TACS USER MANUAL

Tachyon Analog Computer Simulator
Nicholas G. Lordi April 2018

This manual lists and describes the TACS vocabulary needed to solve differential equations on the propeller. Appendix 1 adds a listing of words relating to the math coprocessor and display device. Some knowledge of Forth and especially Tachyon is assumed. Appendix 2 describes the TACS hardware and lists the software components.

Utilities

== 345 == *rate* defines *rate* as a constant, which pushes 345 on the stack when executed..

TO An assignment operator for scalar parameters defined as constants. 500 **TO** *rate* changes its value to 500.

DECIMAL This is the default base. Execute to avoid misinterpretation of number assignments and output. #34 is a DECIMAL number while $34 is a HEX value. %101011 represents binary numbers.

.S Displays the current stack state. Useful for debugging.

FORGET Executing **FORGET (TACS)** deletes the TACS system. Use to remove recent code that is not needed.

BACKUP Makes any changes in code permanent.

SET-TO An alias of **REVECTOR**. **SET-TO** *golf tennis* makes *golf* behave as *tennis*.

pub .... ; Define new models or commands, eg, **pub** <*name*> -- existing words -- **;**

INIT Execute to initialize the math coprocessor and LCD module, as well as, set default parameters.

LCD Execute to set console output to the lcd screen.

CON Execute to set console output to default.

QW Execute to display the most recent words added to the vocabulary.

REBOOT Execute to set system to state equivalent to power off/on.

***/** (n1 n2 n3 –) n1*n2/n3 where n1*n2 is a 64 bit multiply and result is 32 bits.
***/y** (n1 n2 –) n3 is Y0
***/s** (n1 n2 –) n3 is scale
***//** (n1 n2 n3 –) n3 is a negative value

System Constants

Y0 The scaling factor for state variables, equivalent to 1.0: default 1000000.

Ym The maximum allowed value of state variables: default 100000000.

scale A constant used to scale model parameters (scalars) : default 10000. For example, #5000 is equivalent to 0.5.

fix A constant which specifies the number of decimal places used in displaying output: default 3.

comint The communication interval, ie, the number of time steps which elapse before output is requested: default 100.

+REF = #1 Constants associated with the reference values **Y0**, **0,** and **-Y0.**
0REF = #2
-REF = #3

Pi Leaves 31416, equivalent to 3.1416, on the stack.

Ei Leaves 27183, equivalent to 2.7183, on the stack.

System Variables, Vectors, and Arrays

Those words used by specific blocks will be defined in the blocks section.
Define variables as follows: simple variables - A (-- value) & >A (value --) and vectors or arrays of long values - A() (index -- value) & >A() (value index --).

Y() & >Y() A vector defining state variables accessed by all blocks: default 52. The first four elements are reserved for system use. Y(0) is the time elapsed, Y(1) is +REF, Y(2) is 0REF, and Y(3) is -REF.

IC() & >IC() **T**he initial condition vector which stores the initial conditions assigned to blocks in a model: default 24. Two elements are used for each initial condition.

K() & >K() The model parameter vector, which is assigned to potentiometers: default 24. Values maybe greater or less than 1 scaled.

N() & >N() An array of words (2 bytes) which stores nodes associated with state variables for which output is requested: default 12.

B() & >B() An array of bytes which stores logic bits -1 (true) or 0 (false) associated with logic block nodes: default 48.

DT & >DT A word variable setting the integration step interval in simulated time: default 10, equivalent to 0.001.

SMAT (row col -- value) leaves a value stored in a predefined matrix in upper (>32K) memory.

>SMAT (value row col --) stores value in the specified matrix location.

outputs

NODES (n --) Sets n number of nodes in a model..

OUT-NODES (n1 n2... --) Specify the specific nodes for which output is requested, other than time, which is automatically included. : 11 7 5 OUT-NODES displays Y(11) Y(7) Y(5). Nodes are stored in the vector N().

DISPLAY (--) Lists time and specified outputs in columns on the active display.

MSTORE (--) Store simulation data in a matrix for further analysis.

MREAD (row --) Read specified row from stored data in the matrix.

