This document was last updated for Rev 3.21 code set

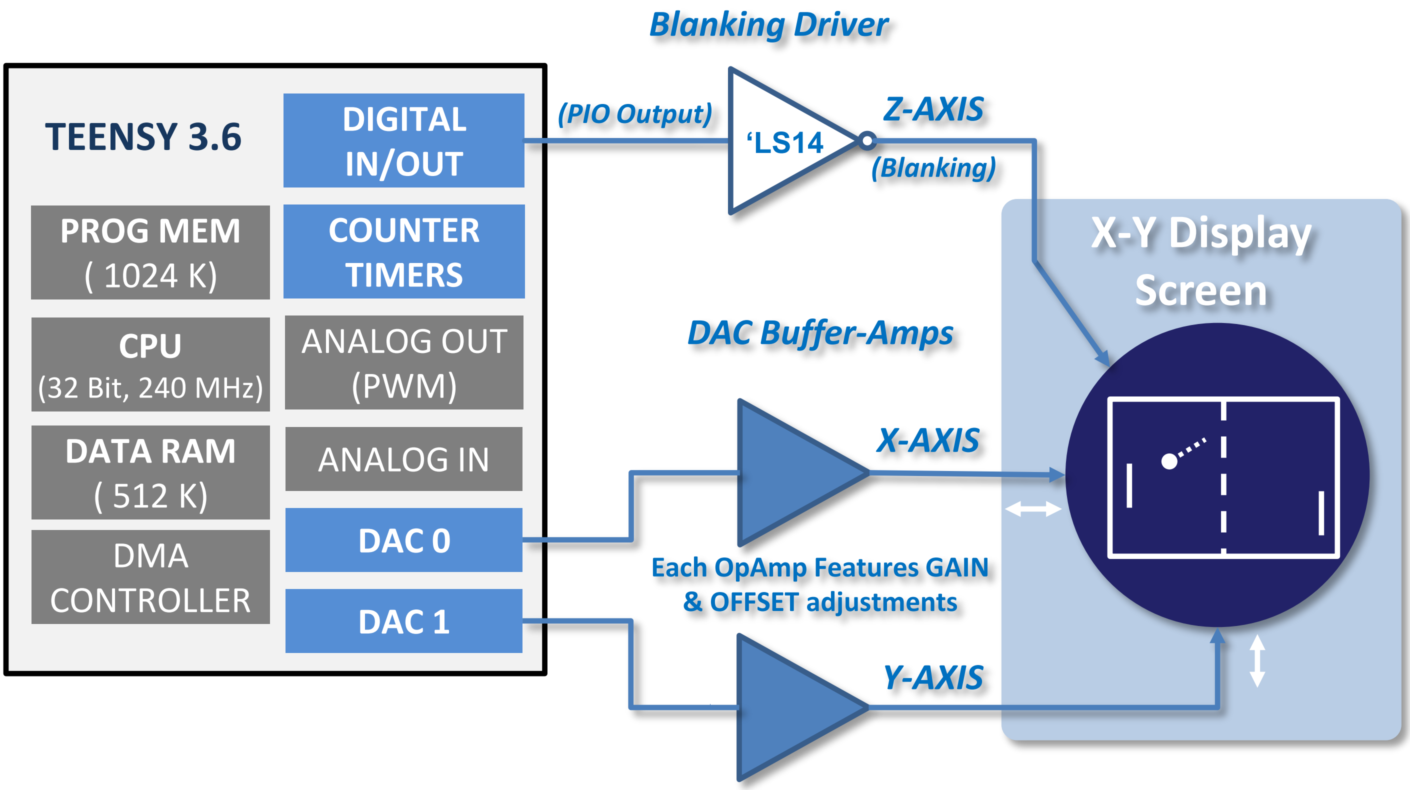
Teensy CRT_SCOPE_CLOCK User Manual

Table of Contents

Hardware/Software Overview

HARDWARE OVERVIEW

The TEENSY GRAPHICS INTERFACE (TGI) board has been designed for use with the ARDUINO TEENSY 3.6 Processor board. A block diagram of the board is shown below.



SOFTWARE OVERVIEW

The CRT_SCOPE_CLOCK program was derived from a test program used as a test bed to develop the ARDUINO GRAPHICS INTERFACE library. While it contains extensive operator I/O (via Arduino IDE Serial Monitor) to select and run many test and demonstration loops, it can also be configured to auto-boot into a CLOCK MODE (a functioning CRT CLOCK) upon power-up.

When running in CLOCK MODE, just enter ? ↵ on your Serial Monitor window. The clock display will terminate and you can begin operator interaction with the test and demo menus.

This document describes the operation of the CRT_SCOPE_CLOCK test program, concentrating on non-CLOCK-MODE interaction..

CRT_SCOPE_CLOCK Overview

The AGI Test program is called CRT_SCOPE_CLOCK. This is a variation of the original CRT_SCOPE program first developed to demonstrate the Arduino Graphics Interface project as published in the FEBRUARY & MARCH 2018 **Nuts & Volts Magazine**. Since that time, the original project was expanded and now supports versions that run on both the Arduino DUE (DMA) and Arduino TEENSY 3.6 (PIO) processors.

In addition to support for the TEENSY processor, the CRT_SCOPE_CLOCK version also contains rudimentary metrics to measures CPU performance using a Dhrystone calculation.

INSTALLATION NOTES: See next page for library (and example programs) installation notes..

CRT_SCOPE_CLOCK.ino	Arduino Test Program Mainline
dhry21a.cpp	Dhrystone CPU performance metrics & measurement routines
dhry.h	Dhrystone Compiler switches and definitions file
HersheyFontROM.h	Font File
keywords.txt	Arduino CRT_SCOPE keywords file
VectorFontROM.h	Font File
XYscope.cpp	AGI Library, Supports BOTH DUE and TEENSY 3.6 Processors
XYscope.h	XYscope Compiler and System Definition file
XYscopeConfig.h	TEENSY and DUE Specific user configuration file;. This file defines all of the timing and setup parameters used by the AGI library. User should open and edit this file to make key selections and tune the performance parameters as needed.

XYscope_V3.20.zip	ZIP file containing all of the above UNZIP this file into a directory called CRT_SCOPE_CLOCK and compile CRT_SCOPE_CLOCK using Arduino ID 1.8.5 (or higher)

Additionally, TEENSY 3.6 and/or Arduino DUE processor support and support-libraries must be installed within the Arduino IDE.

PROGRAM STARTUP

The default configuration for CRT_SCOPE_CLOCK will auto-start into the CLOCK application. In CLOCK mode, an analog clock face and hands are displayed along with optional Digital Time, Day of Week, and Date display. To enter the TEST MODE, connect to the TEENSY, open the serial monitor, and type **?↵** . This will cause the clock mode to terminate and TEST MODE to begin.

OPERATIONAL NOTES:

This program is designed to be run within the Arduino IDE environment, and interacts with the programmer by using the IDE "SERIAL MONITOR" function. The SERIAL MONITOR should be set up to run with the **No Line ending** option running at **115200 baud.** All user interaction with the program is through the MONITOR using menus and help screens as posted by CRT_SCOPE_CLOCK.

The programmer is encouraged to examine and edit the details of the CRT_SCOPE_CLOCK.ino file and their accompanying configuration files as needed to adapt and utilize the AGI library for his/her purposes.

TGI Library Structure

```
Libraries
        |_Other_Arduino_Libraries_1
        |_Other_Arduino_Libraries_2
        |_Other_Arduino_Libraries_nn
        :
        :
        |_TEENSY_XYscope
            |
            |-README.md
            |-LICENSE
            |-Library.properties
            |-.gitattributes
            |__ src
                |    |-keywords.txt
                |    |-VectorFontROM.h
                |    |-HersheyFontROM.h
                |    |-XYscope.ccp
                |    |-XYscope.h
                |    |-XYscopeConfig.h
                |
                +extra
                |    |---  20180627R0 BUILD DOC (Rev 2 TEENSY).pdf
                |    |__  PCB_GerberFiles
                |         |---  xxxxxxx.ZIP
                |
                |
                +examples
                     |__  CRT_SCOPE_CLOCK
                     |    |---  CRT_SCOPE_CLOCK.ino
                     |    |---  dhry.h
                     |    |---  dhry21a.cpp
                     |    |---  20181011 Teensy CRT_SCOPE_CLOCK User Manual (Rev 3.21).pdf
                     |
                     |__  Other Example_1
                     :    |---  OtherExample_1.ino
                     :
                     |__  Other Example_nn
                          |---  OtherExample_nn.ino
```

1. The most current **TEENSY_Xyscope** files may be downloaded from:

2. You should download and install the TEENSY_Xyscope library into the “libraries” directory of your Arduino IDE as shown above.

3. The 20180627R0 BUILD DOC (Rev nn TEENSY).pdf shows the TGI schematics, Bill of Materisls, and PCB board build, test, and adjustment instructions.

4. The 20180627R0 BUILD DOC (Rev 2 TEENSY).pdf is a detailed operators manual for the CRT_SCOPE_CLOCK program. This shows how the program can default into CLOCK-MODE or be used for TGI test/calibration as well as a demonstration program for the XYscope library.

5. The PCB_GerberFiles directory contains files from which PCB boards may be ordered from your favorite PCB board house. Alternately, blank PCB boards are available from the Nuts & Volts store

Operator Input Scheme

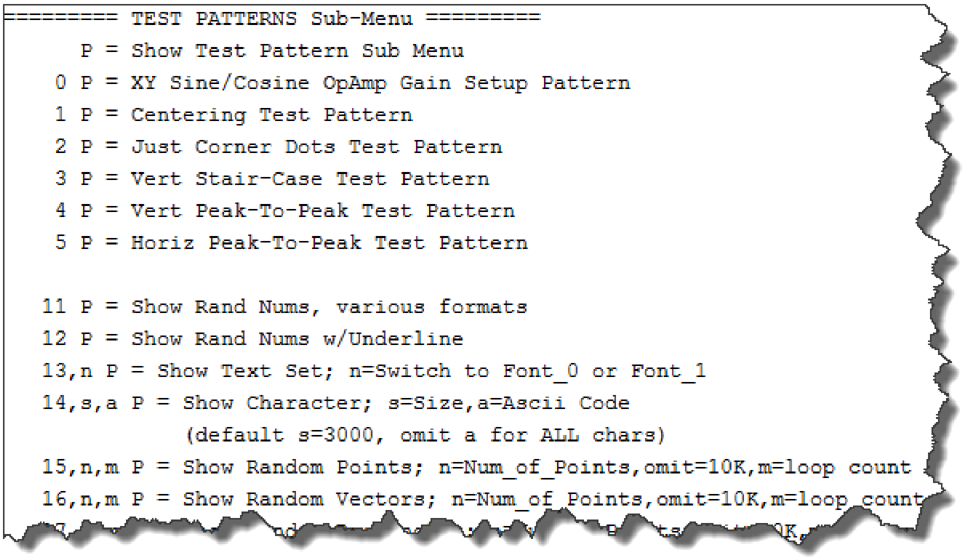
All command options consist of a single letter format followed by the ENTER (↵) key. Note, Some command letters ARE CASE SENSITIVE. For some commands, the command letter may be preceded by 0 to 5, comma-separated parameter values.

The input format is as follows: ***p1,p2,p3,p4,p5,C↵***

where : *p1 = Parameter 1 value*
p1 = Parameter 2 value
p1 = Parameter 3 value
p1 = Parameter 4 value
p1 = Parameter 5 value
C = Command Letter

Parameter values can be positive or negative numbers, integers or floating point numbers. All values will be considered integers unless a decimal point is present.

For example, the command **P** is used to plot various TEST PATTERNS onto the screen.
By entering just **P↵** (without any preceding parameters) the TEST PATTERN Sub-menu will be displayed.



```
========= TEST PATTERNS Sub-Menu =========
      P = Show Test Pattern Sub Menu
    0 P = XY Sine/Cosine OpAmp Gain Setup Pattern
    1 P = Centering Test Pattern
    2 P = Just Corner Dots Test Pattern
    3 P = Vert Stair-Case Test Pattern
    4 P = Vert Peak-To-Peak Test Pattern
    5 P = Horiz Peak-To-Peak Test Pattern

   11 P = Show Rand Nums, various formats
   12 P = Show Rand Nums w/Underline
   13,n P = Show Text Set; n=Switch to Font_0 or Font_1
   14,s,a P = Show Character; s=Size,a=Ascii Code
              (default s=3000, omit a for ALL chars)
   15,n,m P = Show Random Points; n=Num_of_Points,omit=10K,m=loop count
   16,n,m P = Show Random Vectors; n=Num_of_Points,omit=10K,m=loop count
```

Parameter Entry Example:

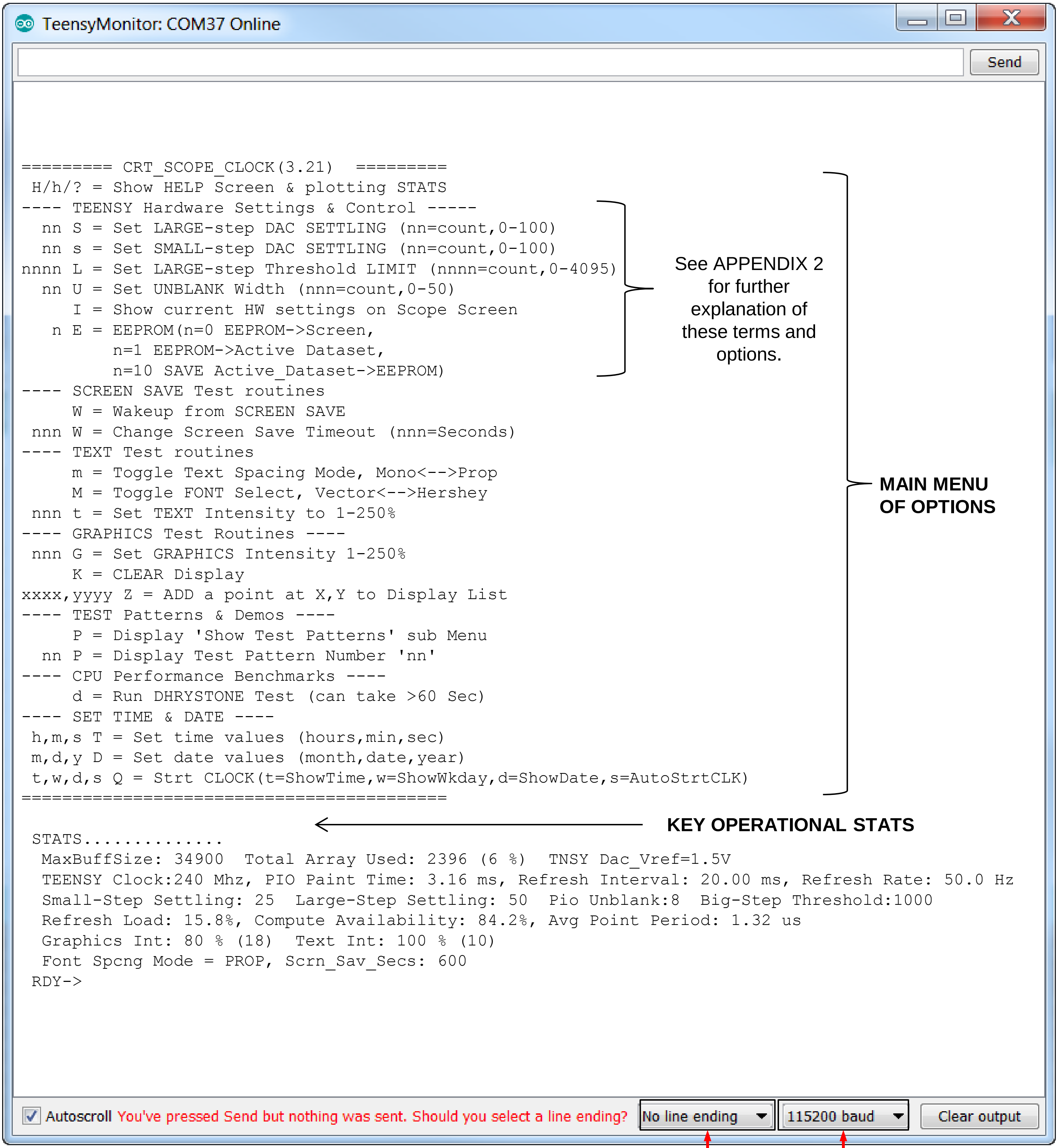
- To select and display a given pattern, Show Random Points for example, type **15 P↵**
    - This command will show a single Random Point pattern with the default value of 10,000 points.
- To display a single Random Point pattern with just 500 points, type **15,500 P↵**
- To display Multiple Random Point patterns with 500 points, repeating for 25 times, type **15,500,25 P↵**

Use the menu guides for the specific parameter descriptions and defaults for each command letter option.

