In [1]:
```
%matplotlib notebook
import matplotlib.pyplot as plt

def plot_to_notebook(time_sec,in_signal,n_samples,out_signal=None):
    plt.figure()
    plt.subplot(1,1,1)
    plt.xlabel("TIME")
    plt.grid()
    plt.plot(time_sec[:n_samples]*1e6,in_signal [:n_samples],"y-",label="Input signal")
    if out_signal is not None:
        plt.plot(time_sec[:n_samples]*1e6,out_signal[:n_samples], 'g-', linewidth=2, la
    plt.legend()
```

In [2]:
```
import numpy as np
#Total time
T=10.26*10**-6
#Sampling Frequency
fs=100e6
#Number of samples
n=int(T*fs)

'''generate the signal '''
t = np.linspace(0, T, n, endpoint=False); # time increment.  generate 2 more samples to
# stagger phases of the sine waves to reduce peak to average
samples = np.loadtxt("TenthInput.txt", dtype=np.int16)
samples=samples.astype(np.int16)
print("Number of samples: ",len(samples))

'''
#Time vector in seconds
t=np.linspace(0,T,n,endpoint=False)
#Samples of the signal
samples= 10000*np.sin(0.2e6*2*np.pi*t)+1500*np.cos(46e6*2*np.pi*t)+2000*np.sin(12e6*2*n
#Convert samples to 32-bit integers
samples=samples.astype(np.int16)
print("Number of samples: ",len(samples))
'''

#Plot signal to the notebook
#plot_to_notebook(t,samples,1026)
```

```
Number of samples:  1026
```

Out[2]:
```
' \n#Time vector in seconds\nt=np.linspace(0,T,n,endpoint=False)\n#Samples of the signal
\nsamples= 10000*np.sin(0.2e6*2*np.pi*t)+1500*np.cos(46e6*2*np.pi*t)+2000*np.sin(12e6*2*
np.pi*t)\n#Convert samples to 32-bit integers\nsamples=samples.astype(np.int16)\nprint
("Number of samples: ",len(samples))\n'
```

In [3]:
```
from scipy.signal import lfilter

#coeffs = [-255,-260,-312,-288,-144,153,616,1233,1963,2739,3474,4081,4481,4620,4481,408
''' load coeffs'''
with open('coeffsDecimal.txt') as f:
    coeffs = f.readlines()
coeffs = np.array([int(coe) for coe in coeffs], dtype=np.int16)
#coeffs
```

In [4]:
```
import time

start_time = time.time()
sw_fir_output = lfilter(coeffs, 70e3, samples)
stop_time = time.time()
sw_exec_time = stop_time - start_time
print("SOFTWARE FIR EXECUTION TIME: ", sw_exec_time)

#plot_to_notebook(t, samples, 400, out_signal=sw_fir_output)
```

```
SOFTWARE FIR EXECUTION TIME:  0.004336118698120117
```

In [5]:
```
from pynq import Overlay
import pynq.lib.dma

#Load the overlay
overlay=Overlay("fir.bit")
#overlay?
```

In [6]:
```
#Load the FIR DMA
dma=overlay.fir_dma
#filter: hierarchy
#fir_dma:name given to dma
#dma.register_map
#dma.running?
```

In [7]:
```
from pynq import allocate
in_buffer=allocate(shape=(n,),dtype=np.int16)
out_buffer=allocate(shape=(n,),dtype=np.int16)
print(n)
```

```
1026
```

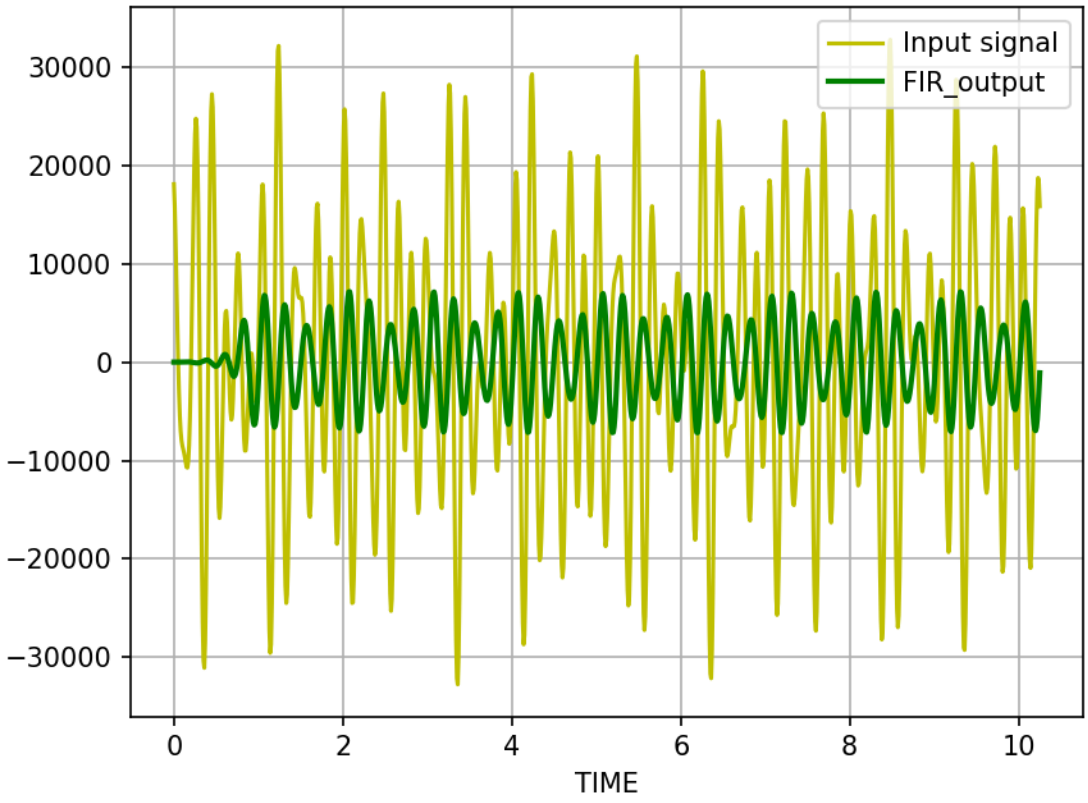
In [8]:
```
#Copy the samples to the in_buffer
np.copyto(in_buffer,samples)
#in_buffer[:] = samples
```

In [9]:
```
#Trigger the DMA transfer and wait for the result
import time
start_time=time.time()
dma.sendchannel.transfer(in_buffer) #errors here
dma.recvchannel.transfer(out_buffer)
dma.sendchannel.wait()
dma.recvchannel.wait()
stop_time=time.time()
hw_exec_time=stop_time-start_time
```

In [10]:
```
print("Hardware FIR execution time: ",hw_exec_time)
print("Hardware acceleration factor: ", sw_exec_time/hw_exec_time)

# Plot to the notebook
plot_to_notebook(t,samples,1026,out_signal=out_buffer)
```

```
Hardware FIR execution time:  0.003695964813232422
Hardware acceleration factor:  1.1732034576183719
```



In [11]:
```
#Free the buffers
in_buffer.close()
out_buffer.close()
```

In [ ]:
```

```