MDSPLY (n --) Reads and displays n rows of stored data.

ZMAT (–) Clears system matrix.

PLOTY (--) Plots node in 1 N() versus time on graphics display.

PLOTY2 (--) Plots nodes in 1 N() (yellow) and 2 N() (orange) versus time.

PLOTXY (--) Plot y-vector (1 N()) vs. x-vector (2 N()), using a specified color.

>YNODE (n --) Set y-node to n.

>XNODE (n --) Set x-node to n.

>COLOR (color name --) Use this word to set current color in plots.

>PLOTY (node --) Change node in 1 N() to be plotterd.

>PLOTX (node --) Change node in 2 N() to be plotted in xy-plt modes..

>YGAIN (value --) Set y-gain in graphics mode : default *scale*.

+Y & -Y (--) Doubles and halves current y-gain.

>XGAIN (value --) Set x-gain in graphics mode: default *scale*.

+X & **-X** (--) Doubles and halves current x-gain.

Note: The default is equivalent to no effect on the plot scale. Increasing or decreasing this value, has a corresponding effect on the magnitude to the plotted data.

MPLOTY (col --) Plots specified column of data in stored matrix. Set number of rows to be

>ROWS (n --) Use to set no. rows to be plotted.

MPLOTXY (xcol ycol --) Select specific columns to be plotted in xy-mode (gph2 & gph3).

The following words set 4 different graphing modes.

gph1 Positive y-time plots.
gph2 Positive/Negative y-time plots.
gph3 Full Screen 4-quadrant xy-plots.
gph4 4-quadrant xy-plots with equal axis.

postcalc is a deferred word which can be programmed to modify simulation output.

output is a deferred word which can be set to a specific output mode and change output modes if necessary. E.G., **SET-TO output DISPLAY** to display data or **SET-TO output PLOTY** to plot data. In the latter case, **gph#** must be executed before running any simulation, using a graphics mode output.

Simulation Control

ASSIGN-IC (... y3 n3 y2 n2 y1 n1 --) Clears the initial condition vector **IC()** and assigns values to it: IC(1) = y1, IC(2) = y2, IC(3) = y3, etc.

ASSIGN-REF (--) Assigns values to reference blocks (**+REF, 0REF, &-REF)**.

SRESET (--) Resets parameters and vectors to initial states. Clears logic and state vectors, signs references, and sets initial conditions to specified state variables.

model is a deferred word +which is set to a word which defines the simulation model.

INITIALIZE (--) Initializes model block outputs to starting values by running the model twice with **DT** = 0, so that blocks other than the integrators will be set to their correct initial states.

precalc is a deferred word which may be defined to do calculations prior to beginning a simulation.

SINIT (--) Executes SRESET, precalc and INITIALIZE.

CHECK (--) Execute to check the "wiring" accuracy. This word lists the state values for all active nodes for comparison to calculated values at each node.

CONT (n --) CONTINUE runs the simulation for n communication intervals. It may be executed to continue a simulation after a run is completed. A run will pause when any key is pressed and continue when another key is pressed. Pressing **E** twice terminates the run.

SCHECK Lists current state variable values.

SRUN (n --) Simulation will run for n communication intervals. It combines **SINIT & CONT.**

XRUN (x n --) Run simulation x times for n communication intervals.

postcalc is a deferred word which is **SET-TO** a word which changes parameter values, allowing XRUN to show the effect of parameter changes on successive simulation runs.

td is a constant used to introduce **td ms** delay in running a simulation: default 0. It can be used to slow down a run.

Blocks

Conventions*:* n1 is the output node
n2, n3... are input nodes
k, if present, represents a parameter value
b1,b2... represent logic nodes

The following notation (stack diagram) is used: (k n1 n2 n3 --) where -- represents the block name. An equivalent statement is: 500 3 2 1 <block name>. All available blocks are listed according to function. All blocks must leave an empty stack after execution.

Integrators

EULER (n1 n2 --) Y(n1) = Integral[Y(n2] where Y(n1) = Y(n2)*DT + Y(n2)
This is the simplest and fastest integrator at a give DT value. However, it is the least accurate, requiring small DT values to achieve reasonable results.

