```
train_length_register = MMIO(train_length_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as␣
 ↪outputs.
train_length_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set␣
 ↪tri-state to output

gate_delay_address = ol.ip_dict['axi_gpio_gate_delay']['phys_addr']
gate_delay_register = MMIO(gate_delay_address, RANGE)
# Write 0x00 to the tri-state register at offset 0x4 to configure the IO as␣
 ↪outputs.
gate_delay_register.write(0x4, 0x0) # Write 0x0 to location 0x4; Set tri-state␣
 ↪to output

def duty(duty):
    duty_register.write(0x00, duty)

def band(band):
    band_register.write(0x00, band)

def dutypct(duty):
    duty_register.write(0x00, round((0x1F*2)/(100/duty)))

def fire():
    flags_register.write(0x00, 1) # bit 0
    flags_register.write(0x00, 0)

def prime():
    flags_register.write(0x00, 2) # bit 1
    flags_register.write(0x00, 0)

def startdelay(startdelay):
    start_delay_register.write(0x0, startdelay);

def trainlength(trainlength):
    train_length_register.write(0x0, trainlength);

def gatedelay(gatedelay):
    gate_delay_register.write(0x0, gatedelay);
```

[7]:
```
duty(0x1F)
```

[8]:
```
dutypct(50)
```

[52]:
```
band(4)
```

[34]:
```
startdelay(20)
```