

```

1  /*
2   * dma_driver.h
3   *
4   * Created on: Dec 11, 2018
5   * Author: Dominic Doty
6   */
7
8  #ifndef DMA_DRIVER_H_
9  #define DMA_DRIVER_H_
10
11 /* INCLUDES */
12 #include "MKL25Z4.h"
13 #include "stddef.h"
14 #include "fsl_common.h"
15 #include "fsl_clock.h"
16
17
18 /* DEFINES & TYPEDEFS */
19
20 // DMA Errors
21 typedef enum
22 {
23     DMA_ERROR_SUCCESS,
24     DMA_ERROR_NULL_PTR,
25     DMA_ERROR_BAD_ADDR,
26     DMA_ERROR_BUSY,
27     DMA_ERROR_BYTE_COUNT,
28     DMA_ERROR_UNKNOWN_DMA
29 } dma_error;
30
31 // DMA Channels
32 typedef enum
33 {
34     DMA_CHANNEL_0,
35     DMA_CHANNEL_1,
36     DMA_CHANNEL_2,
37     DMA_CHANNEL_3
38 } dma_channel;
39
40 // DMA Data Sizes
41 typedef enum
42 {
43     DMA_SIZE_32,
44     DMA_SIZE_8,
45     DMA_SIZE_16
46 } dma_size;
47
48 // DMA Modulos
49 typedef enum
50 {
51     DMA_MOD_NONE,
52     DMA_MOD_16b,
53     DMA_MOD_32b,
54     DMA_MOD_64b,
55     DMA_MOD_128b,
56     DMA_MOD_256b,
57     DMA_MOD_512b,
58     DMA_MOD_1k,
59     DMA_MOD_2k,
60     DMA_MOD_4k,
61     DMA_MOD_8k,
62     DMA_MOD_16k,
63     DMA_MOD_32k,
64     DMA_MOD_64k,
65     DMA_MOD_128k,
66     DMA_MOD_256k
67 } dma_mod;
68
69 // DMA Link Channel Mode

```

```

70 typedef enum
71 {
72     DMA_LINK_NONE,
73     DMA_LINK_LCH1_ON_CS_LCH2_AND_BCR_ZERO,
74     DMA_LINK_LCH1_ON_CS,
75     DMA_LINK_LCH1_ON_BCR_ZERO
76 } dma_link_mode;
77
78 // DMA Link Channels
79 typedef enum
80 {
81     DMA_LINK_DMA_CHAN_0,
82     DMA_LINK_DMA_CHAN_1,
83     DMA_LINK_DMA_CHAN_2,
84     DMA_LINK_DMA_CHAN_3
85 } dma_link_channel;
86
87 // DMA Mux Configuration
88 typedef struct
89 {
90     DMAMUX_Type* dma_mux;
91     dma_channel channel;
92     bool channel_enable;
93     bool trigger_mode;
94     dma_request_source_t slot;
95 } dma_mux_config;
96
97 #define DMA_MUX_CONFIG_DEFAULT \
98 { \
99     .dma_mux = DMAMUX0, \
100     .channel = DMA_CHANNEL_0, \
101     .channel_enable = false, \
102     .trigger_mode = false, \
103     .slot = kDmaRequestMux0ADC0 \
104 }
105
106 // DMA Configuration
107 typedef struct
108 {
109     DMA_Type* dma;
110     dma_channel channel;
111     volatile void* src_addr;
112     volatile void* dest_addr;
113     uint32_t byte_count;
114     bool interrupt;
115     bool peripheral_en;
116     bool steal_cycles;
117     bool auto_align;
118     bool async_en;
119     bool src_inc;
120     dma_size src_size;
121     dma_mod src_mod;
122     bool dest_inc;
123     dma_size dest_size;
124     dma_mod dest_mod;
125     bool auto_disable_req;
126     dma_link_mode link_mode;
127     dma_link_channel link_chan_1;
128     dma_link_channel link_chan_2;
129     bool start;
130 } dma_init_config;
131
132 #define DMA_INIT_CONFIG_DEFAULT \
133 { \
134     .dma = NULL, \
135     .channel = DMA_CHANNEL_0, \
136     .src_addr = NULL, \
137     .dest_addr = NULL, \
138     .byte_count = 0, \
139     .interrupt = false, \

```

```

140     .peripheral_en = false,           \
141     .steal_cycles = false,           \
142     .auto_align = false,             \
143     .async_en = false,               \
144     .src_inc = false,                \
145     .src_size = DMA_SIZE_32,         \
146     .src_mod = DMA_MOD_NONE,         \
147     .dest_inc = false,               \
148     .dest_size = DMA_SIZE_32,        \
149     .dest_mod = DMA_MOD_NONE,        \
150     .auto_disable_req = false,       \
151     .link_mode = DMA_LINK_NONE,      \
152     .link_chan_1 = DMA_LINK_DMA_CHAN_0, \
153     .link_chan_2 = DMA_LINK_DMA_CHAN_0, \
154     .start = false                   \
155 }
156
157 /* FUNCTION DECLARATIONS */
158
159 // DMA Initialization
160 dma_error dma_init(dma_init_config* config);
161
162 // DMA Mux Initialization
163 dma_error dma_mux_init(dma_mux_config* config);
164
165 // Enable or Disable a Mux Channel
166 void dma_mux_channel_enable(DMAMUX_Type* dma_mux, dma_channel channel, bool enable);
167
168 // Used to restart a DMA transfer on an already configured DMA Channel (resets peripheral_en)
169 void dma_transfer_restart(DMA_Type* dma, dma_channel channel, volatile void* buffer_ptr, uint32_t byte_count);
170
171 #endif /* DMA_DRIVER_H_ */

```