TRAPZ (n1 n2 --) Y(n1) = Integral[Y(n2)] where Y(n1) = (Y_n + Y_{n+1})*DT/2. This method requires taking the average of the previous and current value of Y(n2).

integr A deferred word which can be used in place of EULER or TRAPZ in model definitions, allowing the user to change integrators without redefining a model. Simply state
SET-TO integr EULER or **TRAPZ** activates the desired integrator. This is a significant advantage of forth compared to other languages for this type of application.

Mathematical

INV (n1, n2 --) Y(n1) = -Y(n2) Inverts the sign of the input vector.

ABS (n1 n2 --) Y(n1) = Absolute Value of Y(2)

POT (n1 k n2 --) Y(n1) = k*Y(n2) k is a scalar defined in the vector K(). The default scale is #10000. It may be set to values <> 1. If k is negative, the **POT** block functions as an \ implicit inverter.

OFFSET (n1 k n2 --) Y(n1) = Y(n2) + k where k is a parameter scaled as a state variable, which may be + or -.

SUM (n1 n2 n3.... --) Y(n1) = Y(n2) + Y(n3) + ... The output vector is the sum of the inputs, where any number of inputs is allowed. Subject to overflow condition if result is >Ym. If an overflow exists, the simulation will be terminated.

MULT (n1 n2 n3 --) Y(n1) = Y(n2) * Y(n3) Two input state variables are multiplied. Also subject to an overflow condition.

DIV (n1 n2 n3 --) Y(n1) = Y(n2) / Y(n3) Division - note the order of the vectors. If Y(n3) is 0, the simulation will terminate.

SQR (n1 n2 --) Y(n1) = Y(n2) * Y(n2)

The following blocks use the FPUV3 co-processor.

SQRT (n1 n2 --) Y(n1) = Square Root[Yn2)] Simulation will terminate if Y(n2) is negative.

SIN (n1 n2 --) Y(n1) = Sine[Y(n2)]

COS (n1 n2 --) Y(n1) = Cosine[Y(n2)]

TAN (n1 n2 --) Y(n1) = Tangent[Y(n2)]

LN (n1 n2 --) Y(n1) = Natural Log[Y(n2)] Terminates if Y(n2) is negative.

LOG (n1 n2 --) Y(n1) = Log Base10[Y(n2)] Terminates if Y(n23) is negative.

EXP (n1 n2 --) Y(n1) = Exp[Y(n2)]

Input Functions (Optional)

Note: **ye, yo, slope, -slope, +slope** and **period** are constants whose values are defined using **TO**.

TIME (--) Increments DT time units at each execution, in effect, generating a linear output with a slope of 1 in 0 Y().

RNDN (n1 --) Y(n1) is a random number in the range **yo – ye**, which much must be preset.

RAMP (n1 --) Y(n1) is linear from **yo to ye** with a specified **slope**.
If Y(n1) > Y0, Y(n1) = 0. If FLG is true, Y(n1) = **ye**; if FLG false, Y(n1)=0.

>FLG is a word variable used to set a flag as on or off.

In all the following words, **the period** is set to a specific time = n* **DT**.

STEP (n1 --) Generate steps at intervals set at time = period and step height = ye.

IMPULSE (n1 --) Generate a train of impulses (width = DT) at period intervals with **ye** height.

SQWAVE (n1 --) Y(n1) is a train of square waves with a specified period and width determined by the variable **XT**, specified in units of time. and amplitude **ye** and zero-level **yo.**

>XT scalar

TRIWAVE (n1 --) Y(n1) is a repetitive triangular wave which may have different positive and negative slopes depending on the values the constants **+slope and -slope**. The period depends on the slope values, the range is 0 to Y0, **ye** sets the peak value and **yo** sets the base-level.

CSLOPE (--) This word sets the slopes determined by **+slope** and **-slope**. Execute first.

PULSE (n1 --) Y(n1) is a pulse beginning at **XT**, with width **period**, height **ye** and base **yo.**

>XT Set **XT** value.

Nonlinear Functions (Optional)

-CLIP (n1 n2 --) Y(n1) = only positive Y(n2) values.

+CLIP (n1 n2 --) Y(n1) = only negative Y(n2) values.

BANG-BANG (n1 n2 --) If Y(n2) is positive, Y(n1) = Y0, else Y(n1) = -Y0.

DEAD (k1 k2 n1 n2 --) **DEAD-ZONE** sets Y(n1) = k12 if Y(n2) >k1 and <k2.

LIMIT (k1 k2 n1 n2 --) The **LIMIT** block sets Y(n1) to k1 if Y(n2)<k1 or to k2 if Y(n2)>k2, otherwise, Y(n1) = Y(n2).

STOP (n2 n3 --) Terminate the run if Y(n2) > Y(n3) - there is no output. A *reset* is executed when a run is terminated.

DELAY (n1 n2 --) Y(n1) = Y(n2) is delayed DN*DT time units. The delay line block is emulated as a FIFO shift register with each cell delayed DT time units.

DN & **>DN** a word variable defining the number of cells in a delay line.
DLY() & **>DLY()** the delay line vector: default 256 cells.
CDELAY clears the delay line.

Logic and Logic-to-Analog Interfaces (Optional)

T/H (n1 n2 b1 --) Track/Hold block must precede an integrator. If B(b1) is true, Y(n1)=0, otherwise Y(n1) = Y(n2). The integrator output is held at its current value when Y(n1) = 0.

CMP (n2 n3 b1 --) The comparator block sets B(b1) true if Y(n2) > Y(n3), else B(b1) is false. This block must precede a SWITCH block.

SWITCH (n 1 n2 n3 b1 --) If B(b1) is true, Y(n1) = Y(n3) else Y(n1) = Y(n2). +REF, 0REF or -REF can be substituted for Y(n2) or Y(n3). IF on or off is substituted for b1, SWITCH functions ass a manual spdt switch.

SOR (b1 b2 b3 --) B(b1) = B(b2) OR B(b3)
SAND (b1 b2 b3 --) B(b1) = B(b2) AND B(b3)
SNOT (b12 b2 --) B(b1) = invert B(b2)

<u>ADDENUM</u> *(Optional)*

Blocks and commands which are not included with the basic system. These are examples of additional blocks or commands which can be added to the TACS vocabulary if needed.

FUNGEN (n1 n2 --) Y(n1) = F[Y(n2)] Function generator simulates the classic diode function generators which produce user defined wave forms used as inputs. It was used to generate log and other functions. It requires a table of slopes defined at equal break points. TACS allows up to 4 function generators (**fg1, fg2, fg3 & fg4).**

slopes is a deferred word which can be SET-TO fg1, fg2, fg3 or fg4.
fperiod & fgain - constants

Break points (maximum: 20) are set at equal intervals determined by the constant **fperiod.**

Use **>NBPT** to set the number of break points and the constant **fgain** to adjust the output level. IF NBPT is exceeded during a run, the slope will be set to 0. If n2=0, the generator will produce a time-dependent waveform, which is the normal input mode of operation. Slopes are entered in tables, eg, **TABLE fg1 100000 , 50000 , 25000 , 60000 , ... ,**

IEULER (n1 n2 --) Implicit Euler integrator.

RK2 (n1 n2 --) The 2nd order Runge-Kutta integrator corrects the Euler method, resulting greater accuracy a larger DT values.

SCHART (n --) Emulates a strip chart in graphics mode outputs.

TestInput (--) Test input functions which are **SET-TO sinput** <name>.

.

Appendix 1

uM-FPU V3.1 Mathematics Coprocessor

Initialization of the FPU requires execution the following sequence: **FPUV3 Y0 SETFPU**.

SETREGA (n --) Assigns register A as register #n: default 1.

SETREGX (n --) Assigns register X as register n: default 10.

FPUV3 (--) Initializes the SPI engine, resets the FPU chip. Starts SPI engine, sets reg A & reg X.

FP! (byte --) Loads byte on stack to FPU.

I>FP (value --) Stores 32 bit value in in reg A and converts it to floating point.

FP>I (--- value) Leaves integer form of floating point value in reg A on stack.

SETFPU (scale --) Stores scale as a fp-number in register 2.

fin (value --) A state variable is converted to floating point and divided by Y0 on the FPU.

fout (value –) The result is multiplied by Y0 and transferred to the stack as an integer.

The following commands perform the indicated functions on a state variable, leaving the result on the stack.

FSIN, FCOS, FTAN. FLN, FLOG, FLN, FEXP, FSQRT

uLCD V4.3   Serial 4.3 Display

Useful Constants

Available Colors: **black, blue, green, violet, red, orange, yellow, white**

Fonts: **font1, font2, font3**

Character Format: **default, bold, italic, inverse, underline**

ypix 480 horizontal pixels
xpix 272 vertical pixels

SD File Mode: **r, w, a**

Display Commands

ulcd (--) Initializes LCD for operation.

clr (--) clears screen and sets defaults with origin at 0,0.

mode (n --) Sets screen orientation: n is 0 Landscape 1 Landscape Reverse
2 Portrait 3 Portrait Reverse

bgrd (color --) Set screen background color.

cursor (col line --) Set cursor position.

contrast (n –) n = 0 screen blank n = 1 - 15 screen on.

origin (ypos xpos --) Set origin location: default 0,0.

Text Related Commands

txtc (color --) Set text foreground color.

txtb (color --) Set text background color.

putc (char –) Places character at current cursor position.

putstr (string --) Displays string (.) at current cursor location.

font (height width font# –) Sets text font and size where height & width are set to 1-16.

LCD (--) Make uLCD the console. Execute **CON** to return to the default console.

-

Graphics Commands

putx (color ypos xpos --) Place a pixel at xy position.

line (color y2 x2 y1 x1 --) Draw a line between x1,y1 and x2,y2 positions.

rect (color y2 x2 y1 x1 --) Draw a rectangle with opposite corners at x1,y1 and x2,y2.

pline (pattern –) Draw lines using pattern: 0 – solid, 1 -

hline (color ypos –) Draw a horizontal line at specified ypos (0-271).

vline (color xpos –) Draw a vertical line at specified xpos (0-479).

SD FAT16 File Storage Commands

mount (--) Mount a FAT16 SD-Card.

unmount (--) Dismount the SD-Card.

fopen There are 3 options :
Open a new file for storage of an image using capture -
w " file name " fopen stores file handle in variable **handle**
Open a new or current file for appending images using capture. Can store multiple images in one file.
a " file name" fopen
Open a current file for displaying stored image(s) using **display.**
r " file name" fopen

fclose (handle –) Close current open file identified by its handle.

capture (handle –) Stores current image in file identified by its handle.

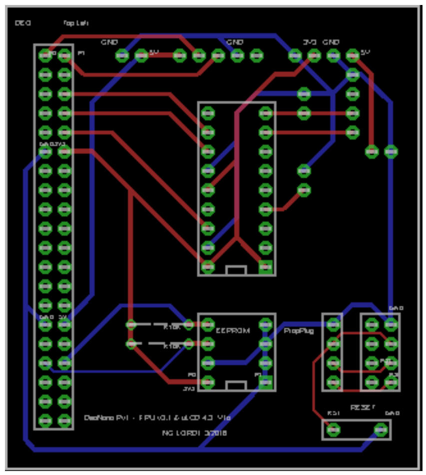
display (handle –) Displays image stored in file identified by its handle.

Appendix 2

Hardware/Software Description

1. A modified Cluso's 4 Cog P1 version was installed on a Deo Nano FPGA. All tables were removed so that a total of 64 Kb of ram memory was available.