Additionally, Appendix 2 provides a more details explaination for settling time and unblank time adjustment parameters.

Main Menu Display - TEENSY

To see this help screen at any time, simply type 'h', 'H', or '?' followed by return (aka: Enter).



```
========= CRT_SCOPE_CLOCK(3.21)  =========
 H/h/? = Show HELP Screen & plotting STATS
---- TEENSY Hardware Settings & Control -----
  nn S = Set LARGE-step DAC SETTLING (nn=count,0-100)
  nn s = Set SMALL-step DAC SETTLING (nn=count,0-100)
nnnn L = Set LARGE-step Threshold LIMIT (nnnn=count,0-4095)
  nn U = Set UNBLANK Width (nnn=count,0-50)
     I = Show current HW settings on Scope Screen
   n E = EEPROM(n=0 EEPROM->Screen,
         n=1 EEPROM->Active Dataset,
         n=10 SAVE Active_Dataset->EEPROM)
---- SCREEN SAVE Test routines
     W = Wakeup from SCREEN SAVE
 nnn W = Change Screen Save Timeout (nnn=Seconds)
---- TEXT Test routines
     m = Toggle Text Spacing Mode, Mono<-->Prop
     M = Toggle FONT Select, Vector<-->Hershey
 nnn t = Set TEXT Intensity to 1-250%
---- GRAPHICS Test Routines ----
 nnn G = Set GRAPHICS Intensity 1-250%
     K = CLEAR Display
xxxx,yyyy Z = ADD a point at X,Y to Display List
---- TEST Patterns & Demos ----
     P = Display 'Show Test Patterns' sub Menu
  nn P = Display Test Pattern Number 'nn'
---- CPU Performance Benchmarks ----
     d = Run DHRYSTONE Test (can take >60 Sec)
---- SET TIME & DATE ----
 h,m,s T = Set time values (hours,min,sec)
 m,d,y D = Set date values (month,date,year)
 t,w,d,s Q = Strt CLOCK(t=ShowTime,w=ShowWkday,d=ShowDate,s=AutoStrtCLK)
==========================================

 STATS..............
  MaxBuffSize: 34900  Total Array Used: 2396 (6 %)  TNSY Dac_Vref=1.5V
  TEENSY Clock:240 Mhz, PIO Paint Time: 3.16 ms, Refresh Interval: 20.00 ms, Refresh Rate: 50.0 Hz
  Small-Step Settling: 25  Large-Step Settling: 50  Pio Unblank:8  Big-Step Threshold:1000
  Refresh Load: 15.8%, Compute Availability: 84.2%, Avg Point Period: 1.32 us
  Graphics Int: 80 % (18)  Text Int: 100 % (10)
  Font Spcng Mode = PROP, Scrn_Sav_Secs: 600
 RDY->
```

☑ Autoscroll You've pressed Send but nothing was sent. Should you select a line ending? | No line ending | 115200 baud

See APPENDIX 2 for further explanation of these terms and options.

MAIN MENU OF OPTIONS

KEY OPERATIONAL STATS

Note Monitor Window Settings

MAIN MENU Details (1 of 2)

```
========= CRT_SCOPE_DHRY(3.21)  =========
 H/h/? = Show HELP Screen & plotting STATS
```

This option refreshes and displays this menu

`

```
---- TEENSY Hardware Settings & Control -----<- Options vary by CPU (TEENSY 3.6 CPU Shown)
  nn S = Set LARGE-step DAC SETTLING (nn=count,0-100)
```

This option displays and sets a DAC Settling time constant that is used when ever LARGE CHANGES in X or Y values occur. Larger SETTLING values define a longer settling time, smaller values sets a shorter settling time.

```
  nn s = Set SMALL-step DAC SETTLING (nn=count,0-100)
```

This option displays and sets a DAC Settling time constant that is used when ever SMALL CHANGES in X or Y values occur. Larger SETTLING values define a longer settling time, smaller values sets a shorter settling time.

```
nnnn L = Set LARGE-step Threshold LIMIT (nnnn=count,0-4095)
```

This option displays and sets the threshold limit the will be used to define a small VS large step change. Point data values that are LESS THAN the LIMIT are consider 'SMALL', Point data Values >= limit will be treated as LARGE steps.

```
  nn U = Set UNBLANK Width (nnn=count,0-50)
```

This option displays and sets constant that defined the Z-Axis Point-UNBLANK pulse width. Larger values set a longer pulse width, smaller values set a shorter width.

```
     I = Show current HW settings on Scope Screen
```

This option displays selected timing parameters directly onto the XYZ display screen

```
   n E = EEPROM(n=0 send EEPROM to the Screen,n=1 make EEPROM values ACTIVE,
         n=10 SAVE currently active_Dataset to EEPROM)
```

This option Displays, Restores, or Updates (SAVES) timing setups & parameters. Values affected are: Small-Step count , Large-Step count , Threshold, Unblank count , CLOCK display,& CLOCK Auto-startup

```
---- SCREEN SAVE Test routines
     W = Wakeup from SCREEN SAVE
```

When ever the screen saver timer 'times-out' the screen will blank out. Use this option to 'wakeup' the display from a screen save timeout.

```
 nnn W = Change Screen Save Timeout (nnn=Seconds)
```

Upon power up, the screen save time defaults to 10 minutes (600 Sec). Use this option to change the screen save time out period.

```
---- TEXT Test routines
     m = Toggle Text Spacing Mode, Mono<-->Prop
```

Use this option to 'toggle' back and forth between MONO and PROPORTIONAL spaced text characters. Changing this parameter affects NEW TEXT plotted into the XYlist buffer but will not affect existing text in the display buffer.

```
     M = Toggle FONT Select, Vector<-->Hershey
```

Use this option to 'toggle' back and forth between the two available text fonts. Changing this parameter affects NEW TEXT plotted into the XYlist buffer but will not affect existing text in the display buffer. Note, if only one font is defined 'active' within the XYscopeConfig.h file, this menu option will not be displayed.

```
 nnn T = Set TEXT Intensity to 1-250%
```

Use this option to set the brightness (aka 'intensity') of text characters. Changing this parameter affects NEW TEXT plotted into the XYlist buffer but will not affect existing text in the display buffer.

MAIN MENU Details (2 of 2)

```
---- GRAPHICS Test Routines ----
 nnn G = Set GRAPHICS Intensity 1-250%
```

Use this option to set the brightness (aka 'intensity') of text characters. Changing this parameter affects NEW TEXT plotted into the XYlist buffer but will not affect existing text in the display buffer.

```
     K = CLEAR Display
```

Use this option to clear the display.

```
xxxx,yyyy Z = ADD a point at X,Y to Display List
```

Use this option to add a new point into the display list at location (X,Y)

```
---- TEST Patterns & Demos ----
     P = Display 'Show Test Patterns' sub Menu
```

Use this option to display the TEST PATTERNS sub menu (see next page).

```
  nn P = Display Test Pattern Number 'nn'
```

Enter a test pattern number followed by a "P" to select and display one of the available test patterns.

```
---- CPU Performance Benchmarks ----
     d = Run DHRYSTONE Test (can take >60 Sec)
```

*Enter a '**d** ↵' to run the performance measurement test. Note, depending on the current CPU and the Refresh-load (i.e.: how many points are currently being displayed), this test can take a long time to complete (>30 Seconds). The results of this test are displayed and shown in a CSV format that may be copied out of the monitor display screen and pasted into another program (i.e.: Excel) for graphing and analysis.*

DHRYSTONES EXECUTION OUTPUT EXAMPLE

```
RDY-> d
Board=TEENSY_3.6 (CPU=240 Mhz, F_BUS=60 Mhz)
Execution starts, 2000000.00 runs through Dhrystone...
======== DHRYSTONE SUBROUTINE RUN ==========
Dhrystone Benchmark, Version 2.1 (Language: C)
Execution COMPLETED!
Microseconds for one run through Dhrystone: 2.16
Dhrystones per Second: 462912.71
VAX MIPS rating = 263.47, CSV Dump:
CPU_Mhz,DS_Time_us,DS_Per_Sec,Vax_MIPS,Bus_Mhz,
      actual_PaintTimeMs,NumPoints,
      RefreshPeriodMs,AvgPointPeriodUs
240,2.16,462912.71,263.47,60,6.91,5525,20.00,1.25
```

< - - - Results Summary

< - - - CSV Header Row

< - - - CSV Data Row

Note: Not all data is output when running a DUE CPU.

```
---- SET TIME & DATE ----
 h,m,s T = Set time values (hours,min,sec)
 m,d,y D = Set date values (month,date,year)
```

Use these options to set the TEENSY Real Time Clock (RTC) time parameters. When BAT1 (CR2032) is installed into the TGI, the clock time and date should be maintained even when power is off.

```
t,w,d,s Q = Strt CLOCK(t=ShowTime,w=ShowWkday,d=ShowDate,s=AutoStrtCLK)
```

This routine starts CLOCK-MODE. Use t, w, d, s to set desired format (1=YES, 0=NO).

EXAMPLE: **0,1,1,1Q↵** *will enter CLOCK-MODE display Day of Week and DATE (Digital time will not show).*

Note 1: The when s=1 and change is saved to EEPROM, unit will auto-start into CLOCK-MODE at next power cycle. If s=0 and saved to EEPROM, unit will NOT auto-start to CLOCK-MODE.

Note2 : USE **10E↵** *function to SAVE these configuration values to EEPROM or else changes will NOT be retained through the next power ON-OFF cycle.*

```
==========================================
```

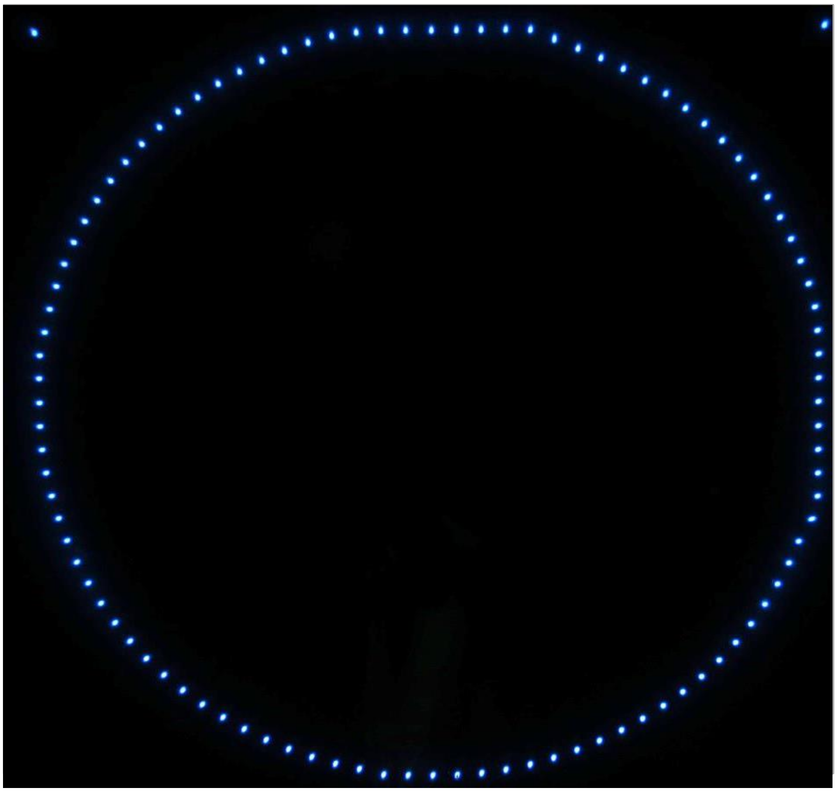
TEST PATTERN SUB-MENU

```
========= TEST PATTERNS Sub-Menu =========
      P = Show Test Pattern Sub Menu
    0 P = XY Sine/Cosine OpAmp Gain Setup Pattern
    1 P = Centering Test Pattern
    2 P = Just Corner Dots Test Pattern
    3 P = Vert Stair-Case Test Pattern
    4 P = Vert Peak-To-Peak Test Pattern
    5 P = Horiz Peak-To-Peak Test Pattern

   11 P = Show Rand Nums, various formats
   12 P = Show Rand Nums w/Underline
   13,n P = Show Text Set; n=Switch to Font_0 or Font_1
   14,s,a P = Show Character; s=Size,a=Ascii Code
              (default s=3000, omit a for ALL chars)
   15,n,m P = Show Random Points; n=Num_of_Points,omit=10K,m=loop count
   16,n,m P = Show Random Vectors; n=Num_of_Points,omit=10K,m=loop count
   17,n,m P = Show Random Rectangles; n=Num_of_Points,omit=10K,m=loop count
   18,n,m P = Show Random Circles; n=Num_of_Points,omit=10K,m=loop count
   19,n,m P = Show Random Ellipses; n=Num_of_Points,omit=10K,m=loop count

   20 P = Demo: Animated Logo Plot
   21 P = Demo: AGI Coordinate System
   22 P = Demo: Graphics Plot
   23,P = BEGIN: Clock Mode (Note: type a key to leave Clock-Mode)
   24 P = Demo: PONG
   25,g,r,s P = Demo: 5m Ball Drop(g=gravity m/s/s,r=restitution %,s=speed %
   26 P = Demo:Happy Holidays from Nuts & Volts!
   27,s,a P = Demo:Fractal Tree, s=size(500-2000),a=branch angle (.2-.7)
==========================================
```

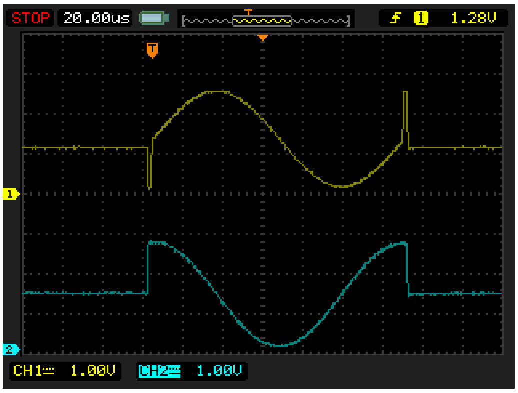
The details for each test pattern option are provided on the following pages.

