In [1]:
```
import sys
sys.path.append(r"C:\Users\disha\AppData\Local\Programs\Python\Python312\Lib\site-packages")
import chipwhisperer as cw
scope = cw.scope()
scope.default_setup()
target = cw.target(scope)
```

```
(ChipWhisperer NAEUSB WARNING|File naeusb.py:713) Your firmware (0.62) is outdated - latest is 0.65 See https://chi
pwhisperer.readthedocs.io/en/latest/firmware.html (https://chipwhisperer.readthedocs.io/en/latest/firmware.html) fo
r more information
```

In [2]:
```
%%bash
cd ../../hardware/victims/firmware/simpleserial-base/
make PLATFORM=CWNANO CRYPTO_TARGET=NONE
```

```
SS_VER set to SS_VER_1_1
SS_VER set to SS_VER_1_1
SS_VER set to SS_VER_1_1
SS_VER set to SS_VER_1_1
make[1]: '.dep' is up to date.
SS_VER set to SS_VER_1_1
SS_VER set to SS_VER_1_1
.
Welcome to another exciting ChipWhisperer target build!!
arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103 (release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

.
Compiling:

simpleserial-base.c: In function 'main':
simpleserial-base.c:118:30: warning: passing argument 3 of 'simpleserial_addcmd' from incompatible pointer type [-W
incompatible-pointer-types]
  118 |  simpleserial_addcmd('k', 0, state_permute);
      |                              ^~~~~~~~~~~~~
      |                              |
      |                              uint8_t (*)(uint64_t *) {aka unsigned char (*)(long long unsigned int *)}
In file included from simpleserial-base.c:5:
.././simpleserial/simpleserial.h:49:61: note: expected 'uint8_t (*)(uint8_t *, uint8_t)' {aka 'unsigned char (*)(un
signed char *, unsigned char)'} but argument is of type 'uint8_t (*)(uint64_t *)' {aka 'unsigned char (*)(long long
unsigned int *)'}
   49 | int simpleserial_addcmd(char c, unsigned int len, uint8_t (*fp)(uint8_t*, uint8_t));
      |                                                   ~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~
```

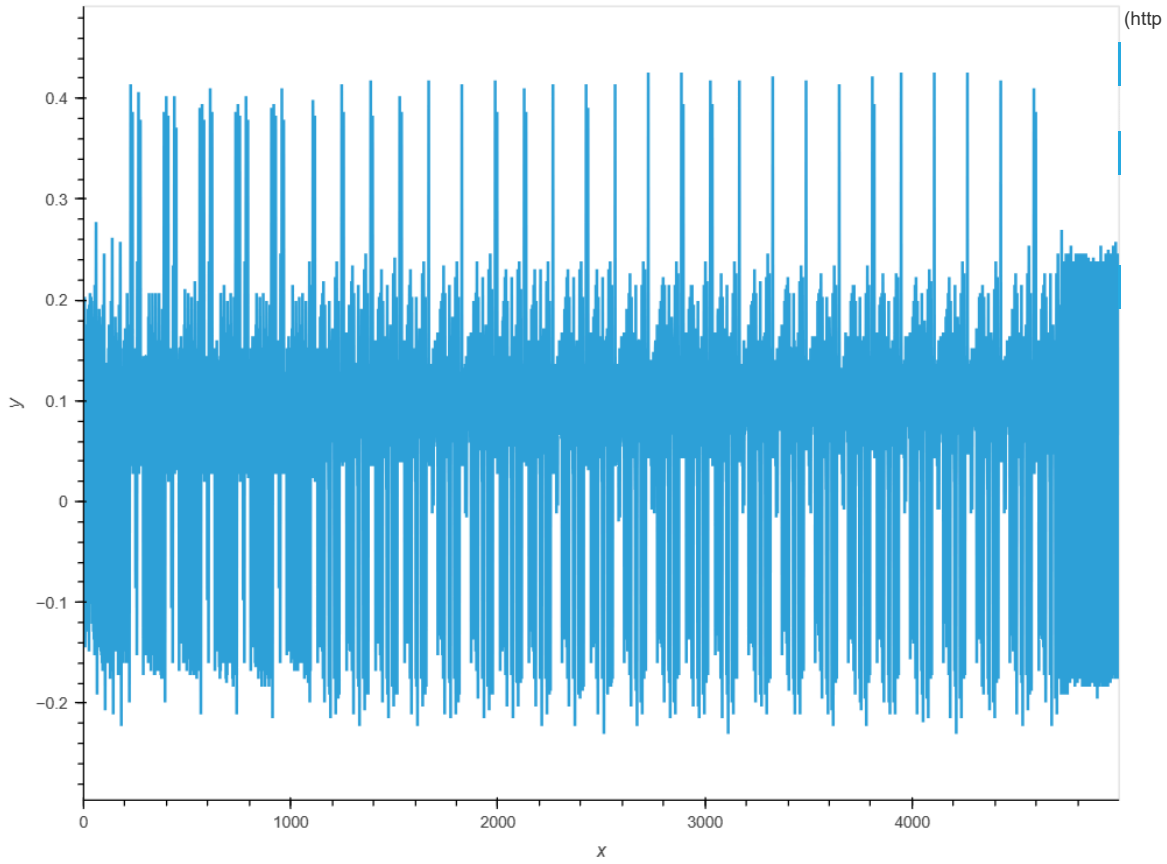
```
    simpleserial-base.c ...Done!
.
Compiling:
    .././simpleserial/simpleserial.c ...Done!
.
Compiling:
    .././hal/stm32f0_nano/stm32f0_hal_nano.c ...Done!
.
Compiling:
    .././hal/stm32f0/stm32f0_hal_lowlevel.c ...Done!
.
Assembling: .././hal/stm32f0/stm32f0_startup.S
arm-none-eabi-gcc -c -mcpu=cortex-m0 -I. -x assembler-with-cpp -mthumb -mfloat-abi=soft -ffunction-sections -DF_CPU
=7372800 -Wa,-gstabs,-adhlns=objdir-CWNANO/stm32f0_startup.lst -I.././simpleserial/ -I.././simpleserial/ -I.././hal
-I.././hal/stm32f0 -I.././hal/stm32f0/CMSIS -I.././hal/stm32f0/CMSIS/core -I.././hal/stm32f0/CMSIS/device -I.././ha
l/stm32f0/Legacy -I.././crypto/ .././hal/stm32f0/stm32f0_startup.S -o objdir-CWNANO/stm32f0_startup.o
.
LINKING:
    simpleserial-base-CWNANO.elf ...Done!
.
Creating load file for Flash: simpleserial-base-CWNANO.hex
arm-none-eabi-objcopy -O ihex -R .eeprom -R .fuse -R .lock -R .signature simpleserial-base-CWNANO.elf simpleserial-
base-CWNANO.hex
.
Creating load file for Flash: simpleserial-base-CWNANO.bin
arm-none-eabi-objcopy -O binary -R .eeprom -R .fuse -R .lock -R .signature simpleserial-base-CWNANO.elf simpleseria
l-base-CWNANO.bin
.
Creating load file for EEPROM: simpleserial-base-CWNANO.eep
arm-none-eabi-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 --no-change-warnings -O ihex simpleserial-base-CWNANO.elf simpleserial-base-CWNANO.e
ep || exit 0
.
Creating Extended Listing: simpleserial-base-CWNANO.lss
arm-none-eabi-objdump -h -S -z simpleserial-base-CWNANO.elf > simpleserial-base-CWNANO.lss
.
Creating Symbol Table: simpleserial-base-CWNANO.sym
arm-none-eabi-nm -n simpleserial-base-CWNANO.elf > simpleserial-base-CWNANO.sym
SS_VER set to SS_VER_1_1
SS_VER set to SS_VER_1_1
Size after:
   text    data     bss     dec     hex filename
   5768      12    1428    7208    1c28 simpleserial-base-CWNANO.elf
+--------------------------------------------------------
+ Default target does full rebuild each time.
+ Specify buildtarget == allquick == to avoid full rebuild
+--------------------------------------------------------
+--------------------------------------------------------
+ Built for platform CWNANO Built-in Target (STM32F030) with:
+ CRYPTO_TARGET = NONE
+ CRYPTO_OPTIONS = AES128C
+--------------------------------------------------------
```