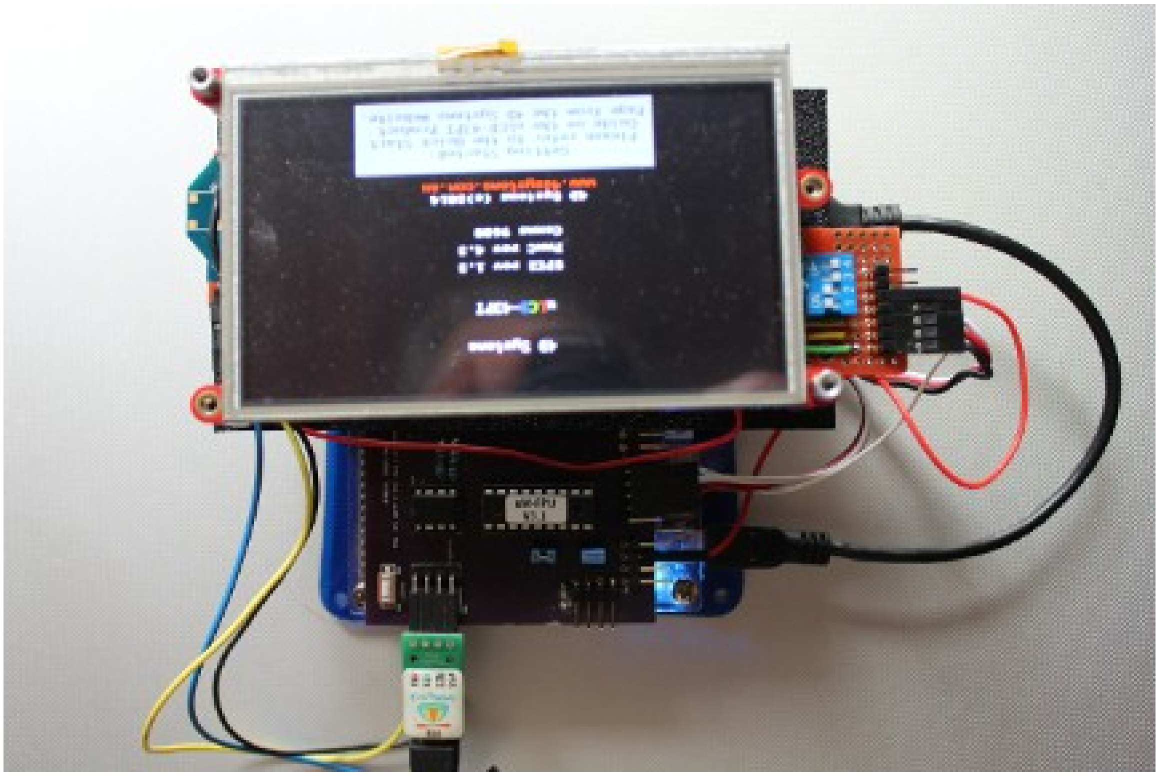
2. A special pcb was designed to access the propeller pins, add a 64 Kb eeprom, a reset button, a prop-plug connector, a fpu v3.1 to enable access to transcendental functions, a serial connection to the graphics terminal, and relevant power distributions.



3. The graphics terminal is a 4D Systems uLCD-43PT, which also includes a SD card for image storage.

4. Power was provided to the Deo Nano and uLCD using an Astro E1 5 volt (6700 mAh) brick.

The following figure shows a top view of the hardware setup. A XBEE is attached to the board beneath the LCD.   A prob plug is shown connected at the bottom. The switches on the right control power to the ULCD and XBEE.   TACS is controlled using a terminal emulation on a PC (115200 baud). I prefer to use Lubos Pekny's Forfiter, a terminal emulation plus a text editor, which is designed to simplify forth coding and testing on micro controllers.   The XBEE can be connected, instead of the prop plug, for wireless communication with a portable terminal.



The software package consists of the following components listed in the order in which they are installed on the propeller:

1. TACHYON V2.6
2. EXTEND   Modified Small Load
3. FPU V3.1
4. uLCD V4.3
5. TACSA
6. TACSB
7. TACSC
8. TACSD

TACSA : Deferred Words, System Parameters, Vectors and Arrays which are stored in the upper 32K memory, Commands to access scalars and state variables, and Utilities.

TACSB: System Messages and Output Commands.

TACSC: System Control and Required Simulation Blocks.

TACSD: Optional Blocks – Input Functions, Nonlinear Functions, Logic and Misc. Addons..