Test Pattern Options (1 of 12)

0 P = XY Sine/Cosine OpAmp Gain Setup Pattern

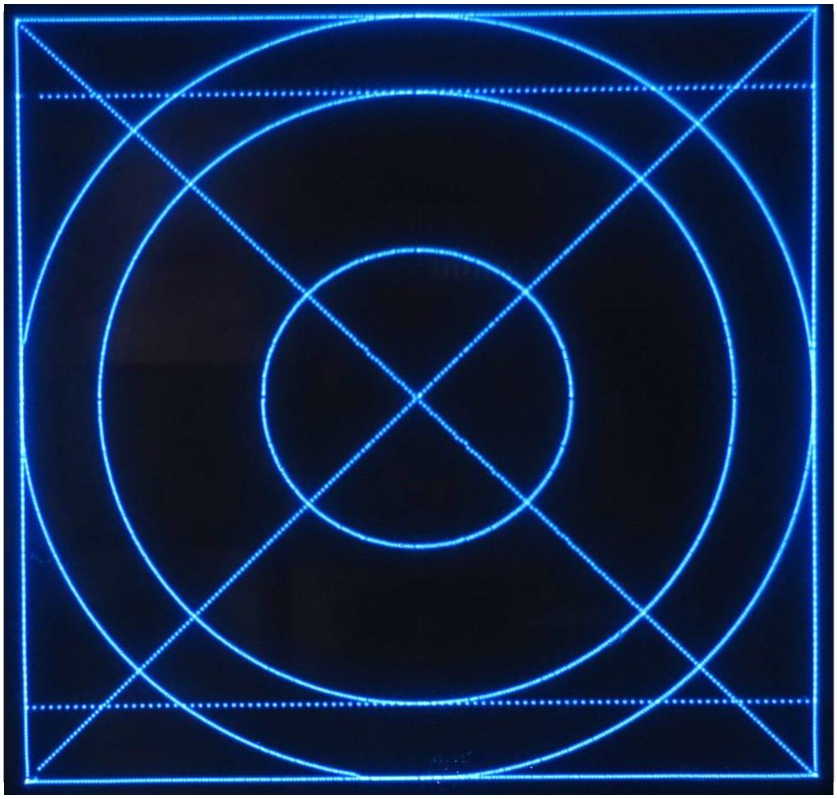


This pattern loads a SINE/COSINE pattern into the XYlist array. These waveforms may be evaluated on an oscilloscope (running in it's normal mode!) to set amplifier gain and offset.



CH1 1.00V CH2 1.00V

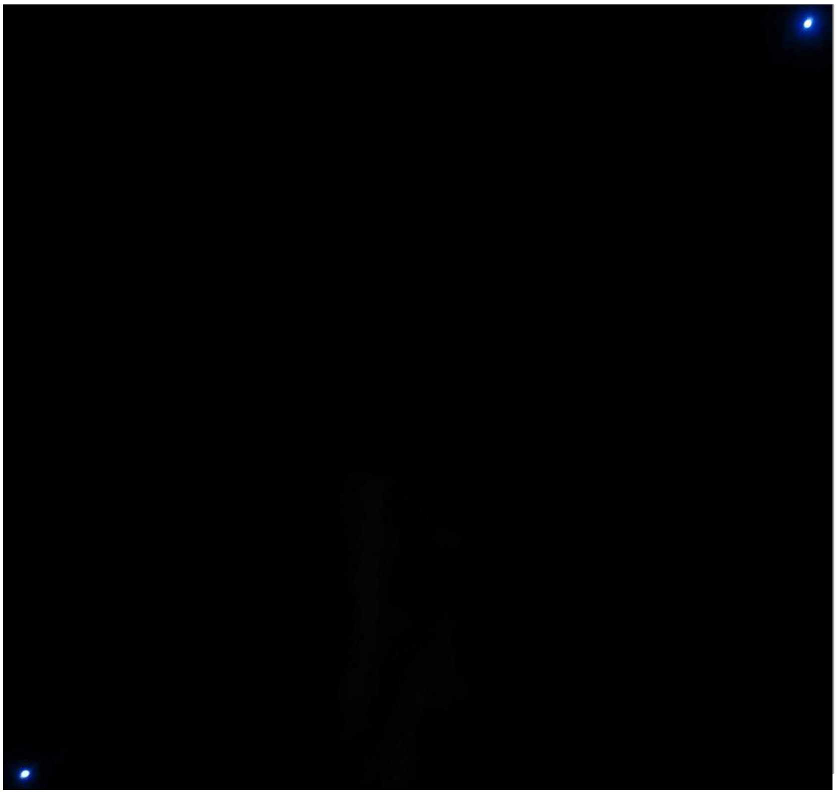
1 P = Centering Test Pattern



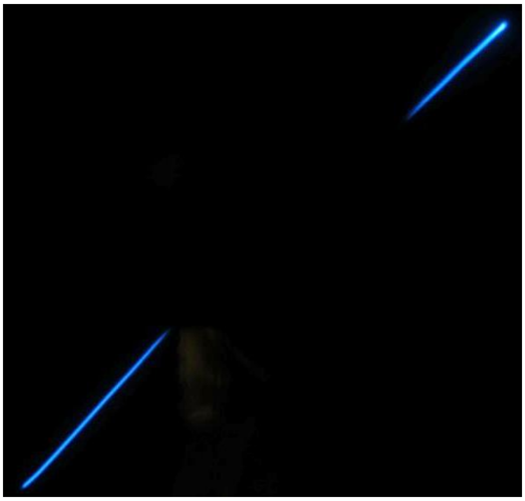
This pattern may be used to quickly adjust the gain and offset. Adjust X and Y channels so that the 'circle' is 'circular'. The outer rectangle defines the outer bounds of the whole plotting space.

Test Pattern Options (2 of 12)

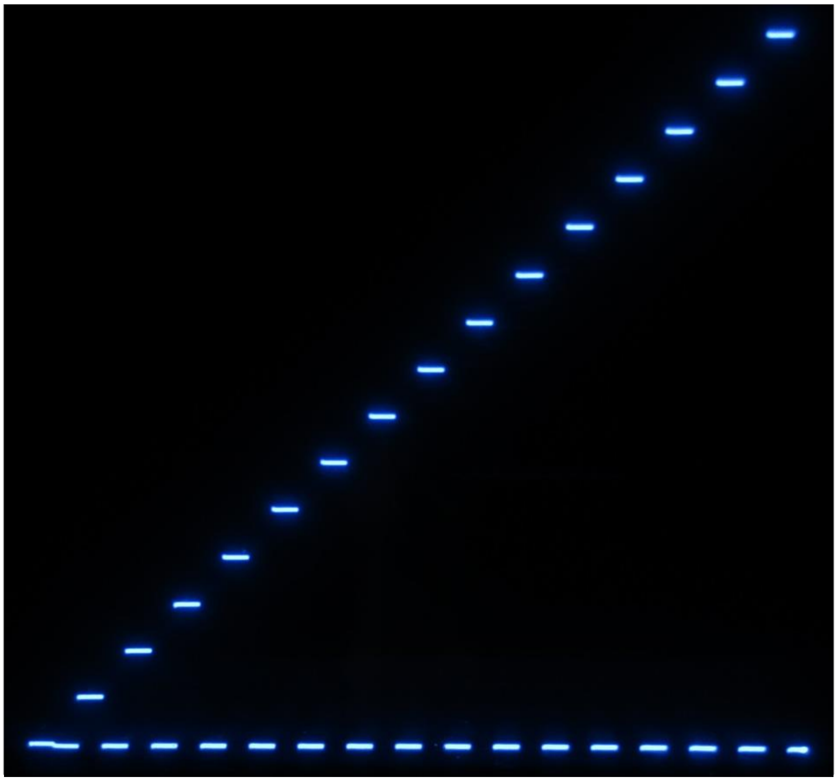
2 P = Just Corner Dots Test Pattern



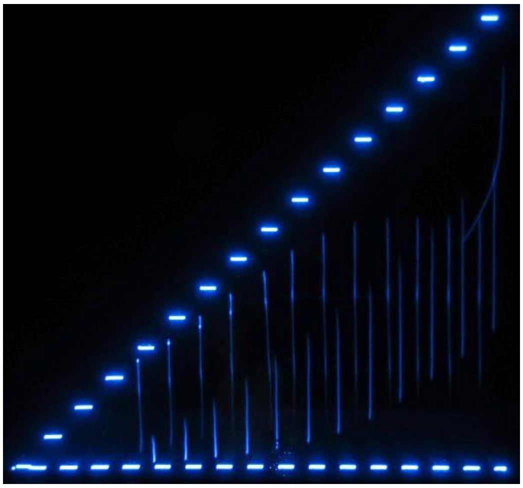
This pattern displays just a couple of 'dots', one in the lower left hand corner (0,0) and one in the upper right hand corner (4095,4095). It can be used to check the settling time and blanking pulse settings. If the settling time is too short, the dots will blur or show 'tails' such as seen below.



3 P = Vert Stair-Case Test Pattern

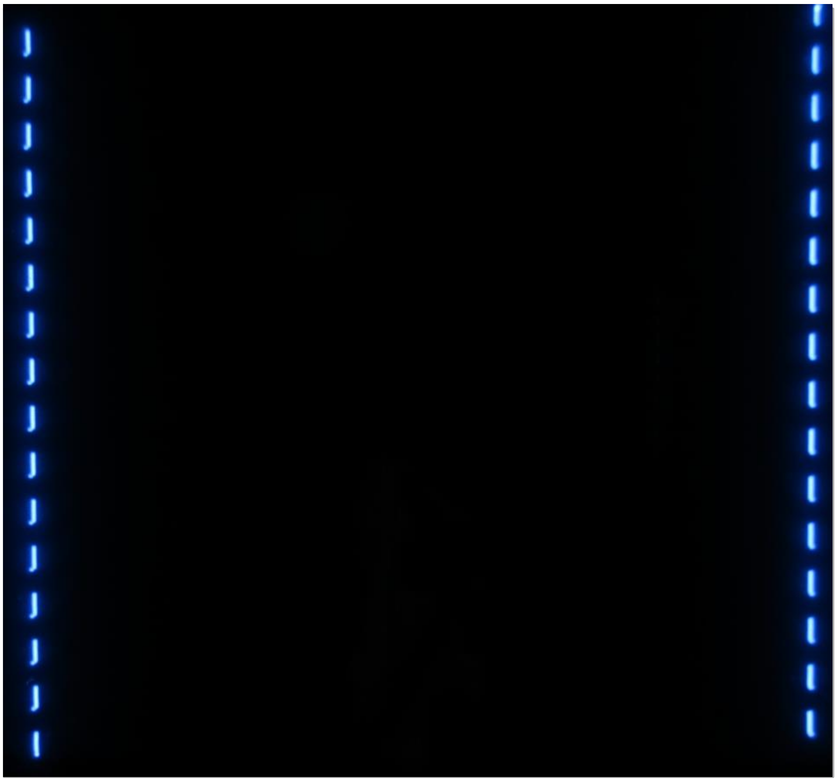


This pattern is used to evaluate and set the BIG SETTLING TIME, SMALL SETTLING TIME, BIG THRESHOLD LIMIT and UNBLANK pulse-width settings. Each step is 255 counts different than the next adjacent stair case step. When the SETTLING TIME values are too short, 'tails' appear on the steps as seen below:



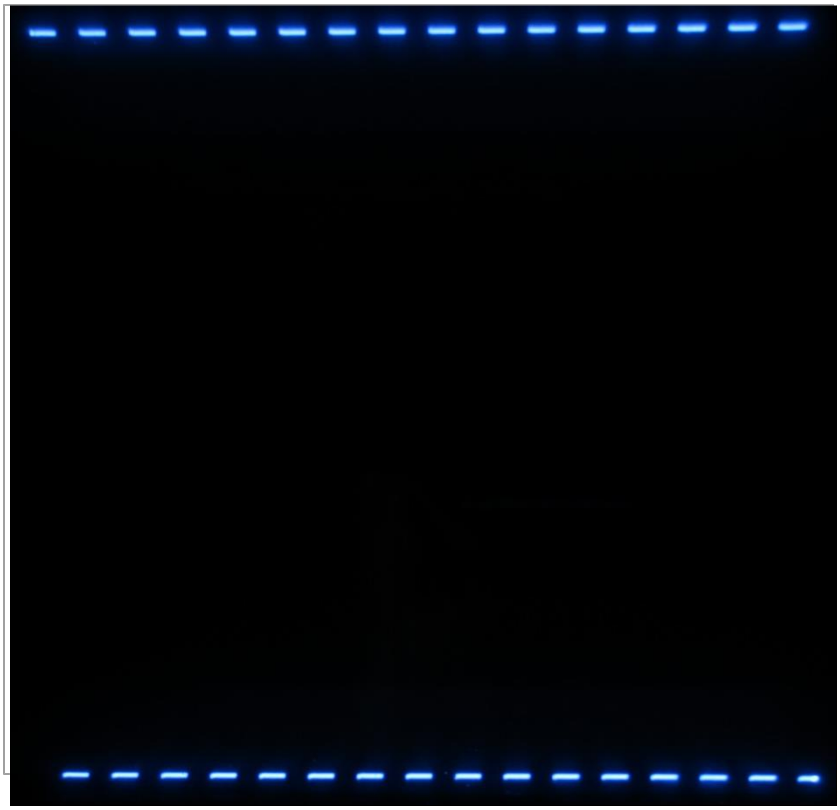
Test Pattern Options (3 of 12)

4 P = Vert Peak-To-Peak Test Pattern



This pattern is used to evaluate and check the SETTLING TIME values.   A full scale deflection from 0 to 4095 occurs on each horizontal jump. Improper adjustment will show small tails or distortions at each edge of the dashed –lines.

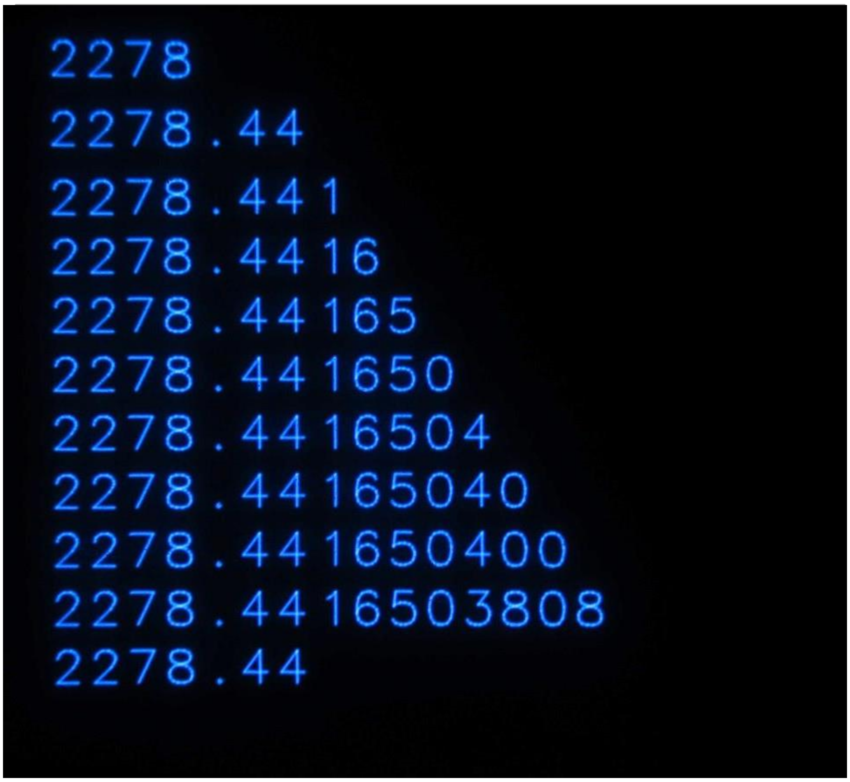
5 P = Horiz Peak-To-Peak Test Pattern



This pattern is used to evaluate and check the SETTLING TIME values.   A full scale deflection from 0 to 4095 occurs on each vertical jump. Improper adjustment will show small tails or distortions at each edge of the dashed –lines.

