Subject: Feature request: Synchronized timers?
Posted by steffen on Thu, 03 May 2012 06:29:38 GMT
View Forum Message <> Reply to Message

Hi,

I'm making a touch screen user interface for displaying information received by ModbusTCP. One part of this is showing alarms as flashing indicators, like ctrl's where I toggle the face color on a timer callback.

If more alarms is active at the same time, I would like them to flash "in sync".

In CtrlTimer, the lowest accessible function is SetTimerCallback, which only takes a delay parameter, I would like an overloaded function which also takes an initial delay parameter, so I could use a modulo of the GetTimeClick() as initial time value.

Current function:

```
void SetTimeCallback(int delay_ms, Callback cb, void *id) {
 Mutex::Lock __(sTimerLock);
 ASSERT(abs(delay_ms) < 0x40000000);
 LLOG("SetTimeCallback " << delay_ms << " " << id);
 sTimeCallback(GetTickCount() + abs(delay_ms), delay_ms, cb, id);
}
```

I simply wish for something like:

```
void SetTimeCallback(int initial_delay_ms, int delay_ms, Callback cb, void *id) {
 Mutex::Lock __(sTimerLock);
 ASSERT(abs(delay_ms) < 0x40000000);
 LLOG("SetTimeCallback " << delay_ms << " " << id);
 sTimeCallback(initial_delay_ms, delay_ms, cb, id);
}
```

Or should I do it in a completely different way?

Subject: Re: Feature request: Synchronized timers?
Posted by mirek on Thu, 03 May 2012 06:36:32 GMT
View Forum Message <> Reply to Message

steffen wrote on Thu, 03 May 2012 02:29Hi,

I'm making a touch screen user interface for displaying information received by ModbusTCP. One part of this is showing alarms as flashing indicators, like ctrl's where I toggle the face color on a timer callback.

If more alarms is active at the same time, I would like them to flash "in sync".

In CtrlTimer, the lowest accessible function is SetTimerCallback, which only takes a delay parameter, I would like an overloaded function which also takes an initial delay parameter, so I could use a modulo of the GetTimeClick() as initial time value.

Current function:

```
void SetTimeCallback(int delay_ms, Callback cb, void *id) {
 Mutex::Lock __(sTimerLock);
 ASSERT(abs(delay_ms) < 0x40000000);
 LLOG("SetTimeCallback " << delay_ms << " " << id);
 sTimeCallback(GetTickCount() + abs(delay_ms), delay_ms, cb, id);
}
```

I simply wish for something like:

```
void SetTimeCallback(int initial_delay_ms, int delay_ms, Callback cb, void *id) {
 Mutex::Lock __(sTimerLock);
 ASSERT(abs(delay_ms) < 0x40000000);
 LLOG("SetTimeCallback " << delay_ms << " " << id);
 sTimeCallback(initial_delay_ms, delay_ms, cb, id);
}
```

Or should I do it in a completely different way?

I would use 50Hz timer and deduced all timings from absolute time.