- Entries are de-duplicated
- Stored in buffer header <0:7> in ASCII format

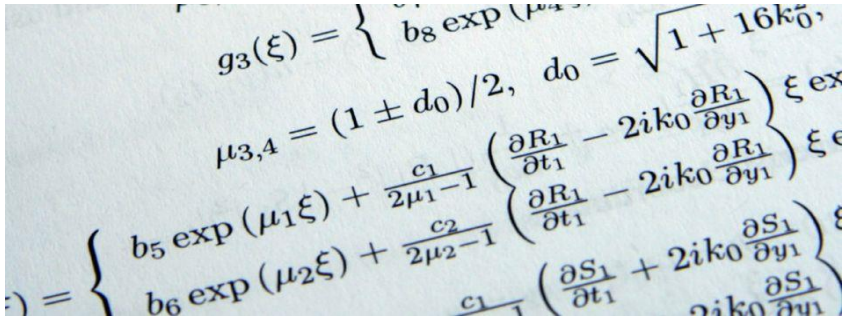
<13:12>	<11:10>	<9:8>	<7:6>	<5:4>	<3:2>	<1:0>
Id#	Size	V1	V2	V3	V4	V5

- Up to 6 MenuVars (A-F, Z) - XYTL are used as scratch
- Mapped to local LBL [A]-[F] *by position in LCD*
- Mapped to FRMLA vars (buffer regs) *by its name*
- Entries are de-duplicated and sorted alphabetically
- Stored in buffer header <6:7> in binary format



<13:12>	<11:10>	<9:8>	<7:6>	<5:4>	<3:2>	<1:0>
Id#	Size	-	00000FLT|ZYXEDCBA	-	-	-

Equation Solver Module
Built-in Mini-Library

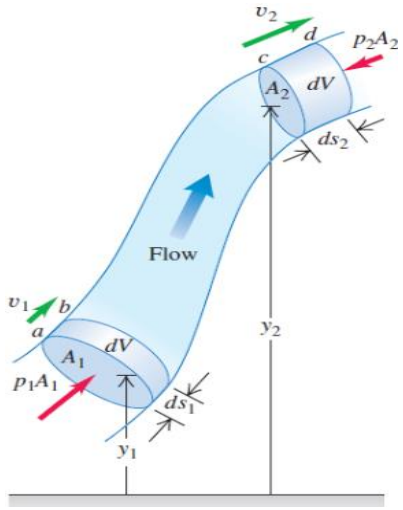


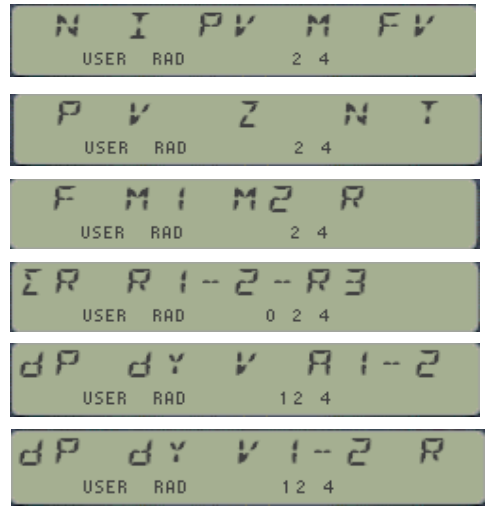
Routine	Equation	LCD Display
3DM	**3D Vector Module** M = SQRT(x^2 + y^2 + z^2) 4 variables, MVARS	:X: :Y: :Z: :M: PRGM
CTRY	**Catenary Curve** d =H [1 - (1/cosh(L/2a))] 4 Variables, MVARS	:A: :H: :L: :D: PRGM
HTX	**Heat Exchangers (Counterflow)** See equations in next pages 5 ½ Variables, MVARS, Chained.	:1: :2: :I: :O: :O: USER RAD 1 4 PRGM
KPL	**Kepler Equation** E – ec. sin E = m 3 Variables, MVARS	:M: :E: :C: RAD PRGM
LMOV	**Linear Movement** x = v.t + a. t^2 /2 4 variables, MVARS	:X: :T: :V: :A: USER RAD 1 4 PRGM
RGAS	**Real Gas Equation** P.V = Z.N.R.T 5 Variables + 1 constant, MVARS	:P: :V: :T: :Z: :N: RAD PRGM
RdK	**Redlich-Kwong EOS** P + a /[sqrt(T).Vm.(Vm+b)] = RT/(Vm-b); 5 Vars + 1 const, MVARS	:P: :V: :T: :A: :B: RAD PRGM
TVMS	**Time Value of Money (uses UF 02)** See chained equations in page 13 5 ½ Variables, MVARS, Chained	:P: :I: :M: :N: :F: USER RAD 01 4 PRGM
VdW	**Van-der-Waals EOS** P + (a/Vm^2) = R.T /(Vm-b) 5 Variables + 1 constant, MVARS	:P: :V: :T: :A: :B: RAD PRGM
Y=P1	**Straight Line Equation** y = A.x + B 4 Variables, MVARS	:A: :B: :X: :Y: RAD PRGM
Y=P2	Quadratic Equation y = A.x^2 + B. x + C 5 Variables, MVARS	:A: :B: :C: :X: :Y: RAD PRGM
Y=P3	**Cubic Equation** Y = x^2 + B.x + C 5 Variables, MVARS	:A: :B: :C: :X: :Y: RAD PRGM
Y=P4	**Quartic Equation** Y = x^4 + A.X^3 + B.x^2 + C.x +D 6 Variables, MVRS+	:A: :B: :C: :D: :E: RAD PRGM

