GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	drivers/source/PwmOut.cpp	**Lines:**	69	98	70.4 %
Date:	2021-05-06 12:39:05	**Branches:**	9	12	75.0 %

Line	Branch	Exec	Source
1			/* mbed Microcontroller Library
2			* Copyright (c) 2006-2019 ARM Limited
3			* SPDX-License-Identifier: Apache-2.0
4			*
5			* Licensed under the Apache License, Version 2.0 (the "License");
6			* you may not use this file except in compliance with the License.
7			* You may obtain a copy of the License at
8			*
9			* http://www.apache.org/licenses/LICENSE-2.0
10			*
11			* Unless required by applicable law or agreed to in writing, software
12			* distributed under the License is distributed on an "AS IS" BASIS,
13			* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14			* See the License for the specific language governing permissions and
15			* limitations under the License.
16			*/
17			
18			#include "drivers/PwmOut.h"
19			
20			
21			#if DEVICE_PWMOUT
22			
23			#include "platform/mbed_critical.h"
24			#include "platform/mbed_power_mgmt.h"
25			#include "platform/mbed_assert.h"
26			
27			namespace mbed {
28			
29		7	PwmOut::PwmOut(PinName pin) :
30			_pin(pin),

```
31                      _deep_sleep_locked(false),
32                      _initialized(false),
33                      _duty_cycle(0),
34          7           _period_us(0)
35                  {
36          7           PwmOut::init();
37          7       }
38
39                  PwmOut::PwmOut(const PinMap &pinmap) : _deep_sleep_locked(false)
40                  {
41                      core_util_critical_section_enter();
42                      pwmout_init_direct(&_pwm, &pinmap);
43                      core_util_critical_section_exit();
44                  }
45
46         14       PwmOut::~PwmOut()
47                  {
48          7           PwmOut::deinit();
49          7       }
50
51          4       void PwmOut::write(float value)
52                  {
53          4           core_util_critical_section_enter();
54          4           pwmout_write(&_pwm, value);
55          4           core_util_critical_section_exit();
56          4       }
57
58          5       float PwmOut::read()
59                  {
60          5           core_util_critical_section_enter();
61          5           float val = pwmout_read(&_pwm);
62          5           core_util_critical_section_exit();
63          5           return val;
64                  }
65
66                  void PwmOut::period(float seconds)
67                  {
68                      core_util_critical_section_enter();
69                      pwmout_period(&_pwm, seconds);
70                      core_util_critical_section_exit();
```

```
71          }
72
73          void PwmOut::period_ms(int ms)
74          {
75              core_util_critical_section_enter();
76              pwmout_period_ms(&_pwm, ms);
77              core_util_critical_section_exit();
78          }
79
80      4   void PwmOut::period_us(int us)
81          {
82      4       core_util_critical_section_enter();
83      4       pwmout_period_us(&_pwm, us);
84      4       core_util_critical_section_exit();
85      4   }
86
87      5   int PwmOut::read_period_us()
88          {
89      5       core_util_critical_section_enter();
90      5       auto val = pwmout_read_period_us(&_pwm);
91      5       core_util_critical_section_exit();
92      5       return val;
93          }
94
95          void PwmOut::pulsewidth(float seconds)
96          {
97              core_util_critical_section_enter();
98              pwmout_pulsewidth(&_pwm, seconds);
99              core_util_critical_section_exit();
100         }
101
102         void PwmOut::pulsewidth_ms(int ms)
103         {
104             core_util_critical_section_enter();
105             pwmout_pulsewidth_ms(&_pwm, ms);
106             core_util_critical_section_exit();
107         }
108
109         void PwmOut::pulsewidth_us(int us)
110         {
```

```
111                 core_util_critical_section_enter();
112                 pwmout_pulsewidth_us(&_pwm, us);
113                 core_util_critical_section_exit();
114             }
115
116             int PwmOut::read_pulsewitdth_us()
117             {
118                 core_util_critical_section_enter();
119                 auto val = pwmout_read_pulsewidth_us(&_pwm);
120                 core_util_critical_section_exit();
121                 return val;
122             }
123
124           6 void PwmOut::suspend()
125             {
126           6     core_util_critical_section_enter();
127       ✓✓  6     if (_initialized) {
128           5         _duty_cycle = PwmOut::read();
129           5         _period_us = PwmOut::read_period_us();
130           5         PwmOut::deinit();
131             }
132           6     core_util_critical_section_exit();
133           6 }
134
135           5 void PwmOut::resume()
136             {
137           5     core_util_critical_section_enter();
138       ✓✓  5     if (!_initialized) {
139           4         PwmOut::init();
140           4         PwmOut::write(_duty_cycle);
141           4         PwmOut::period_us(_period_us);
142             }
143           5     core_util_critical_section_exit();
144           5 }
145
146          11 void PwmOut::lock_deep_sleep()
147             {
148       ✓✗ 11     if (_deep_sleep_locked == false) {
149          11         sleep_manager_lock_deep_sleep();
150          11         _deep_sleep_locked = true;
```

```
151                 }
152          11 }
153
154          11 void PwmOut::unlock_deep_sleep()
155             {
156      ✓✗  11     if (_deep_sleep_locked == true) {
157          11         sleep_manager_unlock_deep_sleep();
158          11         _deep_sleep_locked = false;
159             }
160          11 }
161
162          11 void PwmOut::init()
163             {
164          11     core_util_critical_section_enter();
165
166      ✓✗  11     if (!_initialized) {
167          11         pwmout_init(&_pwm, _pin);
168          11         lock_deep_sleep();
169          11         _initialized = true;
170             }
171
172          11     core_util_critical_section_exit();
173          11 }
174
175          12 void PwmOut::deinit()
176             {
177          12     core_util_critical_section_enter();
178
179      ✓✓  12     if (_initialized) {
180          11         pwmout_free(&_pwm);
181          11         unlock_deep_sleep();
182          11         _initialized = false;
183             }
184
185          12     core_util_critical_section_exit();
186          12 }
187
188             } // namespace mbed
189
190             #endif // #if DEVICE_PWMOUT
```