Test Pattern Options (4 of 12)

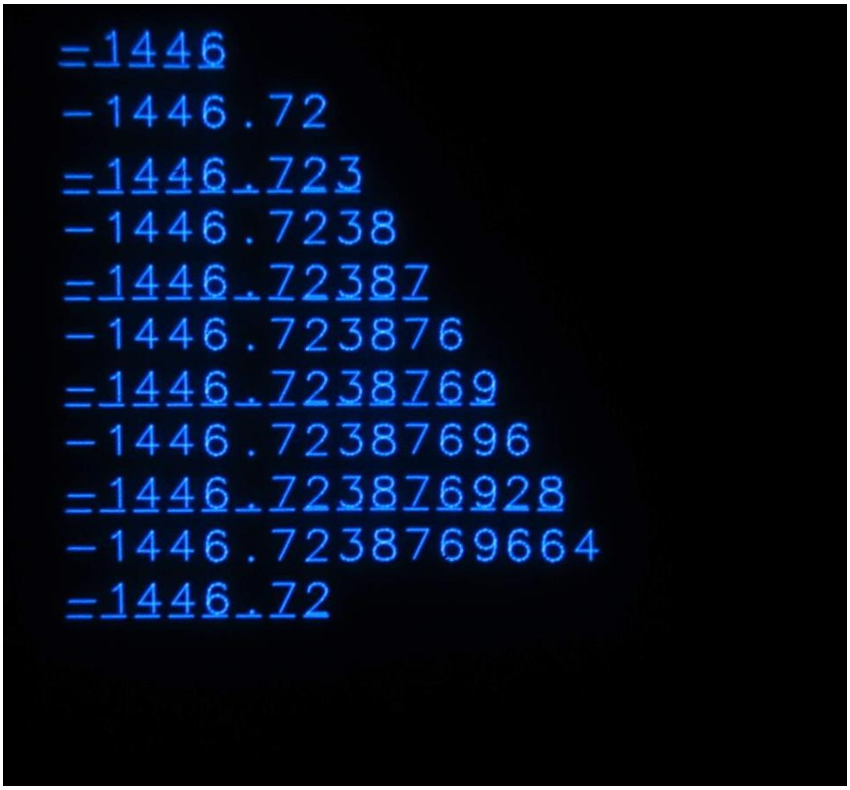
11 P = Show Rand Numbs, various formats



```
2278
2278.44
2278.441
2278.4416
2278.44165
2278.441650
2278.4416504
2278.44165040
2278.441650400
2278.4416503808
2278.44
```

This pattern is displays a random number in various formations. The programmer is encouraged to look through this code block to see the various calls can be used to show different number display formats.

12 P = Show Rand Numbs w/Underline

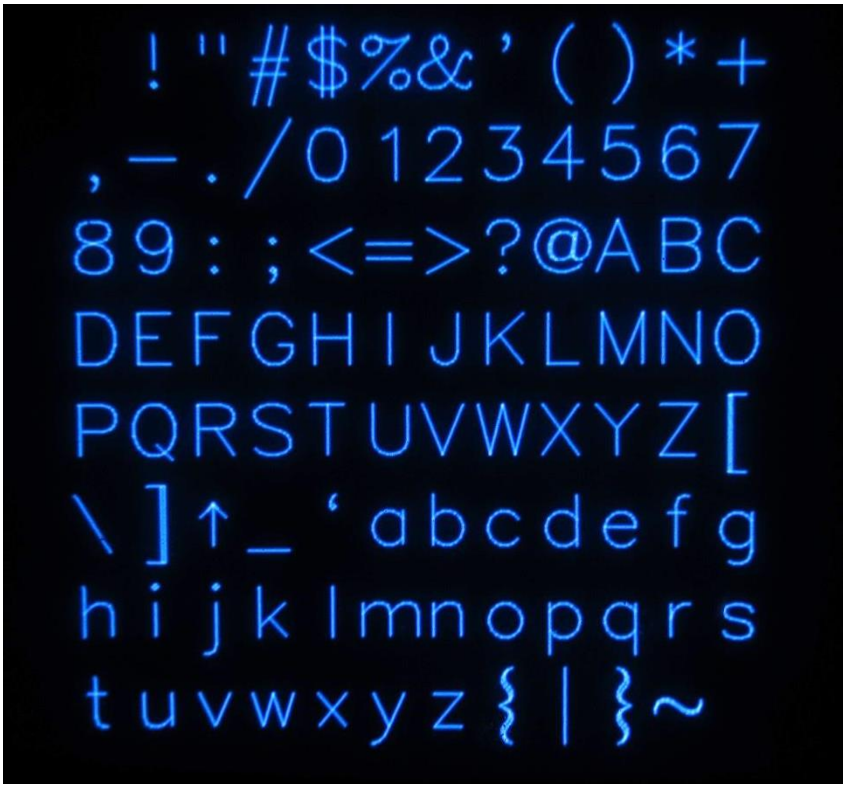


```
-1446
-1446.72
-1446.723
-1446.7238
-1446.72387
-1446.723876
-1446.7238769
-1446.72387696
-1446.723876928
-1446.7238769664
-1446.72
```

This pattern is displays a random number in various formations, every other line displayed with underlined characters.

Test Pattern Options (5 of 12)

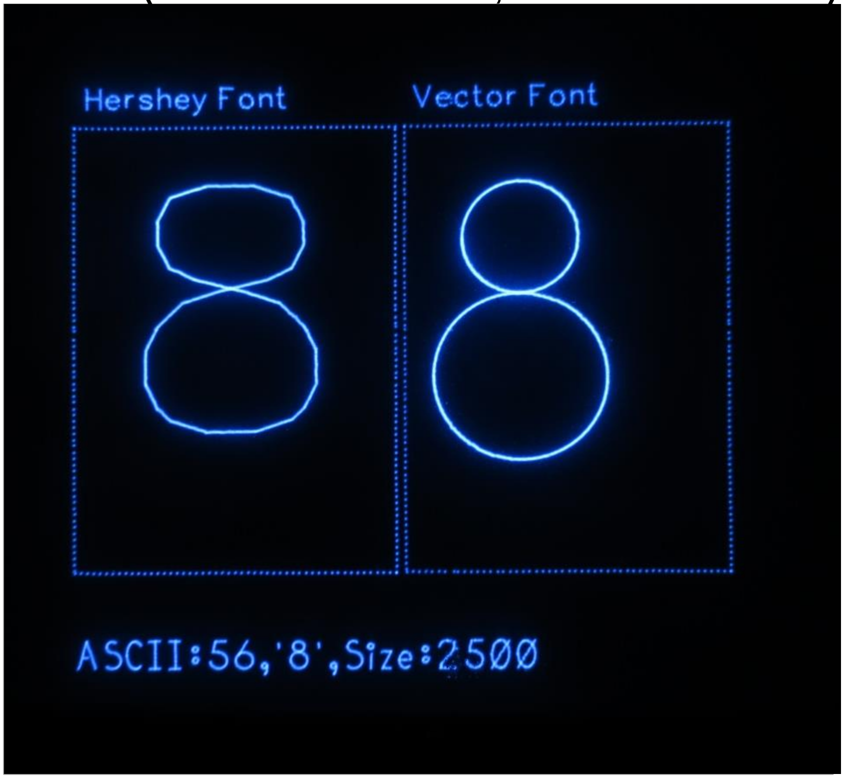
13,n P = Show Text Set; n=Switch to Font_0 or Font_1



This pattern is displays the whole character set. Use option value ‘n’ to select the VECTOR (0) or HERSHEY (1) font style.

Configuration Note: If only one font is configured (see XYscopeConfig.h), font switch is inactive.

14,s,a P = Show Character; ssss=Size,aaa=Ascii Code
(omit ssss and size=3000, omit aaa for ALL chars)



There are two text character sets accessible within XYscope. Use optional data entry values s & a to show a specific character size (height) (‘s’), or to select just one ACII code (‘a’) character for display.

If parameter ‘a’ is omitted, the whole character set is displayed, stepping from one to the other at a 1 second pace.

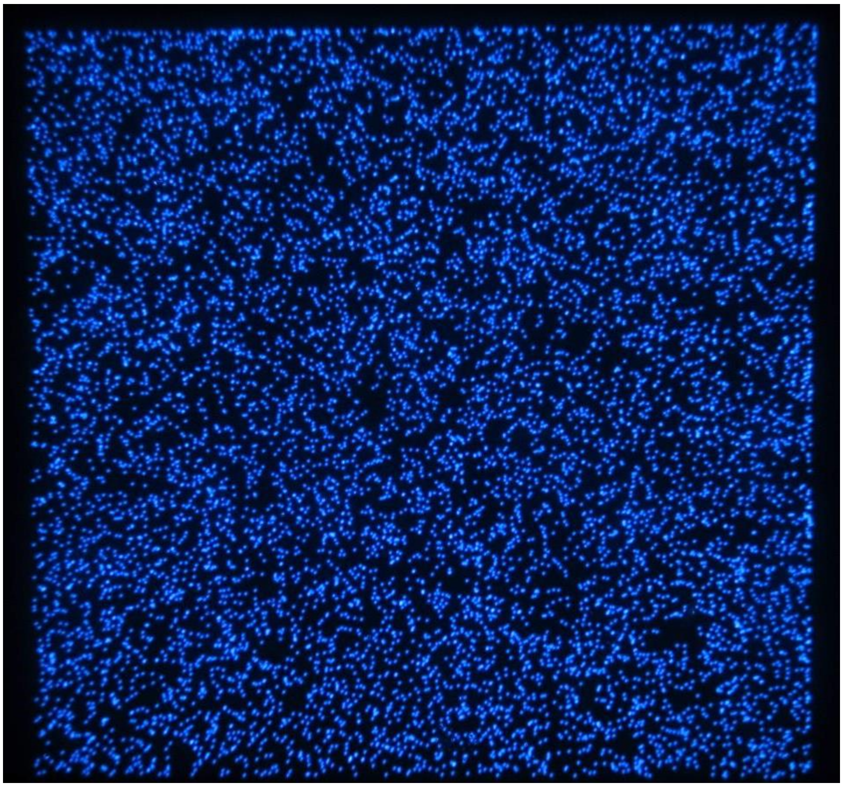
You may interrupt the display sequence at any time by entering any character on the keyboard.

Note: The HERSHEY font set is plotted using connected, straight-line vectors.

The VECTOR font uses both vectors and arcs to form the characters.
While taking longer to draw each character into the XYlist[] array, larger VECTOR font characters do not demonstrate the vertices of joined lines as will appear using the HHERSHEY font.

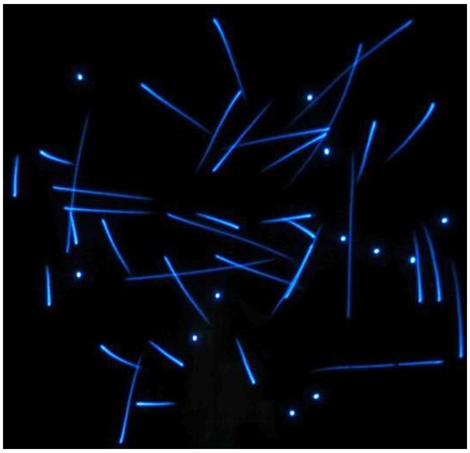
Test Pattern Options (6 of 12)

15,n,m P = Show Random Points; n=Num_of_Points,omit=10K,m=loop count

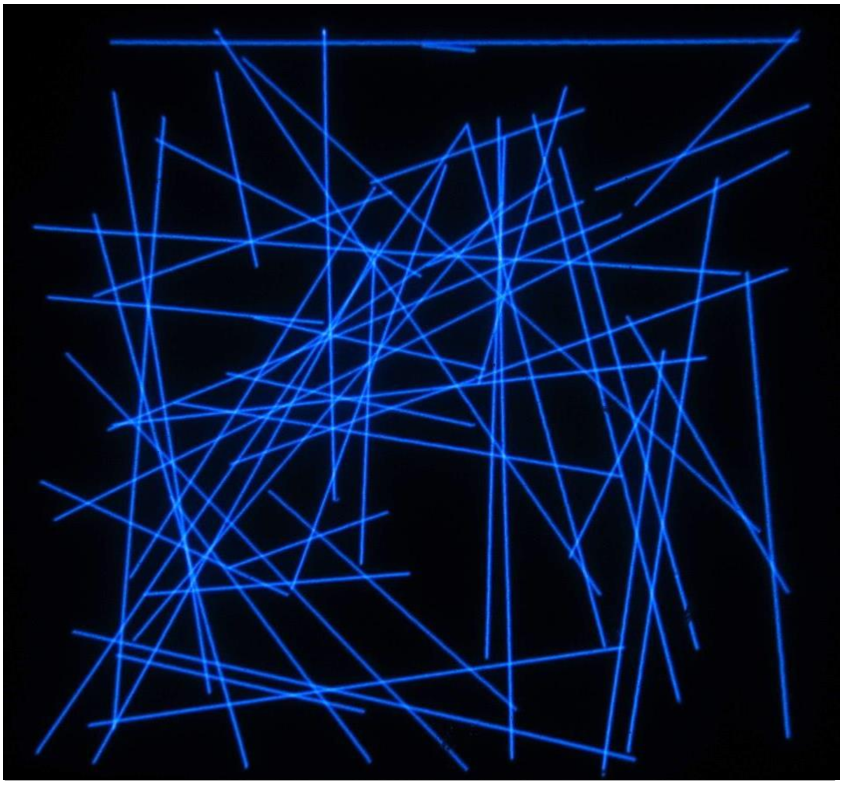


This option displays an array of random x-y points. Optional variable ‘n’ sets the total number of points to plot. Optional variable ‘m’ defines a loop count for repeated pattern display.

