GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
File:	storage/blockdevice/source/ChainingBlockDevice.cpp	Lines:	116	141	82.3 %
Date:	2021-05-06 12:39:05	Branches:	63	114	55.3 %

Line	Branch	Exec	Source
1			/* mbed Microcontroller Library
2			* Copyright (c) 2017 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/ChainingBlockDevice.h"
19			#include "platform/mbed_atomic.h"
20			#include "platform/mbed_assert.h"
21			
22			namespace mbed {
23			
24		3	ChainingBlockDevice::ChainingBlockDevice(BlockDevice **bds, size_t bd_count)
25			: _bds(bds), _bd_count(bd_count)
26			, _read_size(0), _program_size(0), _erase_size(0), _size(0)
27		3	, _erase_value(-1), _init_ref_count(0), _is_initialized(false)
28			{
29		3	}
30			

```
31          9 static bool is_aligned(uint64_t x, uint64_t alignment)
32            {
33          9     return (x / alignment) * alignment == x;
34            }
35
36          3 int ChainingBlockDevice::init()
37            {
38                int err;
39          3     uint32_t val = core_util_atomic_incr_u32(&_init_ref_count, 1);
40
41    ✗✓    3     if (val != 1) {
42                    return BD_ERROR_OK;
43                }
44
45          3     _read_size = 0;
46          3     _program_size = 0;
47          3     _erase_size = 0;
48          3     _erase_value = -1;
49          3     _size = 0;
50
51                // Initialize children block devices, find all sizes and
52                // assert that block sizes are similar. We can't do this in
53                // the constructor since some block devices may need to be
54                // initialized before they know their block size/count
55    ✓✓    9     for (size_t i = 0; i < _bd_count; i++) {
56          6         err = _bds[i]->init();
57    ✗✓    6         if (err) {
58                        goto fail;
59                    }
60
61          6         bd_size_t read = _bds[i]->get_read_size();
62  ✓✓✓✗ ✓✗✓✗ 6         if (i == 0 || (read >= _read_size && is_aligned(read, _read_size))) {
63          6             _read_size = read;
64                    } else {
65                        MBED_ASSERT(_read_size > read && is_aligned(_read_size, read));
66                    }
67
68          6         bd_size_t program = _bds[i]->get_program_size();
69  ✓✓✓✗ ✓✗✓✗ 6         if (i == 0 || (program >= _program_size && is_aligned(program, _program_size))) {
```

```
70          6              _program_size = program;
71                     } else {
72                         MBED_ASSERT(_program_size > program && is_aligned(_program_size, program));
73                     }
74
75          6          bd_size_t erase = _bds[i]->get_erase_size();
76  ✓✓✓✗    6          if (i == 0 || (erase >= _erase_size && is_aligned(erase, _erase_size))) {
    ✓✗✓✗
77          6              _erase_size = erase;
78                     } else {
79                         MBED_ASSERT(_erase_size > erase && is_aligned(_erase_size, erase));
80                     }
81
82          6          int value = _bds[i]->get_erase_value();
83  ✓✓✓✓    6          if (i == 0 || value == _erase_value) {
84          5              _erase_value = value;
85                     } else {
86          1              _erase_value = -1;
87                     }
88
89          6          _size += _bds[i]->size();
90                 }
91
92          3          _is_initialized = true;
93          3          return BD_ERROR_OK;
94
95                 fail:
96                     _is_initialized = false;
97                     _init_ref_count = 0;
98                     return err;
99                 }
100
101         2      int ChainingBlockDevice::deinit()
102                {
103 ✗✓      2          if (!_is_initialized) {
104                        return BD_ERROR_OK;
105                    }
106
107         2          uint32_t val = core_util_atomic_decr_u32(&_init_ref_count, 1);
108
```

```
109  ✗✓  2      if (val) {
110              return BD_ERROR_OK;
111          }
112
113  ✓✓  6      for (size_t i = 0; i < _bd_count; i++) {
114      4          int err = _bds[i]->deinit();
115  ✗✓  4          if (err) {
116                  return err;
117              }
118          }
119
120      2      _is_initialized = false;
121      2      return BD_ERROR_OK;
122      }
123
124      1  int ChainingBlockDevice::sync()
125      {
126  ✓✗  1      if (!_is_initialized) {
127      1          return BD_ERROR_DEVICE_ERROR;
128          }
129
130              for (size_t i = 0; i < _bd_count; i++) {
131                  int err = _bds[i]->sync();
132                  if (err) {
133                      return err;
134                  }
135              }
136
137              return 0;
138      }
139
140      2  int ChainingBlockDevice::read(void *b, bd_addr_t addr, bd_size_t size)
141      {
142  ✓✓  2      if (!_is_initialized) {
143      1          return BD_ERROR_DEVICE_ERROR;
144          }
145
146  ✗✓  1      if (!is_valid_read(addr, size)) {
147              return BD_ERROR_DEVICE_ERROR;
148          }
```

```
149                                
150          1      uint8_t *buffer = static_cast<uint8_t *>(b);
151                                
152                 // Find block devices containing blocks, may span multiple block devices
153   ✓✓✓✗   3      for (size_t i = 0; i < _bd_count && size > 0; i++) {
154          2          bd_size_t bdsize = _bds[i]->size();
155                                
