In [ ]:
```
from pynq import Overlay
ol=Overlay("pwm_pulser2.bit")
```

In [2]:
```
from pynq import MMIO
RANGE = 8 # Number of bytes; 8/4 = 2x 32-bit locations which is all we need for
```

In [3]:
```
duty_address = ol.ip_dict['axi_gpio_duty']['phys_addr']
duty_register = MMIO(duty_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as outp
duty_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set tri-state to outp
```

In [4]:
```
band_address = ol.ip_dict['axi_gpio_band']['phys_addr']
band_register = MMIO(band_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as outp
band_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set tri-state to outp
```

In [ ]:
```
pulsecnt_address = ol.ip_dict['axi_gpio_pulsecnt']['phys_addr']
pulsecnt_register = MMIO(band_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as outp
pulsecnt_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set tri-state to
```

In [5]:
```
flags_address = ol.ip_dict['axi_gpio_flags']['phys_addr']
flags_register = MMIO(flags_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as outp
flags_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set tri-state to out
```

In [6]:
```
def duty(duty):
    duty_register.write(0x00, duty)
def dutypct(duty):
    duty_register.write(0x00, round((0x1F*2)/(100/duty)))
def band(band):
    band_register.write(0x00, band)
def pulsecnt(pulsecnt):
    pulsecnt_register.write(0x00, pulsecnt)
def enable(enable):
    if enable:
        flags_register.write(0x00, 1)
    else:
        flags_register.write(0x00, 0)
```

In [ ]:
```
enable(True)
dutypct(50)
band(5)
```

In [16]:
```
duty(0b01111)
```

In [19]:
```
dutypct(50)
```

In [12]:
```
band(5)
```

In [11]:
```
enable(False)
```

In [ ]:
```

```