GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	drivers/source/Watchdog.cpp	**Lines:**	29	38	76.3 %
Date:	2021-05-06 12:39:05	**Branches:**	11	20	55.0 %

Line	Branch	Exec	Source
1			/*
2			* Copyright (c) 2018-2019 Arm Limited and affiliates.
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			#ifdef DEVICE_WATCHDOG
18			
19			#include "drivers/Watchdog.h"
20			
21			namespace mbed {
22			
23		1	Watchdog::Watchdog() : _running(false)
24			{
25		1	}
26			
27		1	Watchdog::~Watchdog()
28			{
29		1	}
30			

```
31          1  bool Watchdog::start(uint32_t timeout)
32             {
33  ✓✗✗✓    1      MBED_ASSERT(timeout <= get_max_timeout());
34    ✗✓    1      MBED_ASSERT(timeout > 0);
35
36    ✓✗    1      core_util_critical_section_enter();
37             watchdog_config_t config;
38          1  config.timeout_ms = timeout;
39    ✓✗    1      watchdog_status_t sts = hal_watchdog_init(&config);
40    ✓✗    1      if (sts == WATCHDOG_STATUS_OK) {
41          1          _running = true;
42             }
43    ✓✗    1      core_util_critical_section_exit();
44          1      return (sts == WATCHDOG_STATUS_OK);
45             }
46
47             bool Watchdog::start()
48             {
49
50                 return start(get_max_timeout());
51             }
52
53          2  bool Watchdog::stop()
54             {
55                 watchdog_status_t sts;
56          2      bool msts = true;
57          2      core_util_critical_section_enter();
58    ✓✓    2      if (_running) {
59          1          sts = hal_watchdog_stop();
60    ✗✓    1          if (sts != WATCHDOG_STATUS_OK) {
61                         msts = false;
62                     } else {
63          1              _running = false;
64                     }
65
66                 } else {
67          1          msts = false;
68                 }
69          2      core_util_critical_section_exit();
70          2      return msts;
```

```
71          }
72
73          void Watchdog::kick()
74          {
75              core_util_critical_section_enter();
76              hal_watchdog_kick();
77              core_util_critical_section_exit();
78          }
79
80          bool Watchdog::is_running() const
81          {
82              return _running;
83          }
84
85        1 uint32_t Watchdog::get_timeout() const
86          {
87        1     return hal_watchdog_get_reload_value();
88          }
89
90        2 uint32_t Watchdog::get_max_timeout() const
91          {
92    ✓✗  2     const watchdog_features_t features = hal_watchdog_get_platform_features();
93        2     return features.max_timeout;
94          }
95
96          } // namespace mbed
97
98
99          #endif // DEVICE_WATCHDOG
```