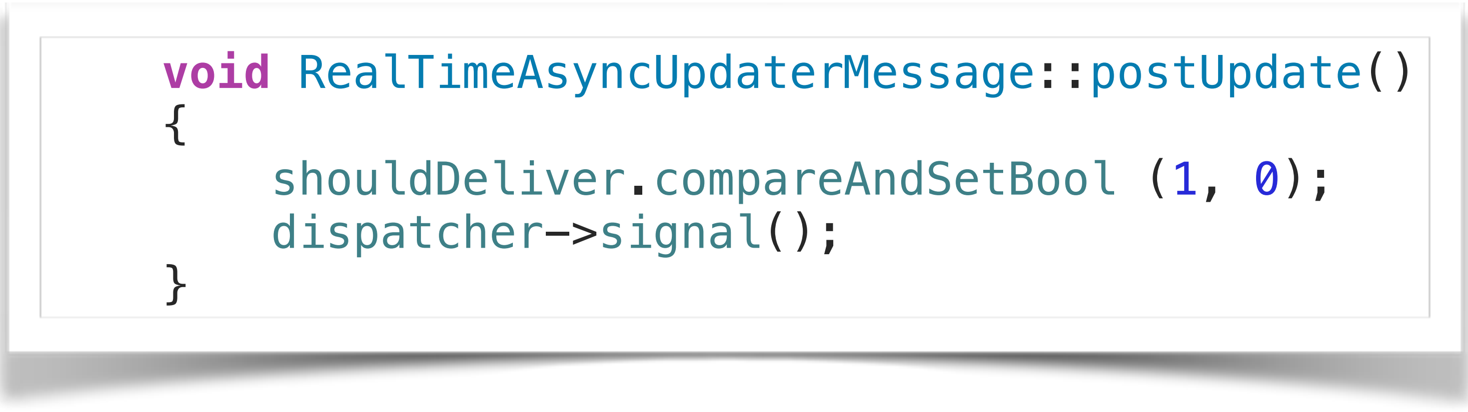
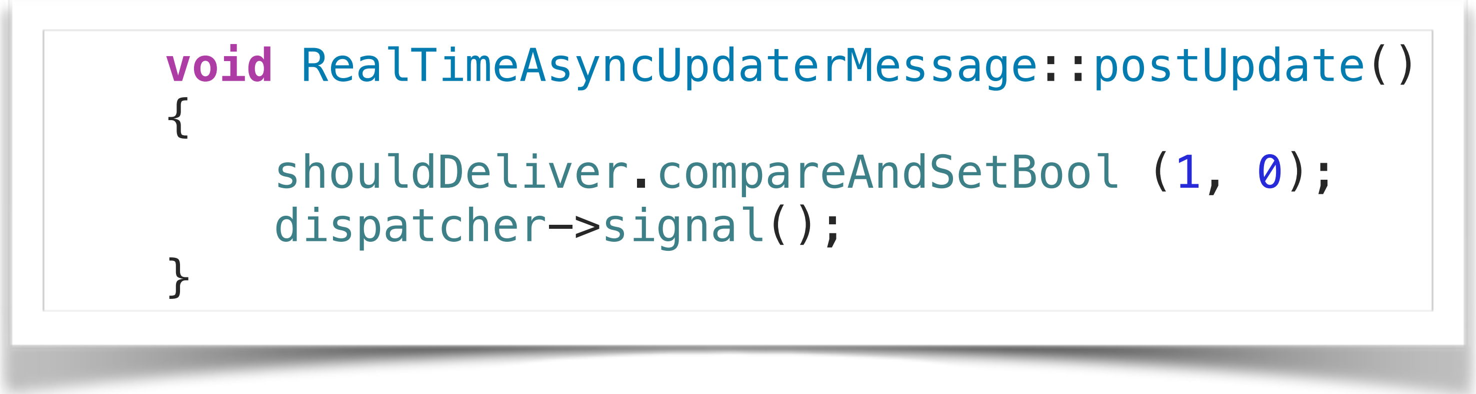
```cpp
class RealTimeAsyncUpdater::RealTimeAsyncUpdateDispatcher   : private Thread,
                                                              private AsyncUpdater
{
public:
    RealTimeAsyncUpdateDispatcher()
        : Thread ("RealTimeAsyncUpdateDispatcher")
    {
        startThread();
    }

    ~RealTimeAsyncUpdateDispatcher()
    {
        cancelPendingUpdate();
        isDestructing = true;
        serviceEvent.signal();
        stopThread (10000);
    }

    void add (RealTimeAsyncUpdaterMessage&);
    void remove (RealTimeAsyncUpdaterMessage&);

    void signal()
    {
        serviceEvent.signal();
    }

private:
    void run() override
    {
        while (! threadShouldExit())
        {
            if (! isDestructing.load())
                serviceEvent.wait (-1);

            triggerAsyncUpdate();
        }
    }

    void handleAsyncUpdate() override
    {
        serviceUpdaters();
    }

    void serviceUpdaters();

    CriticalSection lock;
    Array<RealTimeAsyncUpdaterMessage*> updaters;
    WaitableEvent serviceEvent;
    std::atomic<bool> isDestructing { false };
};
```



```
void RealTimeAsyncUpdaterMessage::postUpdate()
{
    shouldDeliver.compareAndSetBool (1, 0);
    dispatcher->signal();
}
```

```
class RealTimeAsyncUpdater::RealTimeAsyncUpdateDispatcher   : private Thread,
                                                              private AsyncUpdater
{
public:
    RealTimeAsyncUpdateDispatcher()
        : Thread ("RealTimeAsyncUpdateDispatcher")
    {
        startThread();
    }

    ~RealTimeAsyncUpdateDispatcher()
    {
        cancelPendingUpdate();
        isDestructing = true;
        serviceEvent.signal();
        stopThread (10000);
    }

    void add (RealTimeAsyncUpdaterMessage&);
    void remove (RealTimeAsyncUpdaterMessage&);

    void signal()
    {
        serviceEvent.signal();
    }

private:
    void run() override
    {
        while (! threadShouldExit())
        {
            if (! isDestructing.load())
                serviceEvent.wait (-1);

            triggerAsyncUpdate();
        }
    }

    void handleAsyncUpdate() override
    {
        serviceUpdaters();
    }

    void serviceUpdaters();

    CriticalSection lock;
    Array<RealTimeAsyncUpdaterMessage*> updaters;
    WaitableEvent serviceEvent;
    std::atomic<bool> isDestructing { false };
};
```



```
void RealTimeAsyncUpdaterMessage::postUpdate()
{
    shouldDeliver.compareAndSetBool (1, 0);
    dispatcher->signal();
}
```

juce::AsyncUpdater

Average = 20 microsecs, minimum = 5 microsecs, maximum = 102 microsecs

RealTimeAsyncUpdater (Timer Based)

Average = 41 millisecs, minimum = 239 microsecs, maximum = 92 millisecs

RealTimeAsyncUpdater (HighResolutionTimer Based)

Average = 4997 microsecs, minimum = 4643 microsecs, maximum = 5230 microsecs