GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	storage/blockdevice/source/FlashSimBlockDevice.cpp	Lines:	95	106	89.6 %
Date:	2021-05-06 12:39:05	Branches:	47	64	73.4 %

Line	Branch	Exec	Source
1			/* mbed Microcontroller Library
2			* Copyright (c) 2018 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 "blockdevice/FlashSimBlockDevice.h"
19			#include "platform/mbed_assert.h"
20			#include "platform/mbed_atomic.h"
21			#include <algorithm>
22			#include <stdlib.h>
23			#include <string.h>
24			#include "mbed_assert.h"
25			
26			namespace mbed {
27			
28			static const bd_size_t min_blank_buf_size = 32;
29			
30		6	static inline uint32_t align_up(bd_size_t val, bd_size_t size)

```
31                {
32          6         return (((val - 1) / size) + 1) * size;
33                }
34
35          6     FlashSimBlockDevice::FlashSimBlockDevice(BlockDevice *bd, uint8_t erase_value) :
36                    _erase_value(erase_value), _blank_buf_size(0),
37          6         _blank_buf(0), _bd(bd), _init_ref_count(0), _is_initialized(false)
38                {
39    ✗✓    6         MBED_ASSERT(bd);
40          6     }
41
42         12     FlashSimBlockDevice::~FlashSimBlockDevice()
43                {
44          6         deinit();
45    ✓✗    6         delete[] _blank_buf;
46          6     }
47
48          6     int FlashSimBlockDevice::init()
49                {
50                    int err;
51          6         uint32_t val = core_util_atomic_incr_u32(&_init_ref_count, 1);
52
53    ✗✓    6         if (val != 1) {
54                        return BD_ERROR_OK;
55                    }
56
57          6         err = _bd->init();
58    ✗✓    6         if (err) {
59                        goto fail;
60                    }
61          6         _blank_buf_size = align_up(min_blank_buf_size, _bd->get_program_size());
62    ✓✗    6         if (!_blank_buf) {
63          6             _blank_buf = new uint8_t[_blank_buf_size];
64          6             memset(_blank_buf, 0, _blank_buf_size);
65    ✗✓    6             MBED_ASSERT(_blank_buf);
66                    }
67
68          6         _is_initialized = true;
69          6         return BD_ERROR_OK;
70
```

```
71                  fail:
72                      _is_initialized = false;
73                      _init_ref_count = 0;
74                      return err;
75                  }
76
77          13      int FlashSimBlockDevice::deinit()
78                  {
79      ✓✓  13          if (!_is_initialized) {
80           7              return BD_ERROR_OK;
81                      }
82
83           6          uint32_t val = core_util_atomic_decr_u32(&_init_ref_count, 1);
84
85      ✗✓   6          if (val) {
86                          return BD_ERROR_OK;
87                      }
88
89           6          _is_initialized = false;
90           6          return _bd->deinit();
91                  }
92
93           2      int FlashSimBlockDevice::sync()
94                  {
95      ✓✓   2          if (!_is_initialized) {
96           1              return BD_ERROR_DEVICE_ERROR;
97                      }
98
99           1          return _bd->sync();
100                 }
101
102          2      bd_size_t FlashSimBlockDevice::get_read_size() const
103                 {
104     ✓✓   2          if (!_is_initialized) {
105          1              return 0;
106                     }
107
108          1          return _bd->get_read_size();
109                 }
110
```

```
111          14  bd_size_t FlashSimBlockDevice::get_program_size() const
112              {
113      ✓✓  14      if (!_is_initialized) {
114          1           return 0;
115                  }
116
117          13      return _bd->get_program_size();
118              }
119
120          2   bd_size_t FlashSimBlockDevice::get_erase_size() const
121              {
122      ✓✓  2       if (!_is_initialized) {
123          1           return 0;
124                  }
125
126          1       return _bd->get_erase_size();
127              }
128
129          10  bd_size_t FlashSimBlockDevice::get_erase_size(bd_addr_t addr) const
130              {
131      ✓✓  10      if (!_is_initialized) {
132          1           return 0;
133                  }
134
135          9       return _bd->get_erase_size(addr);
136              }
137
138          10  bd_size_t FlashSimBlockDevice::size() const
139              {
140      ✓✓  10      if (!_is_initialized) {
141          1           return 0;
142                  }
143
144          9       return _bd->size();
145              }
146
147          1   int FlashSimBlockDevice::read(void *b, bd_addr_t addr, bd_size_t size)
148              {
149      ✓✗  1       if (!_is_initialized) {
150          1           return BD_ERROR_DEVICE_ERROR;
```

```
151                 }
152
153                 return _bd->read(b, addr, size);
154             }
155
156         7   int FlashSimBlockDevice::program(const void *b, bd_addr_t addr, bd_size_t size)
157             {
158     ✓✓  7       if (!_is_initialized) {
159         1           return BD_ERROR_DEVICE_ERROR;
160                 }
161
162   ✓✗✓✓  6       if (!is_valid_program(addr, size)) {
163         1           return BD_ERROR_DEVICE_ERROR;
164                 }
165
166         5       bd_addr_t curr_addr = addr;
167         5       bd_size_t curr_size = size;
168
169         5       const uint8_t *buf = (const uint8_t *) b;
170     ✓✓ 11       while (curr_size) {
171         5           bd_size_t read_size = std::min(_blank_buf_size, curr_size);
172     ✓✗  5           int ret = _bd->read(_blank_buf, curr_addr, read_size);
173     ✗✓  5           if (ret) {
174                         return ret;
175                     }
176     ✓✓1541          for (bd_size_t i = 0; i < read_size; i++) {
177                         // Allow either programming on blanks or programming the same value
178                         // (as real flash devices do)
179   ✓✓✓✗1538              if ((_blank_buf[i] != _erase_value) && (_blank_buf[i] != *buf)) {
180         2                   return BD_ERROR_NOT_ERASED;
181                         }
182      1536               buf++;
183                     }
184         3           curr_addr += read_size;
185         3           curr_size -= read_size;
186                 }
187
188     ✓✗  3       return _bd->program(b, addr, size);
189             }
190
```

```
191                 5  int FlashSimBlockDevice::erase(bd_addr_t addr, bd_size_t size)
192                    {
193              ✓✓    5      if (!_is_initialized) {
194                    1          return BD_ERROR_DEVICE_ERROR;
195                           }
196
197           ✓✗✓✓     4      if (!is_valid_erase(addr, size)) {
198                    1          return BD_ERROR_DEVICE_ERROR;
199                           }
200
201                    3      bd_addr_t curr_addr = addr;
202                    3      bd_size_t curr_size = size;
203
204                    3      memset(_blank_buf, _erase_value, (unsigned int) _blank_buf_size);
205
206              ✓✓    9      while (curr_size) {
207                    3          bd_size_t prog_size = std::min(_blank_buf_size, curr_size);
208              ✓✗    3          int ret = _bd->program(_blank_buf, curr_addr, prog_size);
209              ✗✓    3          if (ret) {
210                                     return ret;
211                               }
212                    3          curr_addr += prog_size;
213                    3          curr_size -= prog_size;
214                           }
215
216                    3      return BD_ERROR_OK;
217                    }
218
219                    1  int FlashSimBlockDevice::get_erase_value() const
220                    {
221                    1      return _erase_value;
222                    }
223
224                    1  const char *FlashSimBlockDevice::get_type() const
225                    {
226              ✓✗    1      if (_bd != NULL) {
227                    1          return _bd->get_type();
228                           }
229
230                           return NULL;
```

```
231             }
232
233         } // namespace mbed
```

Generated by: GCOVR (Version 4.2)