FOCAL Equation Libraries

From: $f(x)=0$ to: $f(a,b,c,d,e)=0$

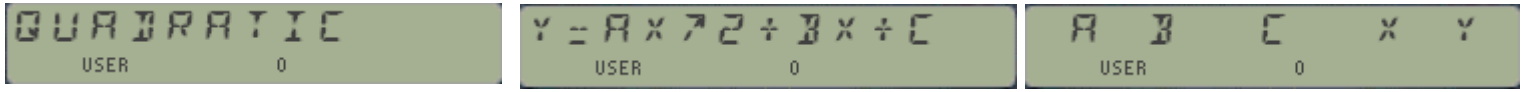
- Up to five variables to solve for. (at least 3)
- Mapped to local Labels [A]-[E] - and {R01–R05}
- FOCAL driver I/O, UF-22 controlled knowns/unknowns
- Other Constants prompted separately





- Two versions available:
  - ✓ `[ISOL]` "Interchangeable Solutions", uses FROOT

  - ✓ `[EQLB]` "Formula-Evaluation" based, uses "SLV$"
  - ✓ ASCII File includes all information. Navigation keys: PREV, NEXT, S&R
  - ✓ *User can add own custom equations*



$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Equation Name	POSROOT	NEGROOT
Equation Expression	X1=-B+SQRT(B^2-4AC)/2A	X2=-B-SQRT(B^2-4AC)/2A
Equation Formula	(#b+Q(b^2-4*a*c))/2/a-d	(#b-Q(b^2-4*a*c))/2/a-d
Equation Menu	A B C X1	A B C X2

Equation Libraries in Detail [ELIB]

#	Eq. NAME	Description	CF00 <- Options -> SF00
1	" ∠) RLC	Delay angle RLC	Serial RLC / Parallel RLC
2	" BERN-A	Bernoulli w/Areas	
3	" BERN-V	Bernoulli w/Velocities	
4	" CATNRY	Catenary Equation	d = H.[1 - (1/cosh(V/2a))]
5	" F=Mα	Force Equation	
6	" F=MM/R2	Gravitation	
7	" HEATX	Heat Exchangers	Counterflow / Parallel Flow
8	" HERON	Heron Formula	
9	" ISENT	Isentropic Flow	
10	" KEPLR	Kepler Equation	
11	" MPAD	Michell Pad	
12	" MVT	Linear Movement	x = x0 + v.t + 1/2 a.t^2
13	" ORBIT	Orbital Trajectory	e = sqrt{1+[Vo^2 - 2g(Ro2/r)][r.Vo.cos a / g.Ro]^2}
14	" PRJT-D	Projectile Displacement	X-Coordinate / Y-Coordinate
15	" PRJT-V	Projectile Velocity	X-Coordinate / Y-Coordinate
16	" PV=ZNT	Ideal Gas EOS	
17	" R 1+R2	Parallel Resistors	
18	" REDLCH	Redlich-Kwong EOS	
19	" RFLC	Resonance Frequency RLC	
20	" RPM=TP	Torque Equation	
21	" SRL-AC	Serial RL AC Control	
22	" TVM	Time Value of Money	Begin mode / End mode
23	" V=IR	Ohm's Law	
24	" VMOD	Vector Module	
25	" VWAALS	Van-der Waals EOS	
26	" Y=αX+b	Linear Equation	P1 = a.x + b
27	" Y=PX2	Quadratic Equation	P2 = a0 + a1.x + a2.x^2
28	" Y=PX3	Cubic Equation	P3 = 1 + a1.x + a2.x^2 + a3.x^3

FROOT Based:

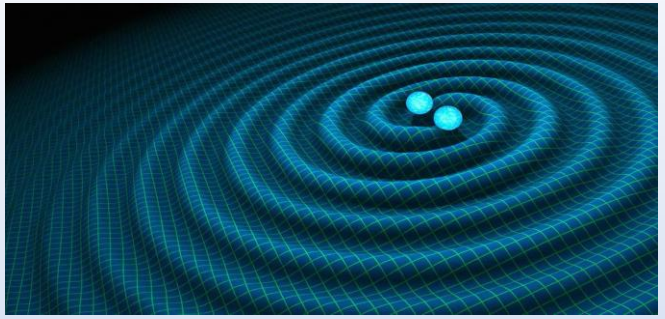
#	NAME	Description	Equation
1	LINEAR	Line Equation	Y = Ax + B
2	QUADRATIC	Quadratic Equation	Y = A.x^2 + .Bx + C
3	CUBIC	Cubic Equation	Y = A.x^3 + B.X^2 + C.x + D
4	4TH ORDER	Quartic Ecuation	
5	POSROOT	Positive Root	x1 = (-B + sqrt(B^2 - 4A.C))/ 2A
6	NEGROOT	Negative Root	x2 = (-B - sqrt(B^2 - 4A.C))/ 2A
7	OHMS LAW	Ohm's Law	E = I.R
8	PARALLEL R	Parallel Resistors	1/R1 = 1/R1 + 1/R2
9	RLC FREQ	RLC Oscillator	F0 = 1/sqrt(L.C)
10	GAS EQUATION	Perfect Gas Equation	P.V = N.R.T
11	LIN. MOTION	Linear Motion	X = V.T + 1/2 A.T^2
12	NEWTONS LAW	Newton's Law	F = G. M1.M2/R^2
13	INTEREST	Compound Interest	
14	+TVM END MODE	Extended version-1	
15	+TVM BEGIN MODE	Extended version-2	
16	*…add your own equations here…*		

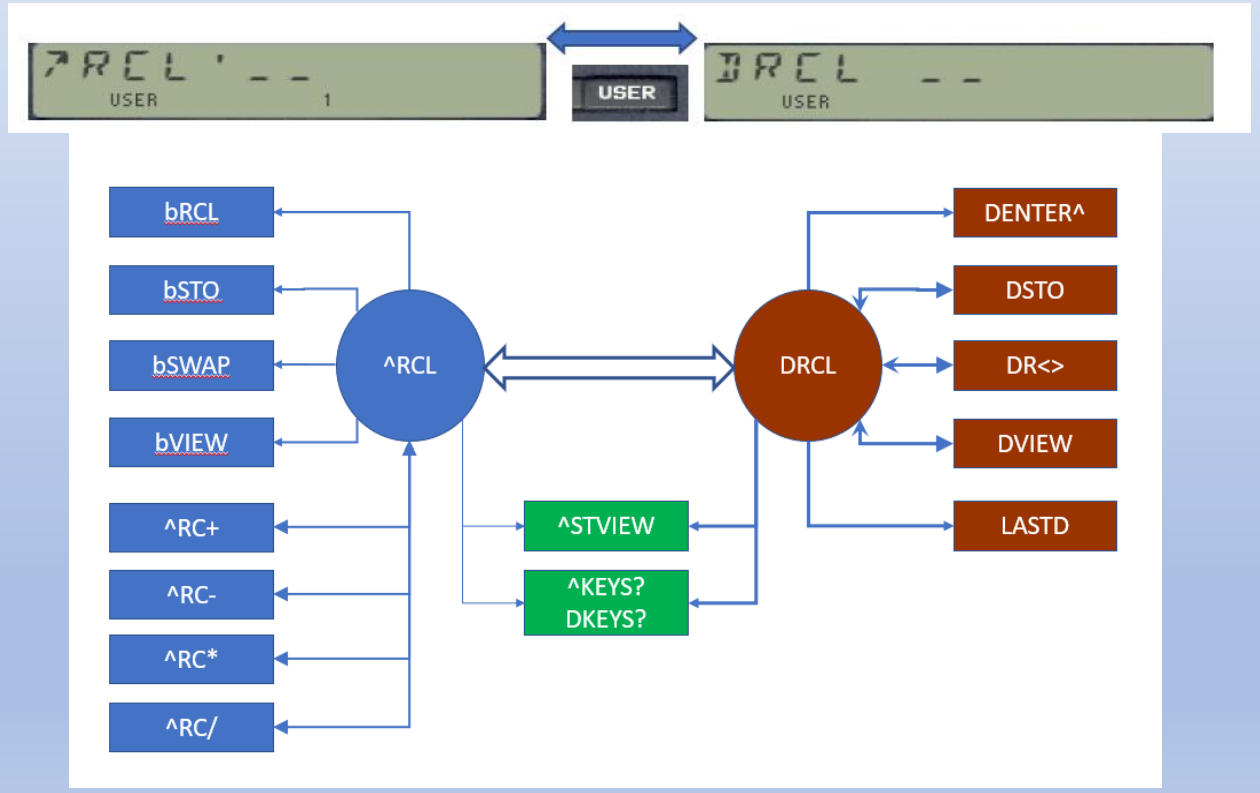
Formula-Eval Based:

1	LBL "ESLV"
2	STO$ 07
3	LBL 00
4	EVALZ
5	X<>Y
6	EVALT
7	Z=T?
8	*RTN*
9	*"Y-Z*(Y-X)/(Z-T) "*
10	EVAL$
11	FS? 10
12	VIEW X(3)
13	RCL$ 07
14	X#Y?
15	*GTO 00*
16	END

1	LBL "ELSV+"
2	LBL 00
3	RCL$ 07
4	EVALT
5	RCL$ 11
6	EVALT
7	R^
8	STO T(0)
9	RDN
10	X<>Y
11	RCL$ 07
12	EVALT
13	Z=T?
14	RTN
15	*"Y-Z*(Y-X)/(Z-T)"*
16	EVAL$
17	FS? 10
18	VIEW X(3)
19	X#Y?
20	*GTO 00*
21	END

Double-Down Module:
Double Length Stack (8-levels)
Dual Real Numbers





		REAL	Register	DUAL		
		LGKT	G:	*Scratch*		
		Scratch	F:	DL		
		Scratch	e:			
		D	d:	DT		
		C	c:			
		B	b:	DZ		Dual Number
Double		A	a:			Stack
Stack		T	T:	DY		
		Z	Z:			
		Y	Y:	DX		
		X	X:			
		L	L:	*Scratch*		

Double-Down Module
I – Double Length Stack

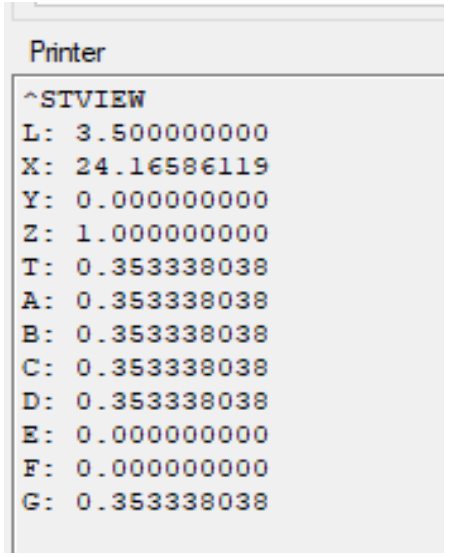
[XSTK]

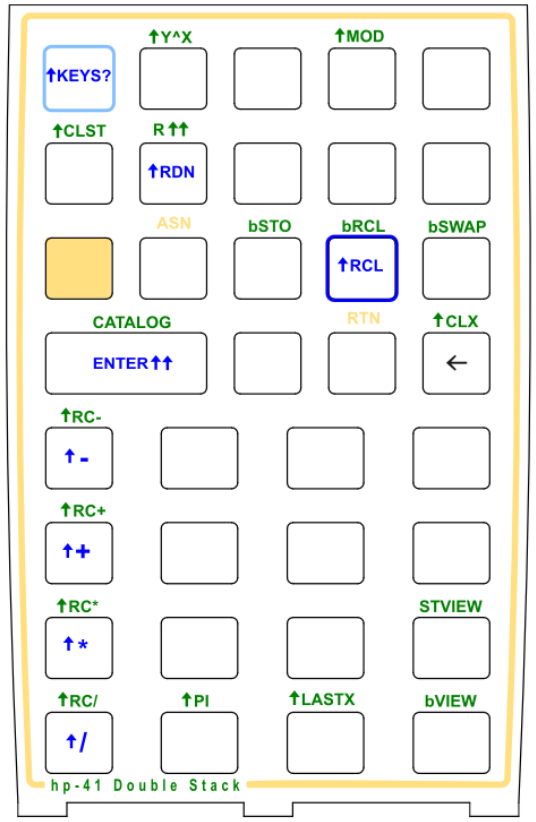
Buffer id#	Buffer Reg	Type	Used for:
07	b7	BCD value	**Last Good Known "T"**
	B6	BCD value	Dual LastD – Dual part
	b5	BCD value	Dual LastD – Real part
	b4	BCD value	Stack Level "D"
	b3	BCD value	Stack Level "C"
	b2	BCD value	Stack Level "B"
	b1	BCD value	Stack Level "A"
	b0	admin	Header

- Seamless support of 8 Stack registers
- Automatic Stack Drop & Duplication
- "Last Good Known-T" Register (LGKT) in buffer #7
- Uses I/O_SVC Polling point to trap BackArrow
- Replacement set of stack-lifting functions
- Supports IND, ST, RG, and combinations