Plot a small # of points (i.e.: 50) to check SETTLING TIME setup. When the SETTLING TIME values are too small, ‘tails’ appear on the points as seen below:



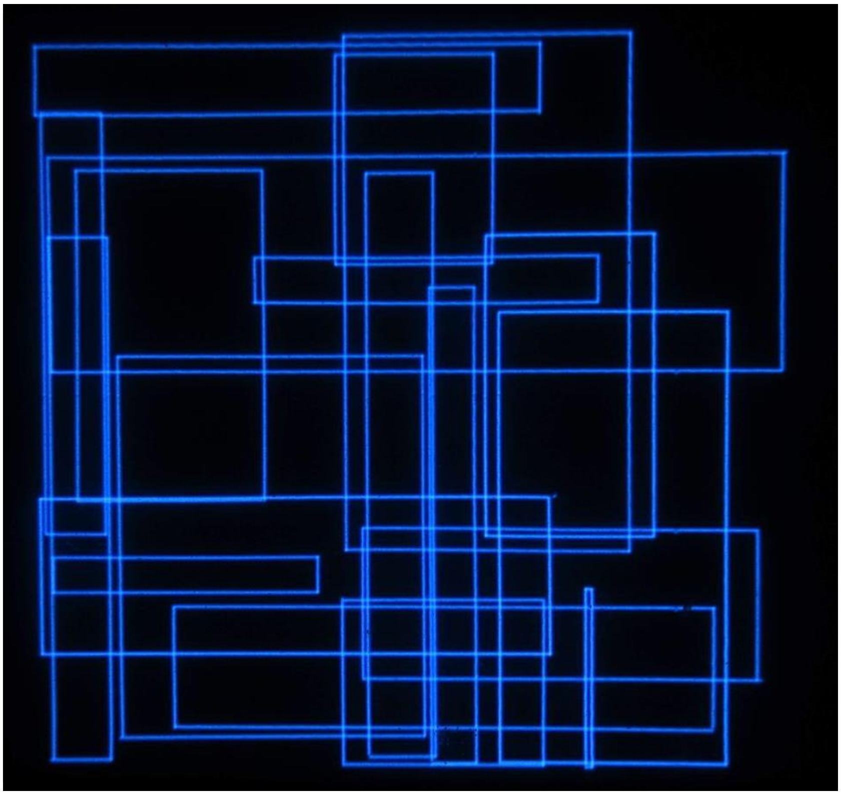
16,n,m P = Show Random Vectors; nnnnn=Num_of_Points,omit=10K,m=loop count



This option displays an array of random x-y vectors. Optional variable ‘n’ sets the total number of points to plot (not the number of lines). Optional variable ‘m’ defines a loop count whereby repeated patterns of random vectors are plotted ‘m; times. The loop can be interrupted at any time if the operator enters any character.

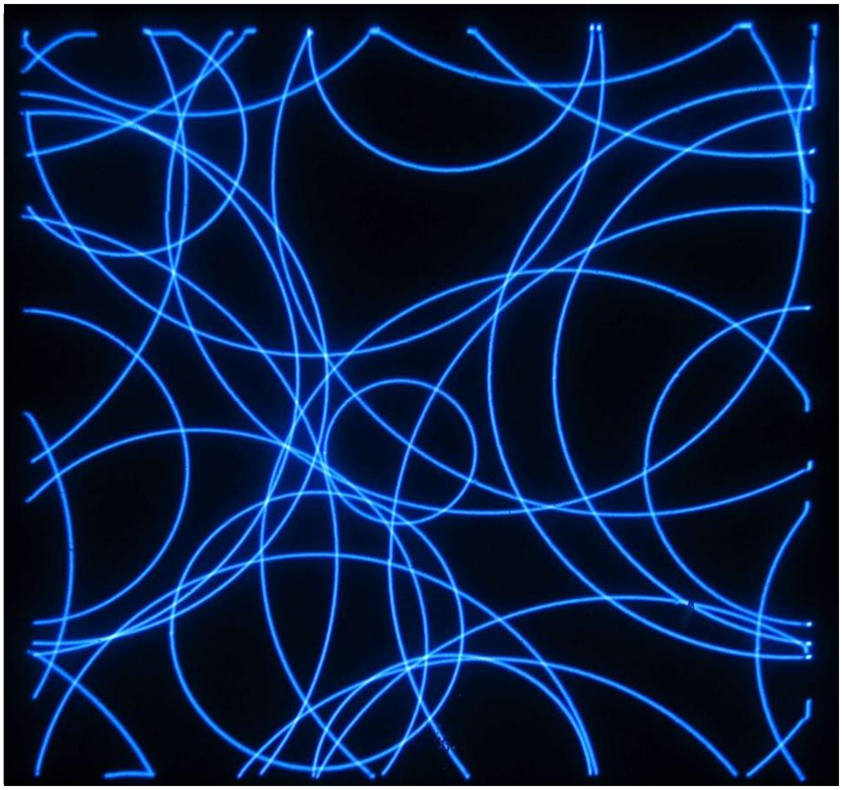
Test Pattern Options (7 of 12)

15,n,m P = Show Random Points; n=Num_of_Points,omit=10K,m=loop count



This option displays an array of random x-y rectangles. Optional variable 'n' sets the total number of points to plot (not the number of rectangles). Optional variable 'm' defines a loop count whereby repeated patterns of random rectangles are plotted 'm; times. The loop can be interrupted at any time if the operator enters any character.

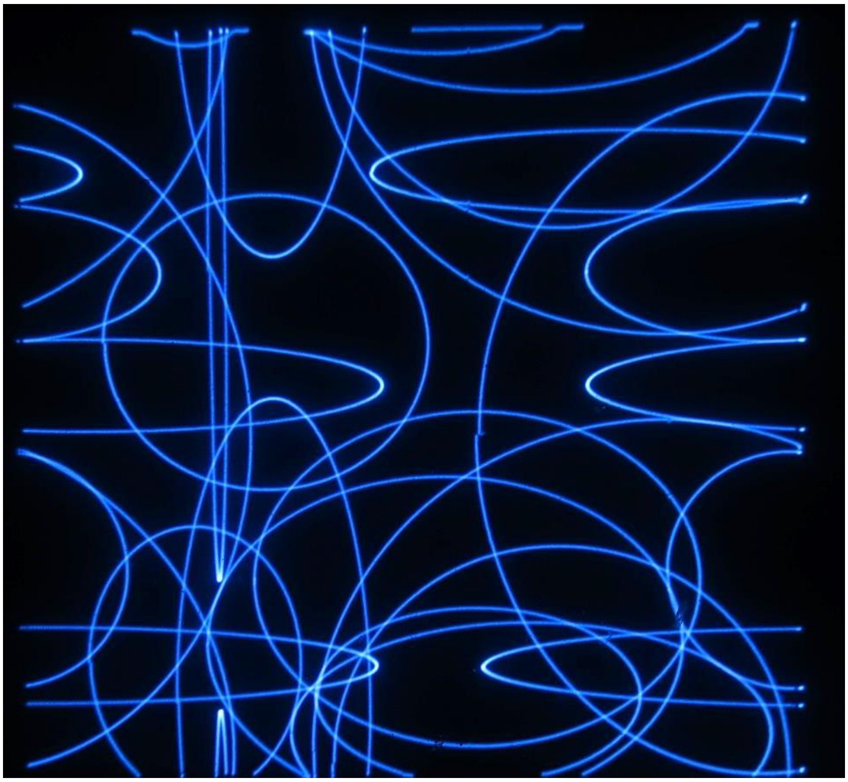
16,n,m P = Show Random Circles; nnnnn=Num_of_Points,omit=10K,m=loop count



This option displays an array of random x-y circles. Optional variable 'n' sets the total number of points to plot (not the number of circles!). Optional variable 'm' defines a loop count whereby repeated patterns of random circles are plotted 'm; times. The loop can be interrupted at any time if the operator enters any character.

Test Pattern Options (8 of 12)

19,n,m P = Show Random Ellipses; nnnnn=Num_of_Points,,m=loop countomit=10K

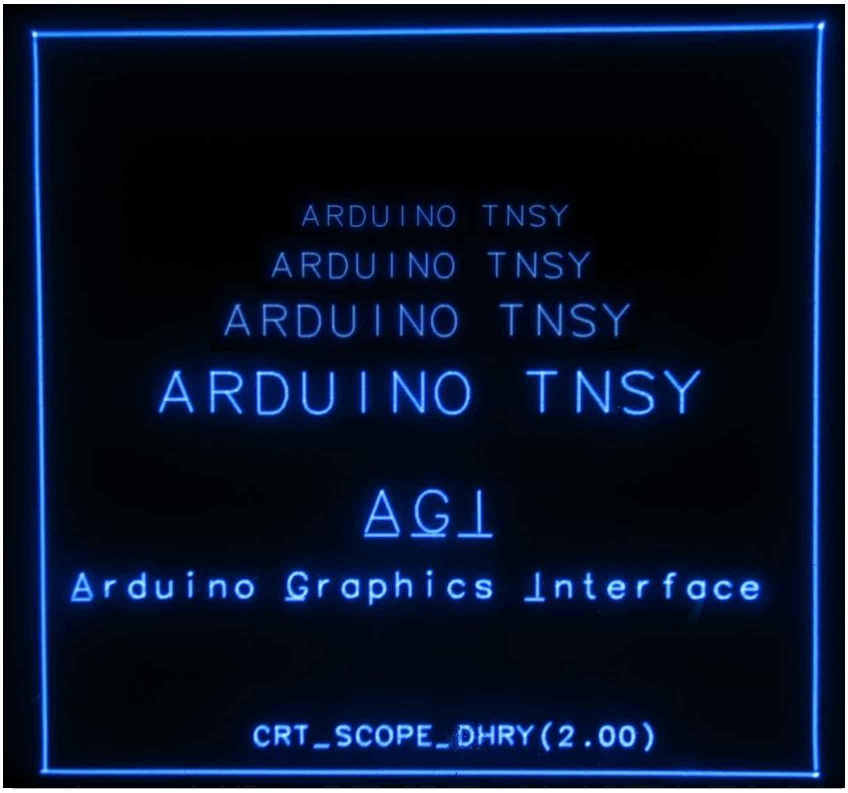


This option displays an array of random x-y rectangles. Optional variable ‘n’ sets the total number of points to plot (not the number of ellipses). Optional variable ‘m’ defines a loop count whereby repeated patterns of random ellipses are plotted ‘m; times. The loop can be interrupted at any time if the operator enters any character.

.

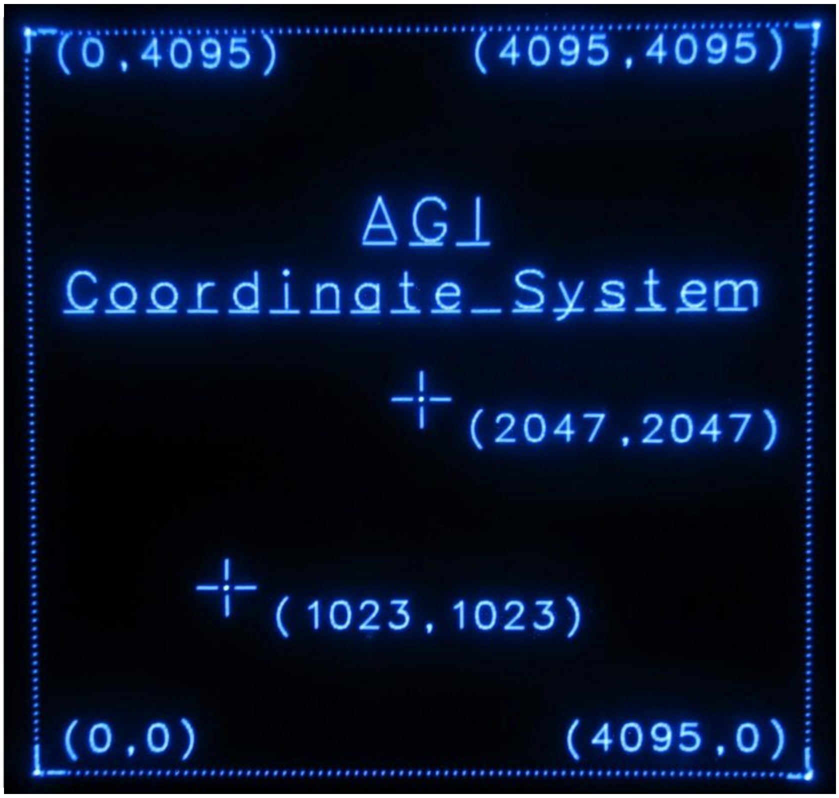
Test Pattern Options (9 of 12)

20 P = Demo: Animated Logo Plot



A simple animation.

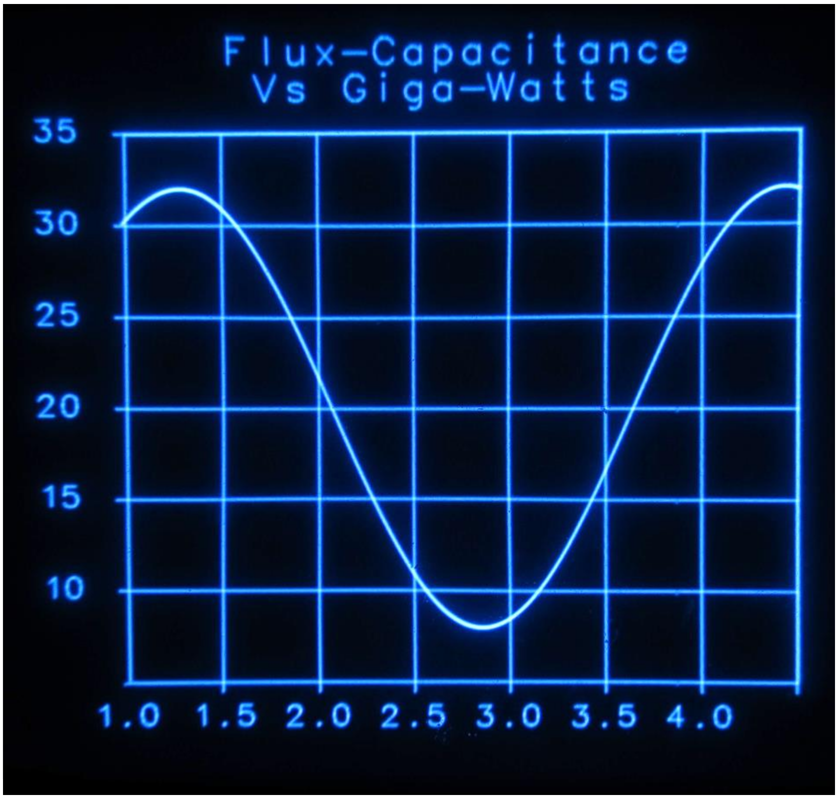
21 P = Demo: AGI Coordinate System



The AGI coordinate system/.

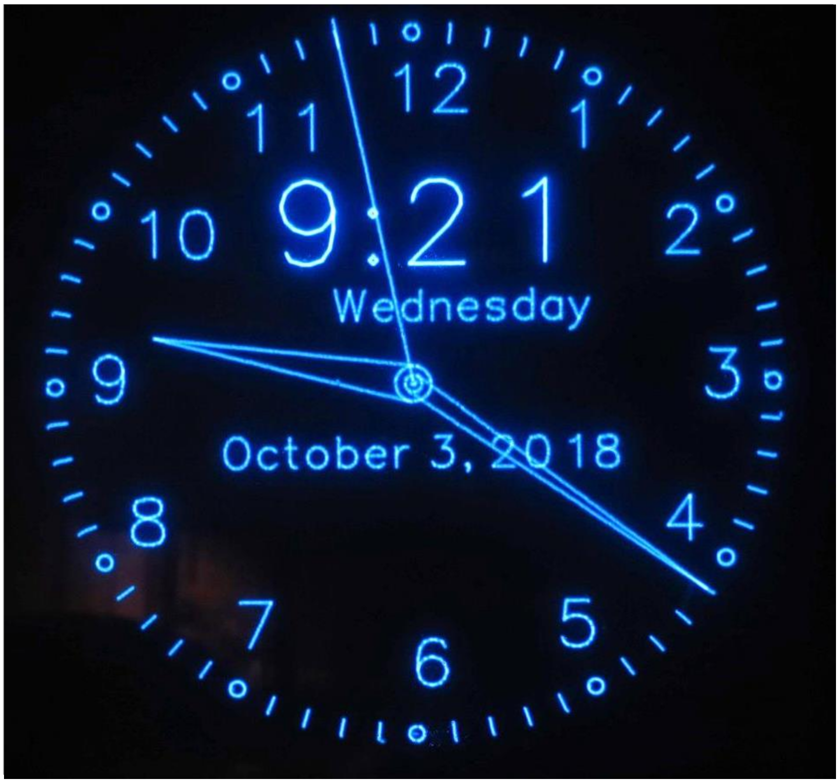
Test Pattern Options (10 of 12)

22 P = Demo: Graphics Plot



A demonstration of a sample graphics plot..