In [3]:
```
cw.program_target(scope, cw.programmers.STM32FProgrammer, "../../hardware/victims/firmware/simpleserial-base/simplese
```

```
Detected known STMF32: STM32F04xxx
Extended erase (0x44), this can take ten seconds or more
Attempting to program 5783 bytes at 0x8000000
STM32F Programming flash...
STM32F Reading flash...
Verified flash OK, 5783 bytes
```

In [4]:
```
splot = cw.StreamPlot()
splot.plot()
```

Out[4]:



In [5]:
```
from tqdm.notebook import trange
import numpy as np

def generate_key_and_capture_trace():
    scope.arm()
    target.simpleserial_write('k', bytearray())
    ret = scope.capture()
    if ret:
        print("Capture timeout.")
        return None

    trace = scope.get_last_trace()
    splot.update(scope.get_last_trace())
    return trace
traces = []
energy_keygen = []
delta_t = 1.34e-7
N=10
for i in trange(N):
    trace = generate_key_and_capture_trace()
    if trace is not None:
        traces.append(trace)
        energy = np.sum(trace)*delta_t
        energy_keygen.append(energy)

traces_array = np.array(traces)
Total_energy_1 = np.sum(energy_keygen)
Total_energy_keygen = Total_energy_1/3600
print(f"Total energy consumption for key generation function in Watst-hour is {Total_energy_keygen}")
print("Standard deviation between the engeries for key generation function is: ", np.std(energy_keygen))
```

```
100%                                                  10/10 [00:00<00:00, 14.15it/s]

Total energy consumption for key generation function in Watst-hour is 1.404007682291667e-07
Standard deviation between the engeries for key generation function is:  1.4970218939830016e-06
```

In [ ]:

```
def collect_traces_keygen(num_traces):
    delta_t = 1.34e-7
    traces = []
    energy_keygen = []
    for i in range(num_traces):
        scope.arm()

        priv, pub = kem_keygen512()
        seed = get_random_bytes(KYBER_SYM_BYTES)
        seed = bytearray([x & 0xFF for x in seed])

        # Check for timeout
        ret = scope.capture()
        if ret:
            print("Capture timed out")
            continue
         # Retrieve and store the trace
        trace = scope.get_last_trace()
        if trace is None:
            print(f"Trace capture failed for iteration {i}. Skipping this trace.")
            continue  # Skip this iteration if trace is None
        traces.append(trace)
        energy = np.sum(trace)*delta_t

        splot1.update(scope.get_last_trace())
        energy_keygen.append(energy)
    return priv, pub, seed, np.array(energy_keygen)

#Execute
priv, pub, seed, energy_keygen = collect_traces_keygen(100)

#Energy Calaculation
Total_energy_1 = np.sum(energy_keygen)
Total_energy_keygen = Total_energy_1/3600
print(f"Total energy consumption for key generation function in Watt-hour is {Total_energy_keygen}")
print("Standard deviation between the engeries for key generation function is: ", np.std(energy_keygen))
```

In [ ]: