Design and write a class which implements the methods

- `unsigned long millis()`
  Return the number of milliseconds / microseconds from the start of run time, looping back to zero after max-unsigned-long milliseconds (~50 days).
- `int micros()`
  Returns the remainder in microseconds.
  E.g if it has been 123456 microseconds since start, millis should return 123 and micros, 456.

Available API:

```
class Timer
{
      void start()               // start the timer
      void stop()                // stop the timer
      void reset()               // reset the timer
      unsigned short read_s()    // number of seconds since start
      unsigned short read_ms()   // number of milliseconds since start
      int read_us()              // number of microseconds since start
}

class Ticker
{

      //attach a function, specifying interval in seconds
      attach (Callback< void()> func, float t)
      //attach a member function, specifying interval in seconds
      attach (T *obj, M method, float t)

      //attach a function, specifying interval in microseconds
      attach_us (Callback< void()> func, us_timestamp_t t)
      //attach a member function, specifying interval in microseconds
      attach_us (T *obj, M method, us_timestamp_t t)

      //detach
      void detach()
}
```

Please note:

- Timer resets every max-int microseconds (~30 minutes)
- Please assume this is a multithreaded environment, and the context can be switched at any point, on any thread, for a significant amount of time (a few milliseconds).