23P = ENTER CLOCK MODE



This option causes the demo program to enter CLOCK MODE. *(The main program can be set to auto start into CLOCK MODE at power up as well)*

CLOCK MODE displays the current time and date as last set (or as retrieved from the TEENSY RTC upon power-up). See APPENDIX 1 to wire in a 4-button control panel that can be used to enter a button based CLOCK SET mode (pg 22).

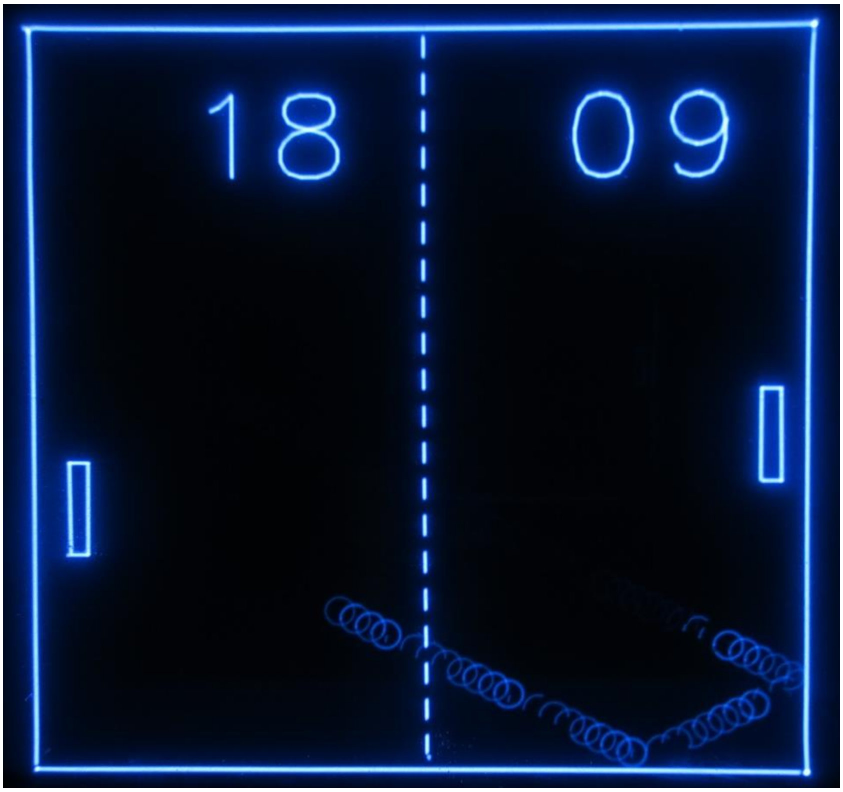
At any time, you may leave CLOCK MODE by entering a character using the serial monitor.

The programmer is encouraged to look through the demo program code as key routines to paint the clock face and clock-hands can be easily leveraged for a more complete CRT-CLOCK implementations.

Note: Clock will continue to display until a key is pushed in the monitor program. <u>Because the SCREEN SAVER function is inactive during continuous time displays, the clock display may cause CRT screen burn-in.</u>

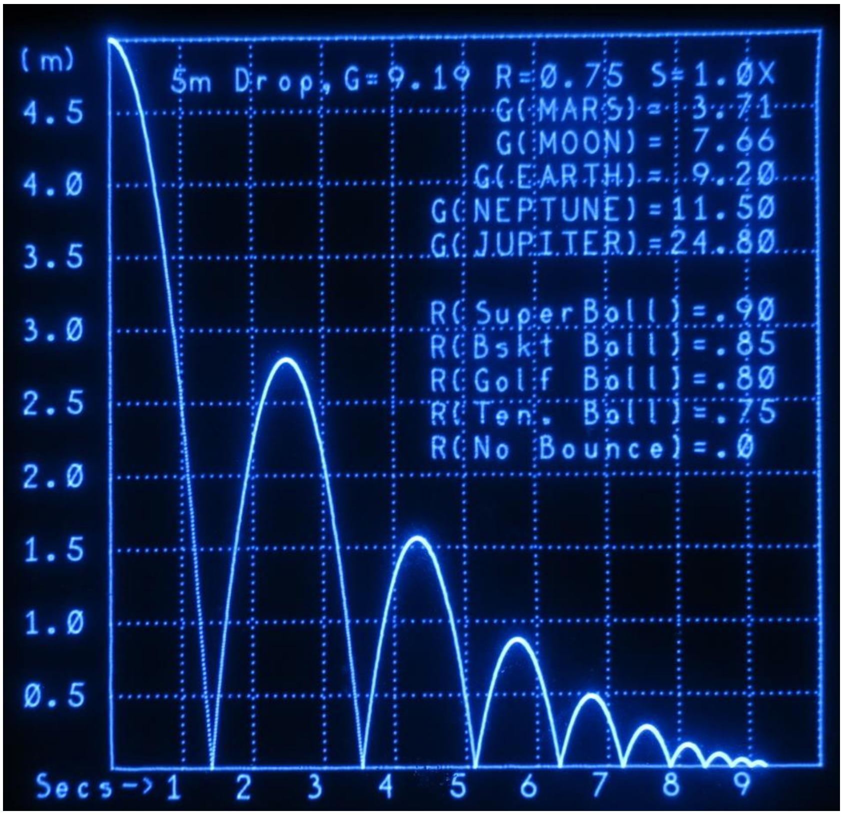
Test Pattern Options (11 of 12)

24 P = Demo: PONG



Show a sample video game ‘in action’; type any key to stop this demo.

25,g,r,s P = Demo: 5m Ball Drop(g=gravity(m/s/s),r=restitution(%),s=speed(%)



```
5m Drop, G=9.19 R=Ø.75 S=1.ØX
                  G(MARS)= 3.71
                  G(MOON)= 7.66
                 G(EARTH)= 9.2Ø
               G(NEPTUNE)=11.5Ø
               G(JUPITER)=24.8Ø

             R(SuperBoll)=.9Ø
             R(Bskt Boll)=.85
             R(Golf Boll)=.8Ø
             R(Ten. Boll)=.75
             R(No Bounce)=.Ø
```

Show a ball drop plot a popular function used to demonstrate early analog computers.

You can enter values for

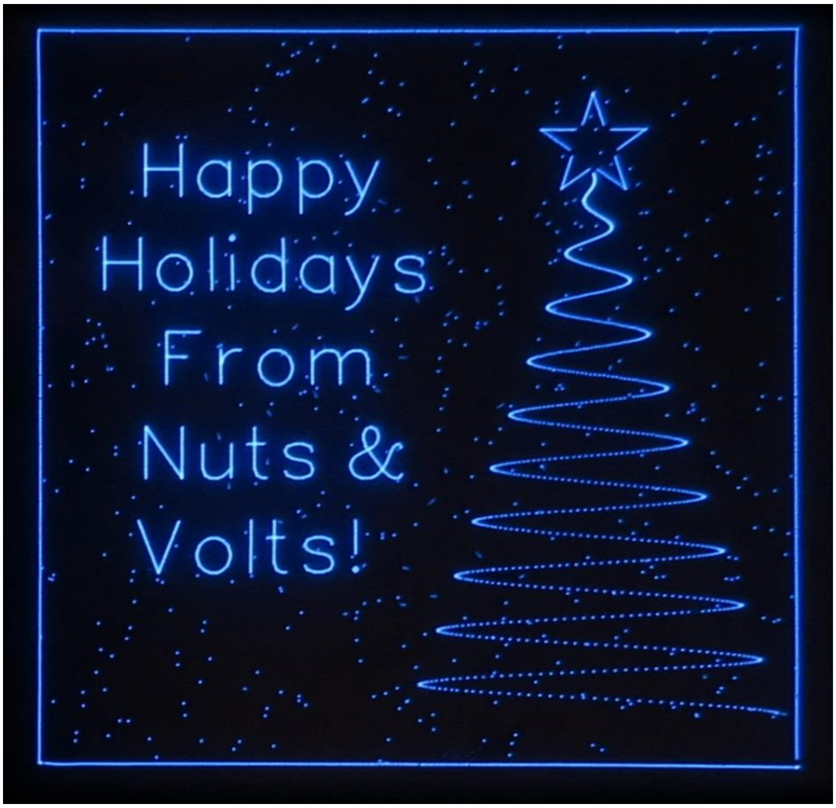
G = Gravity (m/s/s), use values 1.0 to 25.0

R = -Ball Restitution (%) (“bounciness”), use values 0.5 – 0.9

S = Simulation Speed (omit for 1X)
Useful range: .5-10.0 where
1.0 is 1X speed (~real time).
0.5 = half speed
>5 = plot ‘as fast as possible’

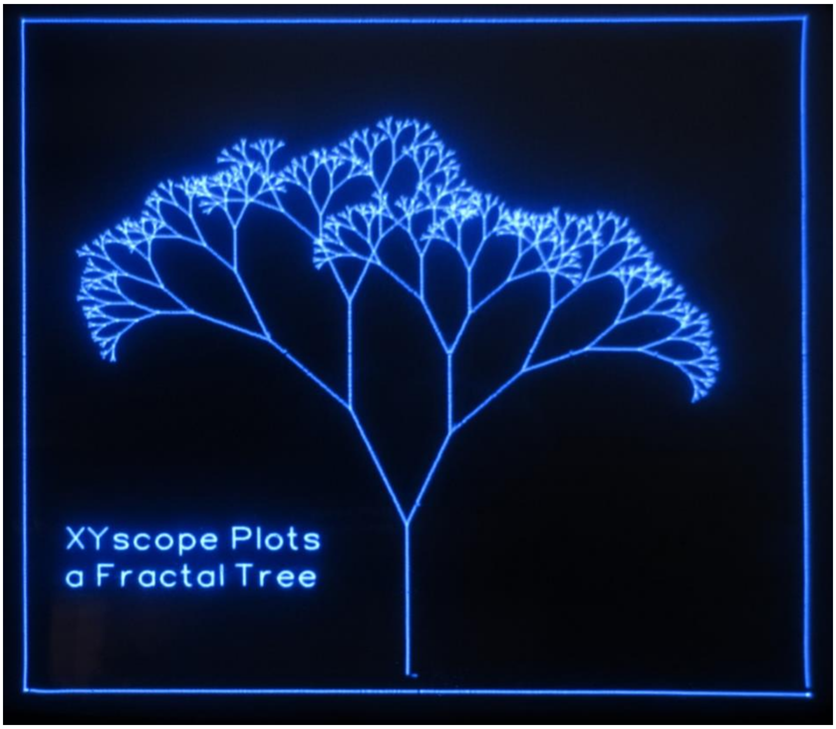
Test Pattern Options (12 of 12)

26 P = Demo:Happy Holidays from Nuts & Volts!



Show a sample plot of a damped sine wave that makes a Christmas Tree.

27,s,a P = Demo:Fractal Tree, s=size(500-2000),a=branch angle (.2-.7)



Generates and displays a unique fractal tree. This piece of demo code shows how a recursive routine that plots a tree-branch can call itself ("recursively") to make a whole tree.

You can enter optional values for

s = Size (length in pixels) of first branch .
Default = 1000
Use values of 500-2000

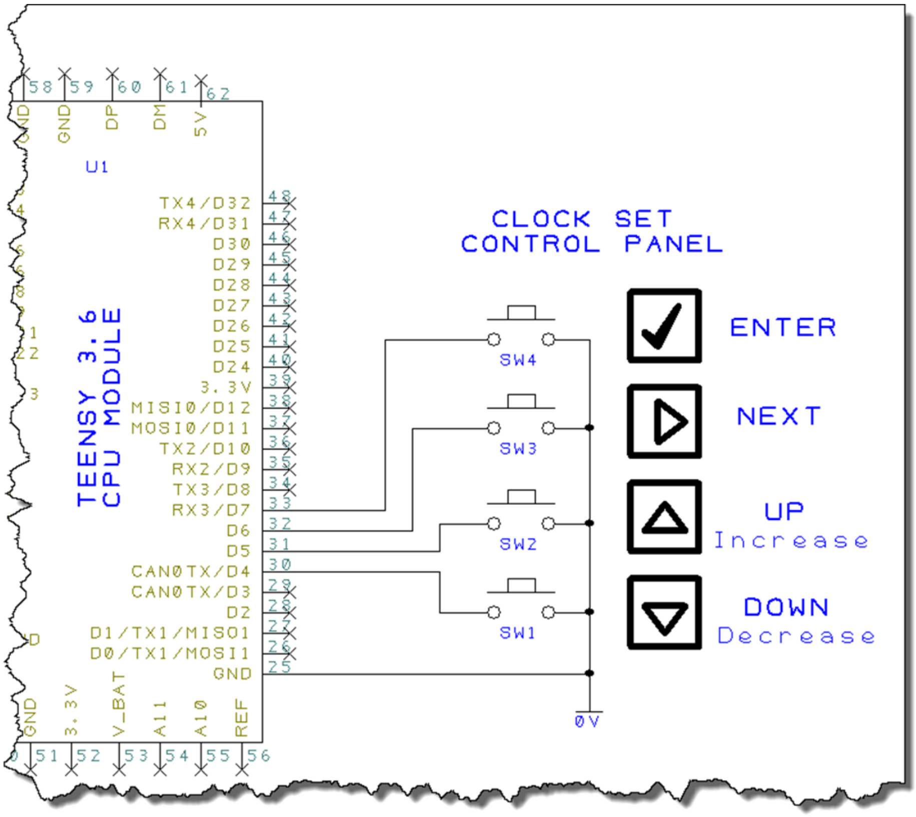
a = Angle (radians) of tilt between connected branches.
Default = .4 (~45deg)
Use values of 0.2-0.7

APPENDIX 1: Control Buttons

(These buttons are active starting with release V 3.10)

Starting with V3.20 code, optional push buttons may be connected to the TEENSY and used to set the clock (While in CLOCK MODE) or vary selected timing parameters (when in TEST & DEMO MODE).

Buttons should be wired as shown to the right.



TEST & DEMO MODE OPERATION

The following values when in Non-Clock-Mode:

Small_Step Settling time
Large_Step_Settling time
UnBlank_time

To activate a button, select the value you wish to adjust by typing (via monitor keyboard):

s↵ Small_Step_Settling time value
S↵ Large_Step_Settling time value
U↵ Unblank time

TEST & DEMO MODE Example:

If you wish to adjust the Unblank time with the control buttons then just type _**U**↵ into the Monitor keyboard.

Now, pressing the UP/DOWN buttons will vary the UNBLANK value. The changed UNBLANK value instantly used and will be displayed on the screen as well.

Press and HOLD a UP/DOWN button to auto repeat (step) the value up or down.