Stack Management		Math Functions	
XROM #	Function Name	XROM #	Function Name
01,06	↗CLST	01,02	↗+
01,07	↗CLX	01,03	↗-
01,08	ENTER↑↑	01,04	↗*
01,09	LASTX↗	01,05	↗/
01,12	R↗↗	01,10	↗MOD
01,13	↗RDN	01,11	PI↗
01,14	↗RCL __	01,20	↗Y↗X
01,15	↗RC+ __		
01,16	↗RC- __	Admin Functions	
01,17	↗RC* __	01.,01	↗KEYS?
01,18	↗RC/ __	01,19	↗STVIEW
04,26	XFILL↗		

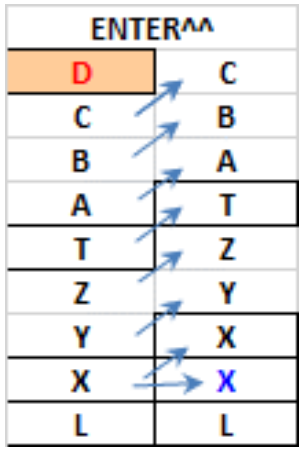
[FRMX]

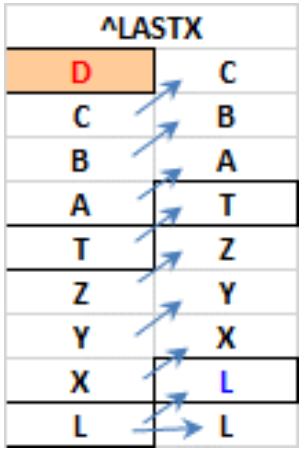


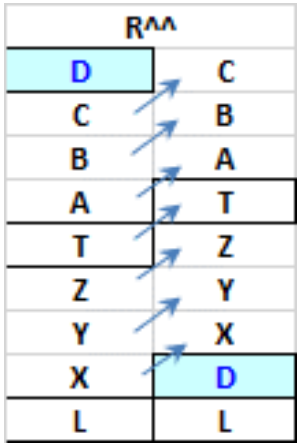


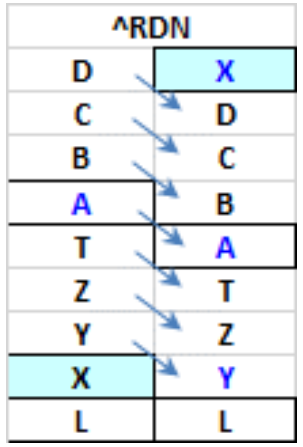
Double-Length Stack

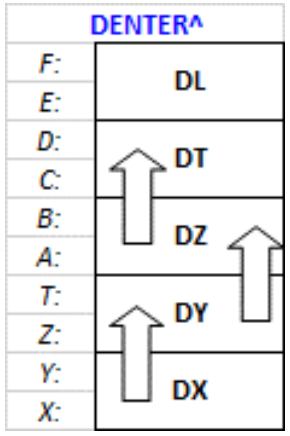
Double-Down Module (Con't)
"Fantastic Four" Diagrams

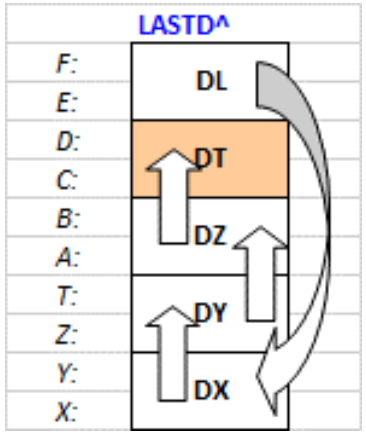


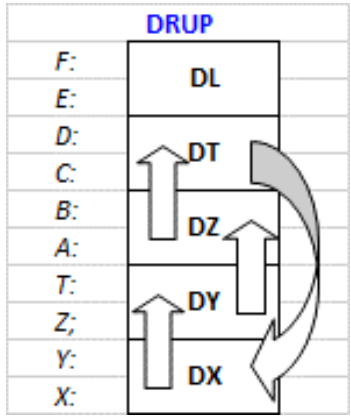


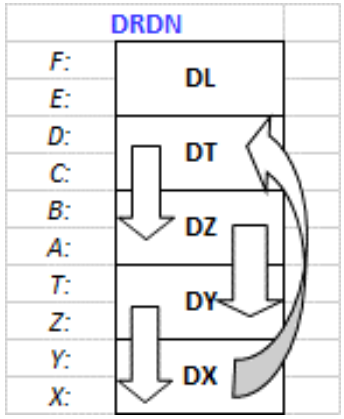












Controlled Events:

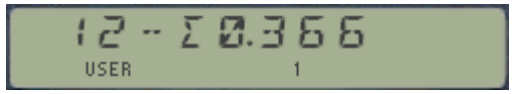
- Stack Drop (Dual Fns.)
- Stack Lift (w/ F11 set)
- Stack Roll
- D-Reg Duplication

I/O Special Events:

- Digit Entry (IO_SVC)
- CLX / BackArrow

Register	*Initial*	*DENTER^*	*digit entry*
F:	DL	DT	DT Real Part
e:			DZ
d:	DT	DZ	
c:			DY
b:	DZ	DY	
a:			DX
T:	DY	DX	
Z:			DX
Y:	DX	DX	
X:			*new digits*

Double-Down Module
II – Dual (Real) Numbers



- Full 4-Level, Dual Number Stack
- Supports IND, ST, RG, and combinations
- Set of Math applications Included

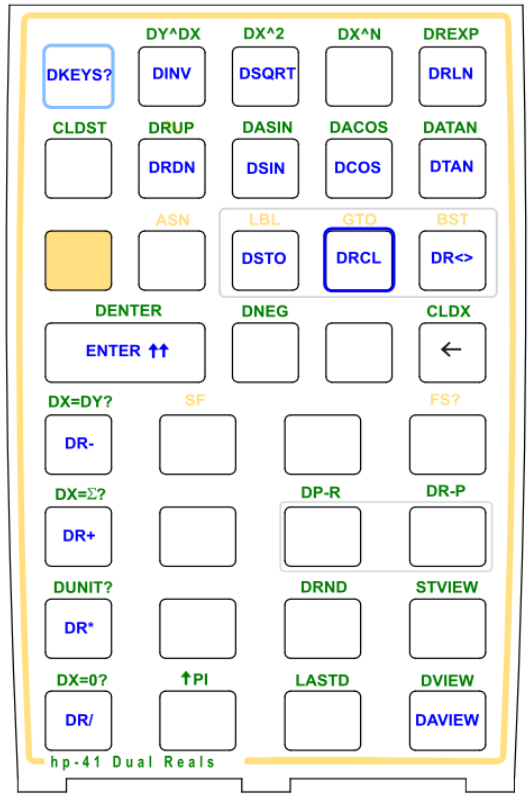
Admin / Stack Management		Math Functions	
XROM #	Function Name	XROM #	Function Name
01,26	CLDST	01,31	DR+
01,27	CLDX	01,32	DR-
01,28	DAVIEW	01,33	DR*
01,29	DENTER↗	01,34	DR/
01,30	DKEYS?	01,35	DR↗N
01,36	DRCL __	01,38	DREXP
01,37	DRDN	01,39	DRINV
01,41	DRND	01,40	DRLN
01,44	DRUP	01,42	DRNEG
01,45	DSTO __	01,43	DRSQRT
01,46	DUNIT?	01,53	DY↗DX
01,47	DX=0?	01,56	DR-P
01,48	DX=S?	01,57	DP-R
01,49	DX=DY?	01,58	DCOS
01,50	DX∠Y __	01,59	DACOS
01,51	DX∠YDY	01,60	DSIN
01,54	DVIEW __	01,61	DASIN
01,55	LASTD	01,62	DTAN
		01,63	DATAN

$$e^{b\varepsilon} = \sum_{n=0}^{\infty} \frac{(b\varepsilon)^n}{n!} = 1 + b\varepsilon,$$

$$W(z) = u + v\varepsilon = W(a) + \epsilon \frac{be^{-W(a)}}{1+W(a)}$$

$$Psi(z) = -\left(Ln\, w + \frac{w}{2} + \frac{w^2}{12}\left(1 - \frac{w^2}{10}\left(1 - \frac{10}{21}w^2\right)\right)\right)$$

