Timers

Timers are very useful for performing a more **complex behavior for a certain period of time**. Wait states (from **`wait1Msec`**) don't let the robot execute commands during the waiting period, which is fine for simple behaviors like moving forward. If calculations or other actions need to occur **during the timed period**, as with the line tracking behavior below, a Timer must be used.

```
task main()
{
 ClearTimer(T1);
 while(time1[T1] < 3000)
  {
    if(SensorValue(lightSensor) < 45)
    {
      motor[motorC]=50;
      motor[motorB]=0;
    }
    else
    {
      motor[motorC] = 0;
      motor[motorB] = 50;
    }
  }
```

Clear the Timer
Clearing the timer resets and starts the timer. You can choose to reset any of the timers, from T1 to T4.

Timer in the (condition)
This loop will run "while the timer's value is less than 3 seconds", i.e. **less than 3 seconds have passed since the reset**. The line tracking behavior inside the {body} will continue for 3

First, you must reset and start a timer by using the ClearTimer() command.
Here's how the command is set up:

```
ClearTimer(Timer_number);
```

The NXT has 4 built in timers: T1, T2, T3, and T4.
So if you wanted to reset and start Timer T1, you would type:

```
ClearTimer(T1);
```

Then, you can retrieve the value of the timer by using **`time1[T1]`**, **`time10[T1]`**, or **`time100[T1]`** depending on whether you want the output to be in 1, 10, or 100 millisecond values.

In the example above, you should see in the condition that we used time1[T1]. The robot will track a line until the value of the timer is less than 3 seconds. The program ends after 3 seconds.