In this way, you can watch the display screen or watch timing waveforms change in 'real time'

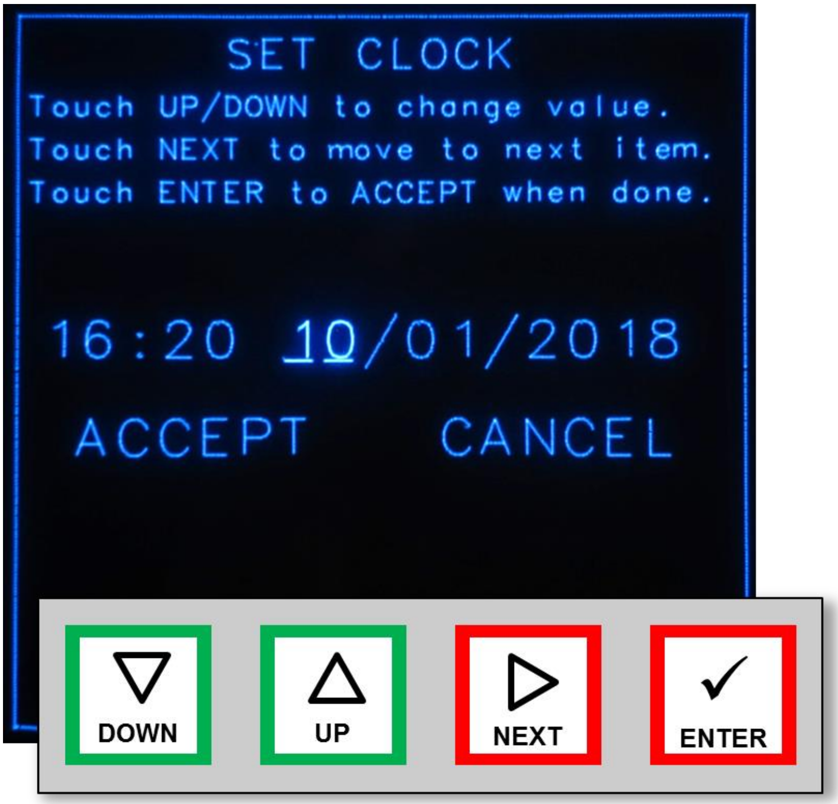
SAVE Timing Values to EEPROM

Use the SAVE to EEPROM option to save the changes to EEPROM where they will be automatically restored upon power up.

CLOCK MODE OPERATION

If you touch any of the 4 push buttons while in CLOCK MODE, the SET CLOCK screen will appear. Follow the instructions on the screen to set time and date. Select ACCEPT then touch ENTER to set the clock to the new values; Select CANCEL and then touch ENTER to leave the SET CLOCK screen *without* changing the current time or date.

Arduino DUE Note: The 4-button control panel may similarly be used with a DUE processor when connected to DUE pins D4-D7.



SET CLOCK

Touch UP/DOWN to change value.
Touch NEXT to move to next item.
Touch ENTER to ACCEPT when done.

16:20 10/01/2018

ACCEPT CANCEL

APPENDIX 2: Optimizing Timing
(1 of 5)

THIS APPLIES FOR TEENSY 3.6 PIO MODE ONLY

What are the Steps to Transfer an XY point pair to the scope screen?

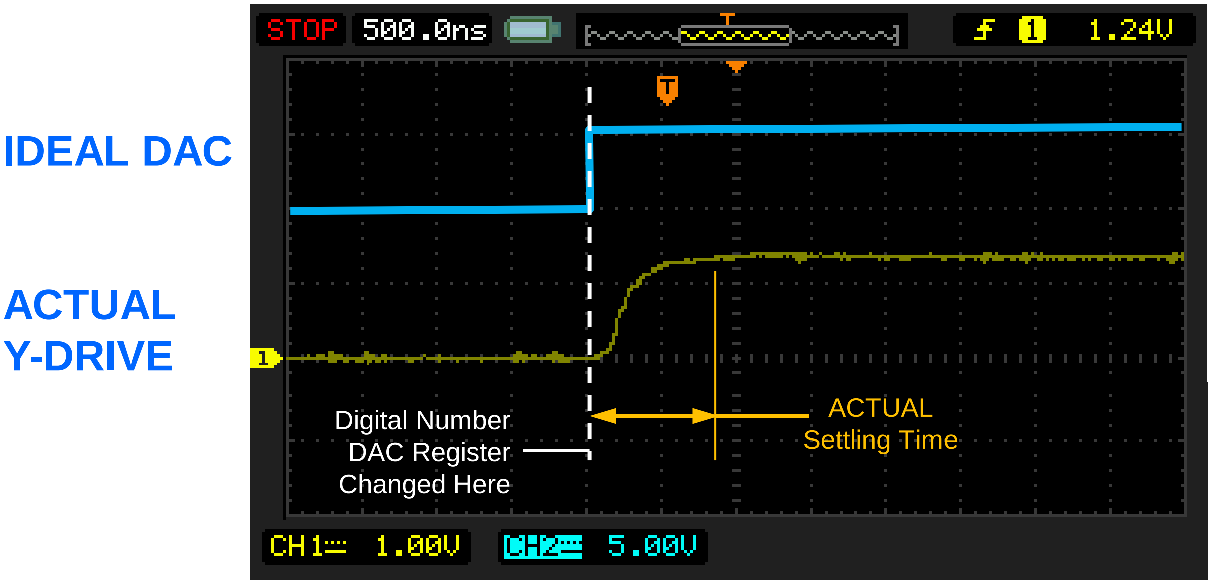
The process to transfer data from the microprocessor memory to the Digital to Analog Converter (DAC) consists of the following steps:

1. Read the X-value out of the XY_List [ ] array.
2. Move the X-value into the X-DAC data register.
3. Read the Y-value out of the XY_List [ ] array.
4. Move the Y-Value into the X-DAC data register.
5. Wait WAIT for a specified time for the DAC output voltages to stabilize and reach their final value (a.k.a. the Settling Time)
6. Set the UNBLANK I/O port = 1 TURN THE BEAM ON
7. Wait KEEP BEAM (a.k.a. “UNBLANK time” or “UNBLANK delay”)
8. Set the UNBLANK I/O port = 0 TURN THE BEAM OFF

There’s a few more house keeping tasks such as updating array pointers and such, but the above list has all of the time critical elements. In order to plot as many points to the screen as possible within our 20ms refresh target, we want to perform these tasks as quickly as possible, However, the actual pace we set must consider the real world delays of the DACs, buffer amplifiers, and the scope response itself. So, several time delay settings and adjustments need to be made to get this process running just right. In this case, “just right” means *FAST*, but not *TOO FAST!*

What is Settling Time?

DAC Settling-Time is the time it takes for a change made at the digital input register of the DAC to be translated to a new and stable analog output voltage. An IDEAL DAC would have zero settling time, that is, the output voltage would change instantaneously with an input DAC register change. Real-world DACs (including those inside the TEENSY 3.6) are ‘not ideal’ and as such require some time for the output voltage to reach it’s intended value after a value change. Also, some additional time is needed for external buffer-amplifiers and the circuits in the oscilloscope to settle as well.



APPENDIX 2: Optimizing Timing (2 of 5)

THIS APPLIES FOR TEENSY 3.6 PIO MODE ONLY

How “wide” is the UNBLANK pulse? How is it adjusted?

The pulse width of the UNBLANK signal is controlled by the “UNBLANK count” program variable. In general, the larger this value is, the brighter the CRT display will be. However, you will want to keep the UNBLANK time as short as possible to keep the Point Period as short as possible.

Point Period?

The point period is the sum total of all of the it takes to plot a single point to the screen. The point period is therefore the fixed program execution time to move the data to the DAC registers, PLUS the settling time PLUS the unblank time. The point period can be measured in nanoseconds (ns) or microseconds (us). We want the Point Period to be as SHORT AS POSSIBLE so that we can plot as many points as possible within the target refresh period of 20ms.

Maximizing Display Quality

As noted, we use a Z-Axis UNBLANK signal to turn the BEAM ON and OFF. For best display quality, we need to keep the scope electron beam turned-off during the “settling time period” and just “flash the point ON” after the spot has fully settled. If we don’t wait long enough before UNBLANKING the scope electron beam, the beam will still be moving and rather than illuminate a single point, we will illuminate a “beam-in-motion line” or line-fragment.

Settling Time Varies by Distance Moved

In practice, the settling time takes longer for points spaced far apart from one another VS points that are close to one another. This means that we can use different settling time delays based on the distance that one point must travel with respect the prior point. In recognition that moving a small distance (smaller DAC voltage changes) needs a shorter settling time that moving a long distance (larger DAC voltage changes), XYscope uses two different settling time values. The **small-step** value is used for closely spaced points, The **large-step** value is used for widely separated points.

What Defines A Small-Step Vs a Large-Step?

The actual distance value (in pixels) that defines what is a **small-step** VS what is a **large-step** I call the **Threshold** value. The **threshold** is measured in X-Y counts (or coordinate values)**.** The default setting for the **threshold is 1000** (pixels)**.** This means that when the beam must make a move in X or Y by less than 1000 pixels, the settling time delay will use the **small-step** value. Similarly, when the beam must make a move in X or Y equal to or greater than 1000 pixels, the settling time delay will use the **large-step** value.

Settling Time Adjustment

Inside of the driver code, delay loops insert a number of “NOP” (no-operation) instructions in-line to implement the **small-step** and **large-step** delays. At a CPU clock of 240 MHz, each NOP instructions represents about 25ns of time delay. As you’d expect, larger delay values generate a longer settling delay times, while smaller values generate shorter settling times. The minimum delay is achieved with a delay value of zero (0), that is, no extra NOP instructions will be inserted. At 0 delay, the other the code in the refresh process takes about 500 ns to execute. When the TEENSY is running at 240 MHz and the delay count values = zero, the point period is indeed about 500 ns. Every non-zero delay count specified increases the total delay by about 25ns. This means that delay settling value of 10 results in a total point time of about 750ns (500ns Min + 10*25ns=750). A setting of 20 results in a total settling time of about 1000ns (500ns Min + 20*25ns=1000).

Making Small-step, Large-step, Threshold & Unblank Adjustments

Running in the TEST MODE,

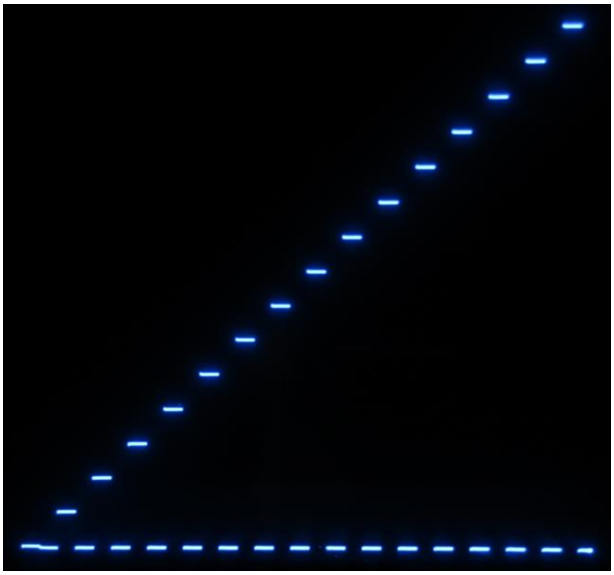
- The **Threshold** value may be viewed and changed using the (**L**↵) command
- The **small-step** value may be viewed and changed using the (**s**↵) command
- The **large-step** value may be viewed and changed using the (**S**↵) command
- The **UNBLANK** value may be viewed and changed using the (**U**↵) command

APPENDIX 2: Optimizing Timing
(3 of 5)

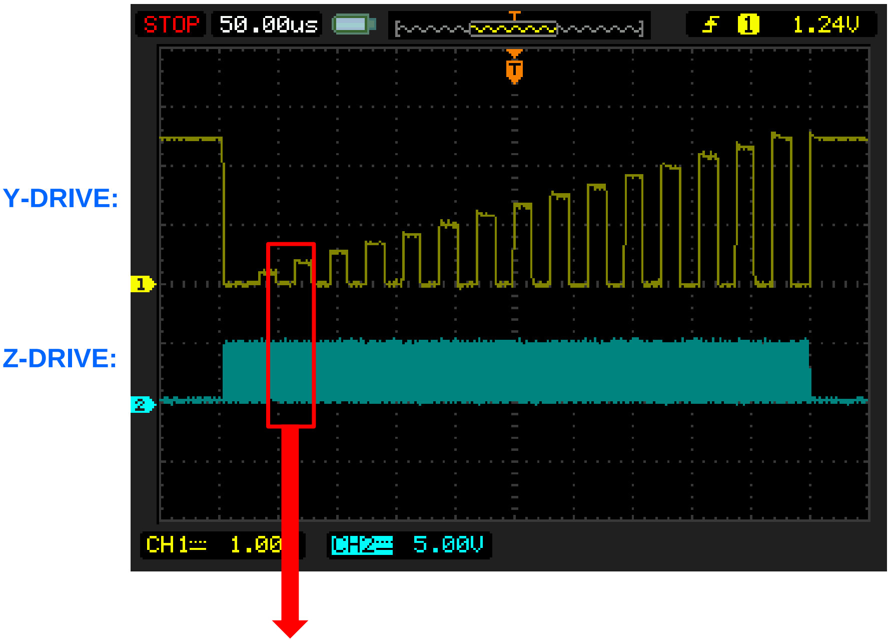
THIS APPLIES FOR TEENSY 3.6 PIO MODE ONLY

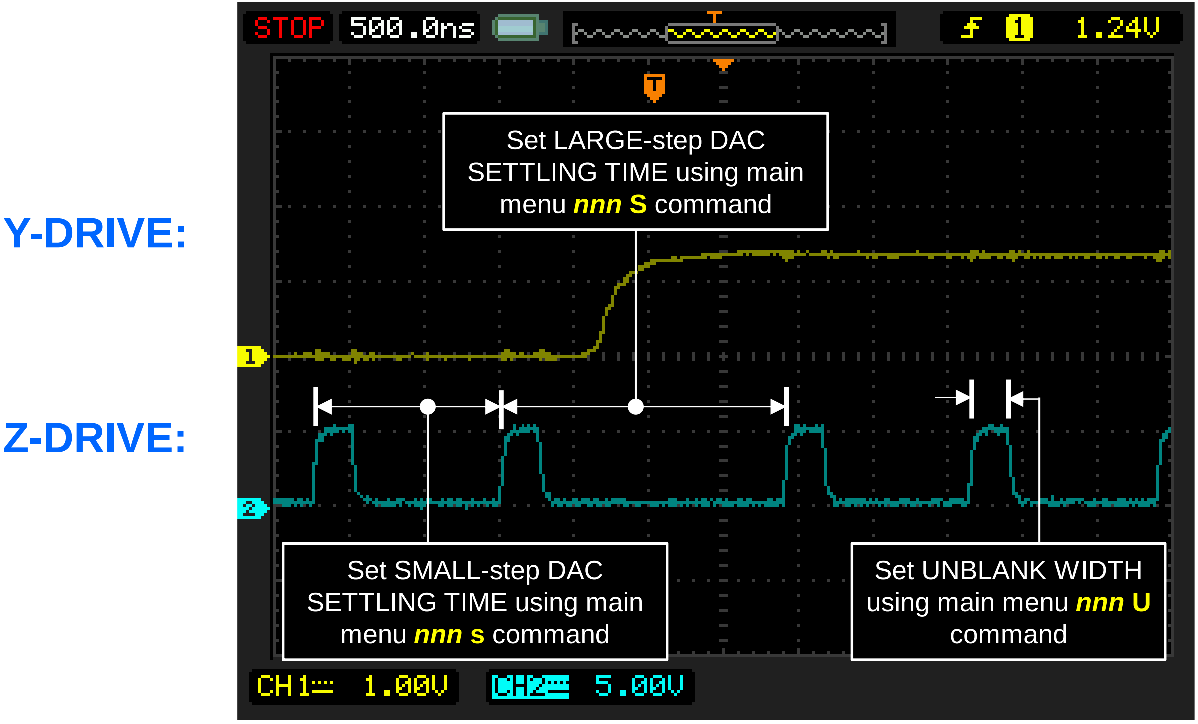
The 3P test pattern is the best pattern to use to determine optimal timing values. Once set using 3P, you can further check & tune the plot quality performance with the other available patterns.