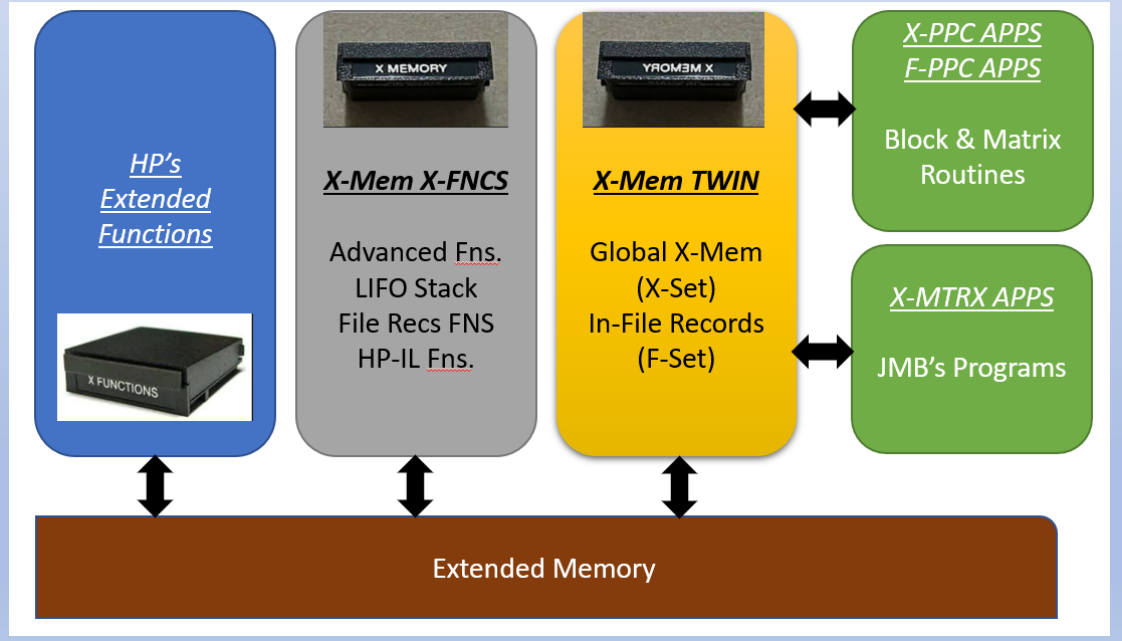
$$f(a+b\varepsilon) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a) b^n \varepsilon^n}{n!} = f(a) + bf'(a)\varepsilon,$$



Dual Number ROM

X-Mem Extensions:
X-Mem e-Twin
X-Mem X-Functions

Functionaliry	Mainframe / X-Fns	X-Mem X-FNCS	XM TWIN
Data File Read	GETX, GETR, GETRX	FLRCL, FLVEW	FRCL, FRC- , FRC+, FRC*, FRC/, FVEW
Data File Write	SAVEX, SAVER, SAVERX	FLSTO, FLX<>	FSTO, FST-, FST+, FST*, FST/, FX<>
Data Find	POSA, POSFL	WORKFL	POSDF, XFINDX
File Renaming and Re-Typing		RENMFL, RETPFL	
Buffer, Keys and Status File		SAVEB, GETB, SAVEK GETK, SAVE/GETST	
File Copying, Checking		FLCOPY, RSTCHK, XMXQ	
Data Regs Management	CLX, CLRG, CLRGX, REGMOVE, REGSWAP	CLXM, CLMM	CLEM, XCLREG, CLRGX XRGMOV, XRGSWP
Data Regs Recall	RCL, ARCL	ARCLIP, FLHD	XRCL, XRC-, XRC+, XRC*, XRC/, XARC
Data Regs Storage	STO, ST-, ST+, ST*, ST/, X<>, ASTO	REC+, REC-, ADVREC	XSTO, XST-, XST-, XST*, XST/, XAST
Sort, View & Exchange	VIEW, X<>Y, X<>	SORTFL, A<>RG, A<>ST, ST<>RG	XVEW, XX<>, A<>XRG, ST<>XRG



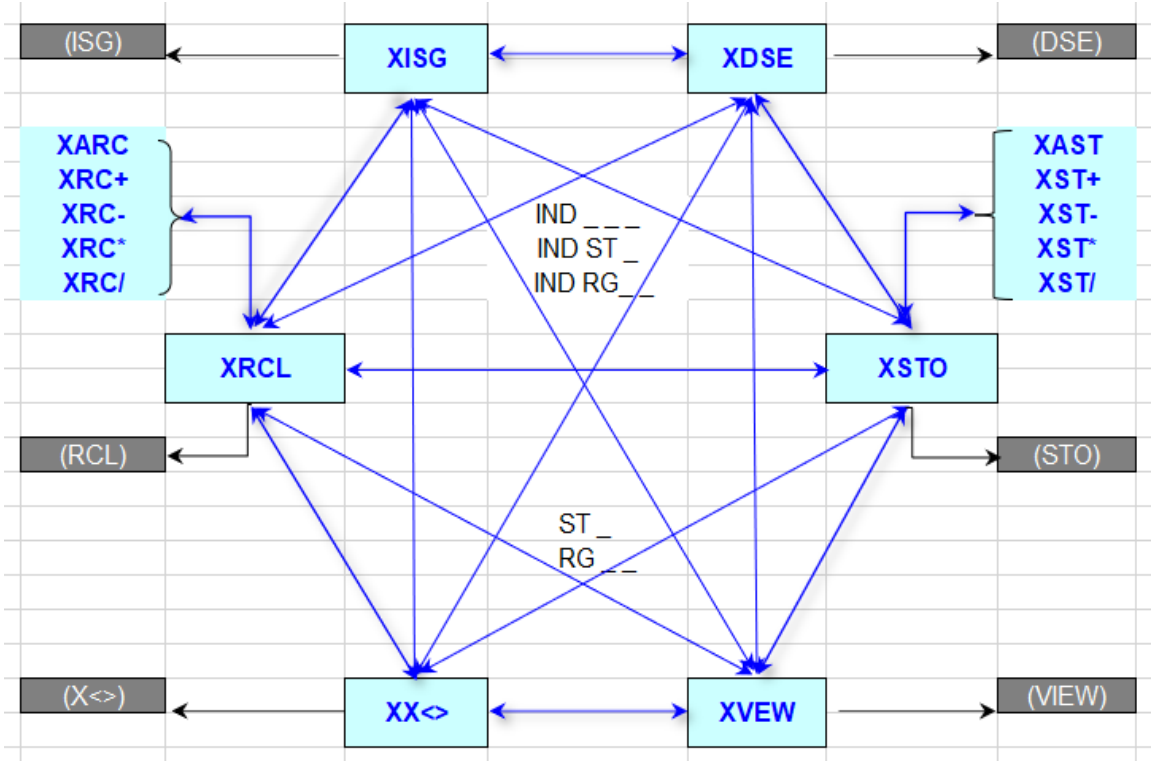
X-Mem TWIN Module
I - **Global** X-Mem Area

