## Logger Results

In [9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [10]:
```
cycles_per_func = np.array([ 1, 10, 25, 50, 75, 100, 125 ])
no_timer = np.array([ 0.077, 0.71, 1.65, 3.37, 5.10, 6.83, 8.87 ])
with_timer = np.array([ 0.250, 0.89, 1.87, 3.59, 5.37, 7.21, 9.32 ])
```

Test codebase samples were generated with special code generator (see *{project_root}/data/codegen.py*). The codebase contains functions with cycles within (generate random number on each iteration). There is one main function called *trampoline()* which runs all this functions consequently. To run the script one should provide three arguments: number of functions to generate (1), number of cycles per function (2), number of calls for each function within *trampoline()* (3).

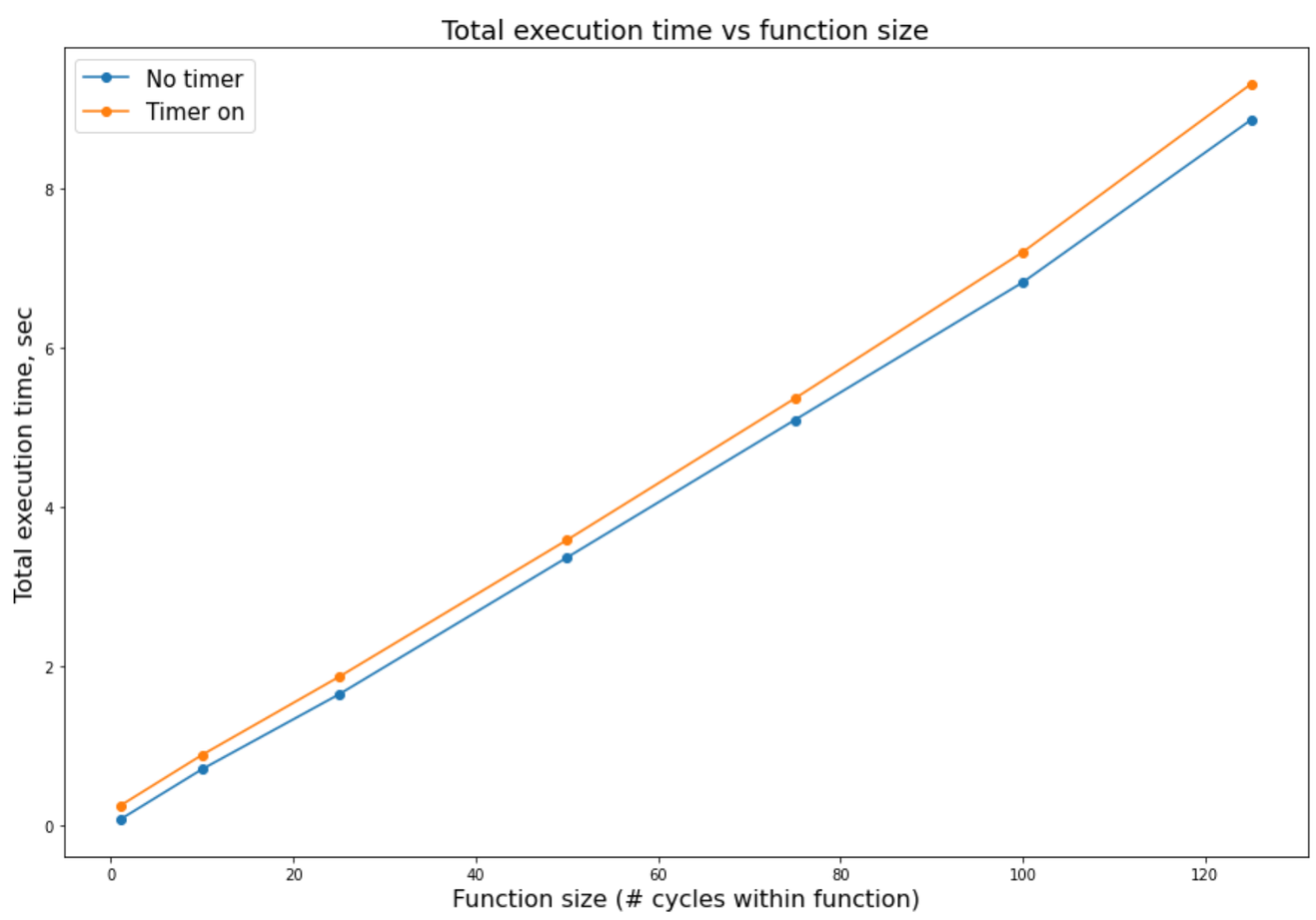
The main function in *main.cpp* runs trampoline in each thread of 10 threads. The codebase used was created with funcs_cnt = 100, calls_cnt = 100000. The execution time of the program is investigated depending on the cycles_per_func parameter.