3 P = Vert Stair-Case Test Pattern



Connect Oscilloscope to Y-DRIVE & Z-DRIVE to observe timing relationships and pulse widths





Set LARGE-step DAC SETTLING TIME using main menu ***nnn* S** command

Set SMALL-step DAC SETTLING TIME using main menu ***nnn* s** command

Set UNBLANK WIDTH using main menu ***nnn* U** command

Use MAIN-MENU commands to adjust pulse spacing and widths as shown above. Once you have determined the best values for your system, you should enter those values into the XYscopeConfig.h file so that the optimum values will automatically be used every time the unit is power up. See XYscopeConfig.h file for more details.

APPENDIX 2: Optimizing Timing

(4 of 5)

THIS APPLIES FOR TEENSY 3.6 PIO MODE ONLY

SETTLING TIME TESTS AND ADJUSTMENTS

1. Enter 3P↵ This will display the Stair-case pattern.

2. Enter xx s↵ where ***xx=0***. This is the **small-step** value.
   - You will see “tails” on the smaller stair steps as shown to the right.
   - Keep INCREASING the xx value (xx sP↵) until “tails disappear”.
   - Choose the smallest value where no “small step tails” are seen.

3. Enter xx S↵ where ***xx=0***. This is the **large-step** value.
   - You will see “tails” on the larger stair steps as shown to the right.
   - Keep INCREASING the xx value (xx SP↵) until “tails disappear”.
   - Choose the smallest value where no “large step tails” are seen.

EXAMPLE IMAGES

3 P = Vert Stair-Case Test Pattern

In the photo to the right, the small-step and large-step values are set to 0 (s=0, S=0). The vertical lines appearing between the stair steps show that the settling times are too short. A well tuned system will show no vertical lines at all for the 3P pattern.

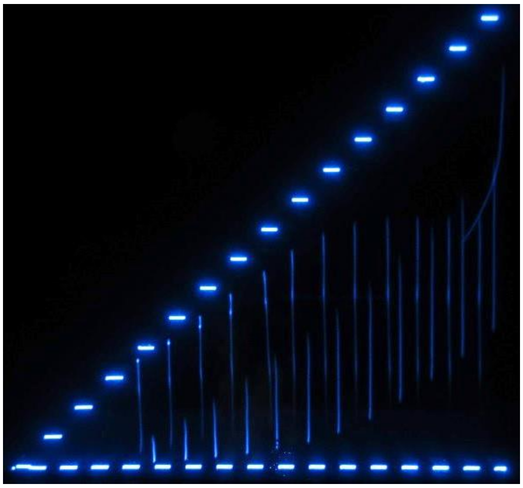
2 P = Just Corner Dots Test Pattern

This pattern should simply display just a couple of ‘dots’, one in the lower left hand corner (0,0) and one in the upper right hand corner (4095,4095). It can be used to check the large-step settling time and blanking pulse settings. If the large-step settling time is too short, the dots will blur or show ‘tails’ such as seen to the right. Increase the large-step settling value until just dots (no tails) are seen.

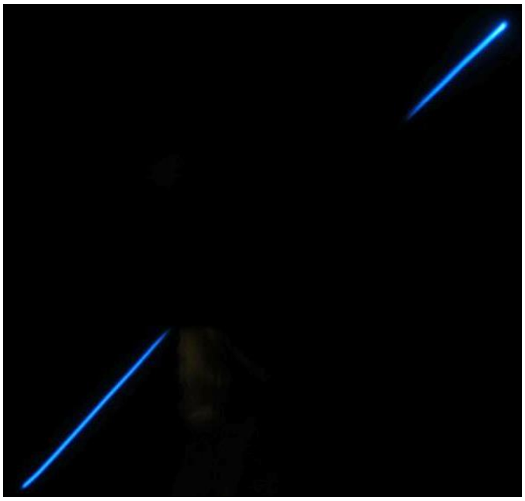
15,50 P = Show 50 Random Points

Us the shown random points command and plot a small # of points (i.e.: 50) to check SETTLING TIME setup. When the SETTLING TIME values are too small, ‘tails’ appear on the points as seen to the right. A well adjusted system will show just random spread of clean, no-tail ‘dots’

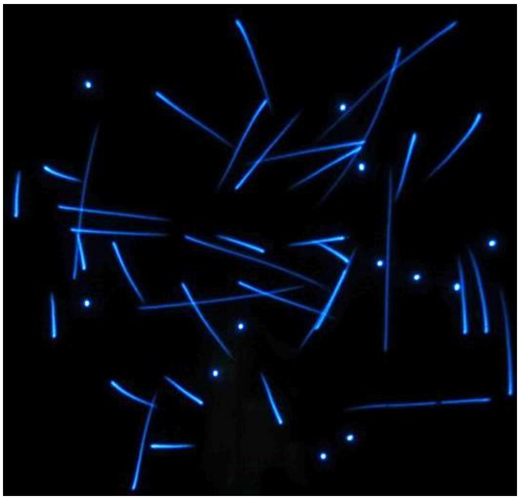
3 P = Vert Stair-Case Test Pattern



2P = Just Corner Dots Pattern



15,50 P = Show 50 Random Points



THRESHOLD ADJUSTMENT

Although usually not necessary, the **threshold** value can be changed.

1. Enter xxxx L ↵ where xxxx is the number you’d like to try.
2. 1000 is normally a good point to be at, but other values may be tried. Use the 3P pattern with zero **large-step** delay and 30 **small step** values. Changing the threshold value will show you how the threshold values affects which settling time values get used.

UNBLANK ADJUSTMENT

1. Enter U↵ and the current unblank pulse setting is output to the serial monitor screen.
2. Enter xx U↵ to enter a new ***xx*** UNBLANK value.
   - Larger values will result in brighter spots, smaller values will result in dimmer spots.
   - Note, use the *intensity* control on the oscilloscope as required to increase overall display brightness as well.
   - Select and use the SMALLEST value that still yields acceptable brightness.

APPENDIX 2: Optimizing Timing
(5 of 5)

THIS APPLIES FOR TEENSY 3.6 PIO MODE ONLY

Making Setup Value Changes PERMANENT; Editing the Start-Up values Inside of XYscopeConfig.h File

The start-up Small-Settling value, Large-Setting value, UNBLANK count, and Threshold values may be permanently changed to meet your needs. For the PROCESSOR SPEED you are running at simply change the values as shown in this portion of the XYscopeConfig.h file. For example, if you are running your TEENSY at 240 Mhz, then simply change the values as shown in the file fragment in **RED highlight** below where:

Small-Settling Time = CFG_PioSmallSettleCount	//Defines Small Step Settling time delay
Large-Settling Time = CFG_PioLargeSettleCount	//Defines Large Step Settling time delay
UNBLANK Time = CFG_PioUnblankCount	//Defines UNBLANK Pulse Width
Threshold value = CFG_NoSettlingTimeReqd	//Defines Small Step Breakpoint

```
     :
//=========== PIO DAC Settling time and Unblank Pulse Width ============
//   F_CPU = TEENSY 3.x ONLY. These values fine tune UNBLANK timing and cope with the relatively slow DAC
//   performance of the TEENSY 3.x processors.  These values can also help cope with performance limitations
//   that may exist in 'slow O-scopes'. Delay values are specified in 'counts'.  The actual delays achieved
//   vary depending on the CPU operating Frequency.  For this reason, the user may make individual adjustments
//   based on CPU operating frequency.  Note: Larger values will increase settling time/increase unblank time.
//   Smaller values decreases settling time/unblank time.
//
//   Note: The values below have been set with Vref = 1.5V using a high speed HP1332A XYZ Monitor;
//         These may not be the optimum values when you have Vref = 3.3V or when using with a lower speed scope!
//
#if defined(__SAM3X8E__)  //Defined for use with ARDUINO DUE processor
                          //(PIO mode only, Not used for DMA!), 84 Mhz CPU Speed
     #define CFG_PioSmallSettleCount 0       //Defines Small Step Settling time delay (n=Num of NOP instructions)
     #define CFG_PioLargeSettleCount 0       //Defines Large Step Settling time delay (n=Num of NOP instructions)
     #define CFG_PioUnblankCount 0           //Defines UNBLANK Pulse Width (n=Num of NOP instructions)
     #define CFG_NoSettlingTimeReqd 200      //Defines Small Step Breakpoint (in DAC counts)
#endif

#if (F_CPU > 216000000)       //TEENSY - More than 216 Mhz CPU Speed (ie: 240 Mhz)
     #define CFG_PioSmallSettleCount 25      //Defines Small Step Settling time delay (n=Num of NOP instructions)
     #define CFG_PioLargeSettleCount 50      //Defines Large Step Settling time delay (n=Num of NOP instructions)
     #define CFG_PioUnblankCount 5           //Defines UNBLANK Pulse Width (n=Num of NOP instructions)
     #define CFG_NoSettlingTimeReqd 1000     //Defines Small Step Breakpoint (in DAC counts)
#endif

#if (F_CPU == 216000000)  //TEENSY - 216 Mhz CPU Speed
     #define CFG_PioSmallSettleCount 18
     #define CFG_PioLargeSettleCount 40
     #define CFG_PioUnblankCount 5
     #define CFG_NoSettlingTimeReqd 1000
#endif

#if (F_CPU == 192000000)  //TEENSY - 192 Mhz CPU Speed
     #define CFG_PioSmallSettleCount 14
     #define CFG_PioLargeSettleCount 35
     #define CFG_PioUnblankCount 5
     #define CFG_NoSettlingTimeReqd 1000
#endif
     :
```

Making Setup Value Changes PERMANENT; using TEENSY EEPROM

The CRT_XYSCOPE_CLOCK.ino program has an option to use the EEPROM of the TEENSY to store the startup settling time values. The EEPROM startup option is enabled by the following CRT_SCOPE_CLOCK.ino code.

```
   const bool PowerUpInitializeFromEEPROM = true;   //true=Startup using EEPROM stored timing values Data-Set
                                                    //false=Startup using XYscopeConfig.h timing values Data-Set
```

Once the desired settings are determined and actually in use, use the EEPROM menu option 10 to save the setup to EEPROM. When enabled, the EEPROM will be read each time the system is powered up and the saved settings will OVER WRITE retrieved XYscopeConfig.h values. See **MAIN MENU Details (Pg 1 of 2)** for more details on EEPROM INSPECT, READ, & SAVE commands.

APPENDIX 3: XYscopeConfig.h

(Table is subject to update and change without notice)

This guideline provides an overview of the configuration parameters available. Please see the current XYscopeConfig.h file for the most up-to-date information.

Key parameters that define how the AGI driver code will operate can be set and controlled by specific entries into parameters found within the XYscopeConfig.h file. These include:

Parameter	Type	DUE	TNSY	Values	Description/Notes
PROCESSOR TYPE	Compiler Switch	X	X	__SAM3X8E__ __MK66FX1M)__	This variable is automatically set within the Arduino IDE when a given board type is selected in the TOOLS menu. This is used throughout the AGI driver to utilize the appropaite code set to match the processor in use.
CFG_MaxArraySize	Integer Constant	X	X	DUE: 15K TEENSY:35K Flicker usually detected beyond 15K points	This parameter sets the overall size of the Xylist point array. The maximum size is constrained by the maximum available RAM AND the number of points that can be plotted to the screen before flicker is objectionable. Two values may be defined, one for DUE, one forTEENSY 3.6.
CFG_CrtMinRefresh_us	Integer Constant	x (PIO)	X	Typ: 2000	Defines the minimum refresh period, in microseconds). Note: The value is normally set to 2000 us which defines a 50 Hz screen refresh rate.
CFG_IncludeHersheyFontROM	Compiler Switch	X	X	true or false (Normally set true)	Enables (true) or disables the font. Note: Atleast ONE font must be enabled.
CFG_IncludeVectorFontROM	Compiler Switch	X	X	true or false (Normally set true)	Enables (true) or disables the font. Note: Atleast ONE font must be enabled.
CFG_StartupFont	Integer Constant	X	X	0 or 1 (Normally set to 1)	When both Fonts are enabled, this variable defines which font is set as the startup default. 0=VectorFontROM, 1 = HersheyFontROM. Value ignored when only one font exists.
CFG_IgnoreUndefinedCharacters	boolean Constant	X	X	true or false (Normally set true)	When set false, characters undefined will plot as "~". When set true, undefined characters are ignored and do not plot.
CFG_DUE_Use_DMA	Compiler Switch	X		COMMENT-OUT this run DUE in DMA mode	When COMMENTED OUT, the DUE will run in a PIO mode (like the TEENSY); when NOT COMMENTED OUT, the DUE will run in preferred DMA mode. Note: DUE performs poorly in PIO mode; DMA is recommended. Switch is only active using the Arduino DUE & is ignored in TEENSY mode.
CFG_Due_DAC0_Pin	Integer Constant	X		DAC0	Defines DUE DAC0 output pin. DAC0 is the only valid value.
CFG_Due_DAC1_Pin	Integer Constant	X		DAC1	Defines DUE DAC0 output pin. DAC1 is the only valid value.
CFG_PioPositiveBlankingLogic	Compiler Switch	x (PIO)	X	COMMENT-OUT this line for NegativeLogic	Recommended: DO NOT COMMENT OUT. Simply use PCB Jumper setting if needed to invert logic going to scope.
CFG_TNSY_DacRefVolts_LOW	Compiler Switch		X	COMMENT-OUT this line for 3.3V DAC REF	COMMENT OUT this line to set a 3.3V DAC Reference Voltage. DO NOT COMMENT-OUT to use a 1.5V DAC Reference Voltage. This item only applies to TEENSY 3.6 Processor. 1.5V DAC REF is the PREFFERED setting.
CGF_TNSY_3_6_ MinimumComputeTimeUs	Integer Constant	x (PIO)	X	Normally set to 2000	This is extra time tacked onto the refresh period when needed to ensure a minimum available compute time when plotting large numbers of points. Larger values give more compute time but can increase display flicker at high point counts.
CFG_PioSmallSettleCount	Integer Constant	x (PIO)	X	0-75 (Varies by CPU clock)	Defines Small Step Settling time delay (where n=Num of NOP instructions). (Note 1)
CFG_PioLargeSettleCount	Integer Constant	x (PIO)	X	0-75 (Varies by CPU clock)	Defines Large Step Settling time delay (where n=Num of NOP instructions). (Note 1)
CFG_PioUnblankCount	Integer Constant	x (PIO)	X	0-25 (Varies by CPU clock)	This value sets the PIO Unblank Pulse Width. (Note 1)
CFG_NoSettlingTimeReqd	Integer Constant	x (PIO)	X	0-4095 (may variy by CPU clock)	This value sets the breakpoint where the driver switches from PioSmallSettleCount to PioLargeSettleCount. (Note 1)
CFG_InitClockFromTNSY_RTC	Bool Constant		X	true or false (Normally set true)	True to initialize TIME software from RTC, false if RTC battery not present or you do NOT want to use TNSY_RTC values.

NOTE 1: Multiple, CPU speed dependent values are present in the file. Use correct value for your CPU speed. Starting with code set V3.10, upon powerup EEPROM stored values (If enabled!) may overide this config.h value.