[XMTW]

- *Can you spot something different?*

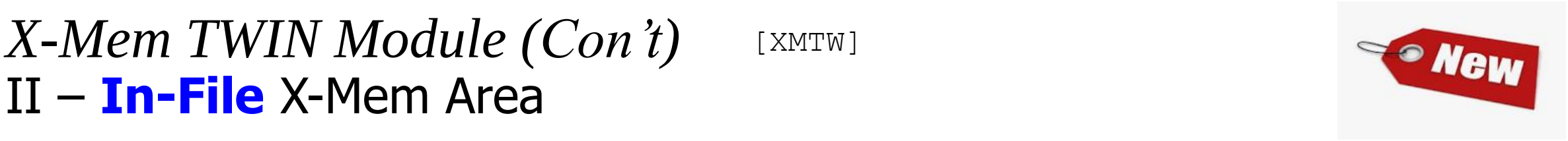
Extended Regs	Store	Recall	Other	X-Blocks
X-Register : From 0 to 605.	XSTO___	XRCL___	XX</>___	CLXRG
	XST+___	XRC+___	XVEW___	CLXRGX
	XST-___	XRC-___	XDSE___	XRGMOV
	XST*___	XRC*___	XISG___	XRGSWP
	XST/___	XRC/___	XFINDX	ST</>XRG___
ALPHA	XAST___	XARC___		A</>XRG___

- ALL 605 XM-Registers Directly accessible (!)
- STO/RCL Math, ISG/DSE, ALPHA, etc…
- Supports IND, ST, RG, and combinations
- *All Programmable but not all meant for PRGM use*

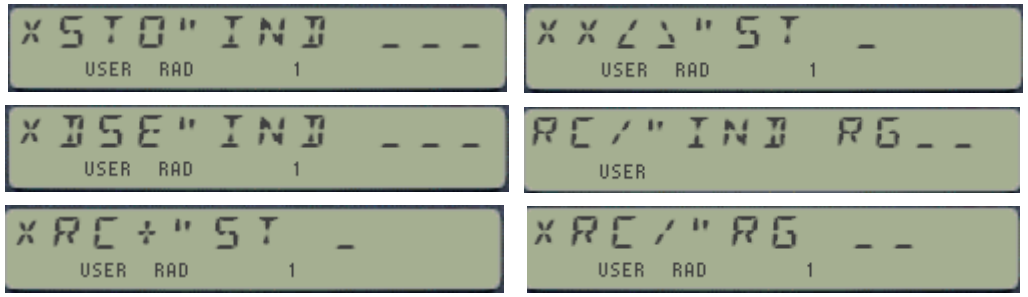


X-Mem TWIN Module (Con't)
II – **In-File** X-Mem Area

[XMTW]



- Data File Registers Directly accessible
- STO/RCL Math, ISG/DSE, ALPHA, etc…
- Supports IND, ST, RG, and combinations
- *All Programmable but not all meant for PRGM use*



- *Can you spot something different?*

Extended Regs	Store	Recall	Other	F-Blocks
F-Register : From 0 to FileSize -1	FSTO___	FRCL___	FX<>___	CLFL
	FST+___	FRC+___	FVEW___	GETR/X
	FST-___	FRC-___	FDSE___	SAVER/X
	FST*___	FRC*___	FISG___	FSHFT
	FST/___	FRC/___	POSDF	RDUMP
ALPHA	FAST___	FARC___