156     ✓✗   2          if (addr < bdsize) {
157          2              bd_size_t read = size;
158     ✓✓   2              if (addr + read > bdsize) {
159          1                  read = bdsize - addr;
160                             }
161                                
162          2              int err = _bds[i]->read(buffer, addr, read);
163     ✗✓   2              if (err) {
164                             return err;
165                             }
166                                
167          2              buffer += read;
168          2              addr += read;
169          2              size -= read;
170                         }
171                                
172          2          addr -= bdsize;
173                     }
174                                
175          1      return 0;
176                 }
177                                
178          2  int ChainingBlockDevice::program(const void *b, bd_addr_t addr, bd_size_t size)
179                 {
180     ✓✓   2      if (!_is_initialized) {
181          1          return BD_ERROR_DEVICE_ERROR;
182                     }
183                                
184     ✗✓   1      if (!is_valid_program(addr, size)) {
185                         return BD_ERROR_DEVICE_ERROR;
186                     }
187                                
188          1      const uint8_t *buffer = static_cast<const uint8_t *>(b);
```

```
189
190                 // Find block devices containing blocks, may span multiple block devices
191 ✓✓✓✗       3      for (size_t i = 0; i < _bd_count && size > 0; i++) {
192                 2          bd_size_t bdsize = _bds[i]->size();
193
194   ✓✗            2          if (addr < bdsize) {
195                 2              bd_size_t program = size;
196   ✓✓            2              if (addr + program > bdsize) {
197                 1                  program = bdsize - addr;
198                                  }
199
200                 2              int err = _bds[i]->program(buffer, addr, program);
201   ✗✓            2              if (err) {
202                                      return err;
203                                  }
204
205                 2              buffer += program;
206                 2              addr += program;
207                 2              size -= program;
208                              }
209
210                 2          addr -= bdsize;
211                         }
212
213                 1      return 0;
214                     }
215
216                 1  int ChainingBlockDevice::erase(bd_addr_t addr, bd_size_t size)
217                     {
218   ✗✓            1      if (!_is_initialized) {
219                                  return BD_ERROR_DEVICE_ERROR;
220                         }
221
222   ✗✓            1      if (!is_valid_erase(addr, size)) {
223                                  return BD_ERROR_DEVICE_ERROR;
224                         }
225
226                         // Find block devices containing blocks, may span multiple block devices
227 ✓✓✓✗       3      for (size_t i = 0; i < _bd_count && size > 0; i++) {
228                 2          bd_size_t bdsize = _bds[i]->size();
```

```
229
230 ✓✗ 2         if (addr < bdsize) {
231    2             bd_size_t erase = size;
232 ✓✓ 2             if (addr + erase > bdsize) {
233    1                 erase = bdsize - addr;
234                  }
235
236    2             int err = _bds[i]->erase(addr, erase);
237 ✗✓ 2             if (err) {
238                      return err;
239                  }
240
241    2             addr += erase;
242    2             size -= erase;
243              }
244
245    2         addr -= bdsize;
246          }
247
248    1     return 0;
249      }
250
251    3 bd_size_t ChainingBlockDevice::get_read_size() const
252      {
253    3     return _read_size;
254      }
255
256    3 bd_size_t ChainingBlockDevice::get_program_size() const
257      {
258    3     return _program_size;
259      }
260
261    2 bd_size_t ChainingBlockDevice::get_erase_size() const
262      {
263    2     return _erase_size;
264      }
265
266    3 bd_size_t ChainingBlockDevice::get_erase_size(bd_addr_t addr) const
267      {
268 ✓✓ 3     if (!_is_initialized) {
```

```
269        1          return 0;
270                   }
271
272        2          bd_addr_t bd_start_addr = 0;
273   ✓✗   3          for (size_t i = 0; i < _bd_count; i++) {
274        3              bd_size_t bdsize = _bds[i]->size();
275   ✓✓   3              if (addr < (bd_start_addr + bdsize)) {
276        2                  return _bds[i]->get_erase_size(addr - bd_start_addr);
277                       }
278        1              bd_start_addr += bdsize;
279                   }
280
281                   // Getting here implies an illegal address
282                   MBED_ASSERT(0);
283                   return 0; // satisfy compiler
284               }
285
286        2      int ChainingBlockDevice::get_erase_value() const
287               {
288        2          return _erase_value;
289               }
290
291        4      bd_size_t ChainingBlockDevice::size() const
292               {
293        4          return _size;
294               }
295
296        1      const char *ChainingBlockDevice::get_type() const
297               {
298        1          return "CHAINING";
299               }
300
301            } // namespace mbed
```

Generated by: GCOVR (Version 4.2)