In [17]:
```
plt.figure(figsize=(15, 10))

plt.title('Total execution time vs function size', fontsize=18)
plt.xlabel('Function size (# cycles within function)', fontsize=16)
plt.ylabel('Total execution time, sec', fontsize=16)

plt.plot(cycles_per_func, no_timer, '-o', label='No timer')
plt.plot(cycles_per_func, with_timer, '-o', label='Timer on')
plt.legend(prop={ 'size' : 15 })
```

Out[17]: <matplotlib.legend.Legend at 0x7ff0cfc08510>

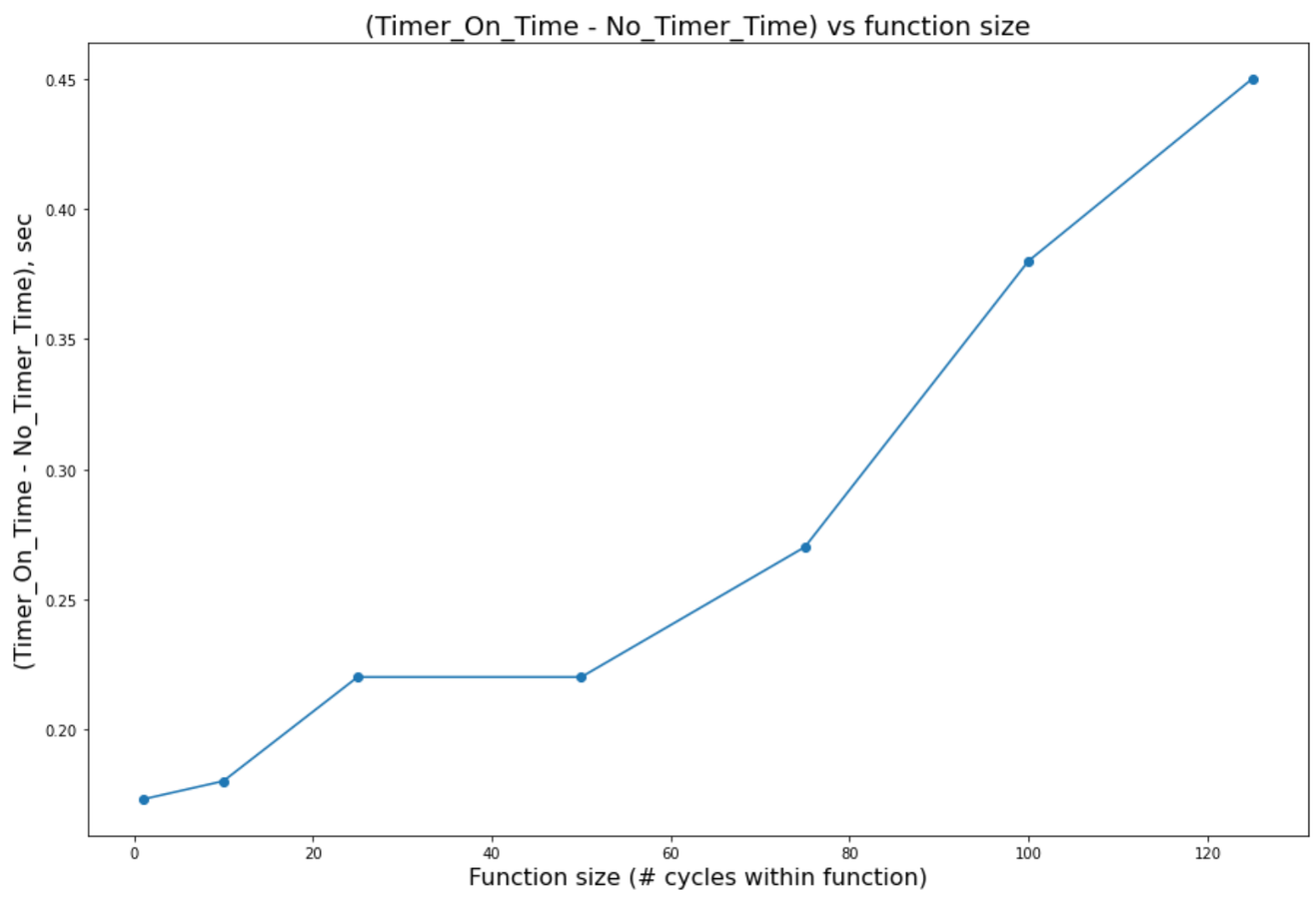



In [18]:
```
plt.figure(figsize=(15, 10))

plt.title('(Timer_On_Time - No_Timer_Time) vs function size', fontsize=18)
plt.xlabel('Function size (# cycles within function)', fontsize=16)
plt.ylabel('(Timer_On_Time - No_Timer_Time), sec', fontsize=16)

plt.plot(cycles_per_func, with_timer - no_timer, '-o')
```

Out[18]: [<matplotlib.lines.Line2D at 0x7ff0cf82e910>]




In [16]: