

```

/**
 * @file project_test.c
 * @brief CMocka unit tests
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-03-04
 */

#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <stdint.h>

#include "memory.h"
#include "conversion.h"
#include "circbuf.h"
#include "data.h"

#define BASE_16 (16)
#define BASE_10 (10)

#define DATA_SET_SIZE_W (10)
#define MEM_SET_SIZE_B (32)
#define MEM_SET_SIZE_W (8)
#define MEM_ZERO_LENGTH (16)
#define MEM_SIZE_EVEN 8
#define MEM_SIZE_ODD 9
#define MEM_SIZE_CHAR 256

#define TEST_MEMMOVE_LENGTH (16)
#define BUFFER_LENGTH (8)

static void test_itoa_invalid_ptr(void **state) {
    uint8_t * ptr = NULL;
    int32_t num = -4096;
    // ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );
    uint32_t digits = my_itoa( num, ptr, BASE_16 );
    assert_int_equal(digits,0);
}

static void test_itoa_zero_integer(void **state) {
    uint8_t * ptr;
    int32_t num = 0;
    ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

    assert_non_null(ptr);
    // ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );
    uint32_t digits = my_itoa( num, ptr, BASE_16 );
    assert_int_equal(digits,1);
}

static void test_itoa_max_sized_integer(void **state) {

```

```

uint8_t * ptr;
int32_t num = 0;
ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

    assert_non_null(ptr);
// ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );
    uint32_t digits = my_itoa( num, ptr, BASE_16);
    assert_int_equal(digits,1);
}

static void test_atoi_invalid_ptr(void **state) {
    uint8_t * ptr = NULL;
    // int32_t num = -4096;
    uint32_t digits = 5;
    int32_t value;

    // digits = my_itoa( num, ptr, BASE_16);
    value = my_atoi( ptr, digits, BASE_16);
    assert_int_equal(value,0);
}

static void test_atoi_zero_integer(void **state) {
    uint8_t * ptr ;
    // int32_t num = -4096;
    uint32_t digits = 0;
    int32_t value;
ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

    assert_non_null(ptr);

    // digits = my_itoa( num, ptr, BASE_16);
    value = my_atoi( ptr, digits, BASE_16);
    assert_int_equal(value,0);
}

static void test_atoi_max_sized_integer(void **state){
    int32_t value;
    //ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );
//assert_non_null(ptr);

    // digits = my_itoa( num, ptr, BASE_16);
    value = my_atoi((uint8_t *)"2147483647", 10, BASE_10);
    assert_int_equal(value,INT32_MAX);
}

static void test_data1(void **state) {
    (void) state; /* unused */

    uint8_t * ptr;
    int32_t num = -4096;
    uint32_t digits;
    int32_t value;

    ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );
}

```

```

assert_non_null(ptr);

digits = my_itoa( num, ptr, BASE_16);
value = my_atoi( ptr, digits, BASE_16);
free_words( (uint32_t*)ptr );

assert_int_equal(value, num);
}

static void test_data2(void **state) {
    uint8_t * ptr;
    int32_t num = 123456;
    uint32_t digits;
    int32_t value;

    ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

    assert_non_null(ptr);

    digits = my_itoa( num, ptr, BASE_10);
    value = my_atoi( ptr, digits, BASE_10);
    free_words( (uint32_t*)ptr );

    assert_int_equal(value, num);
}

static void test_memmove1(void **state) {
    uint8_t i;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    set = (uint8_t*) reserve_words( MEM_SET_SIZE_W );

    assert_non_null(set);

    ptra = &set[0];
    ptrb = &set[16];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++)
    {
        set[i] = i;
    }

    my_memmove(ptra, ptrb, TEST_MEMMOVE_LENGTH);

    for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
    {
        assert_int_equal(set[i + 16], i);
    }
    free_words( (uint32_t*)set );
}

```

```

static void test_memmove2(void **state) {
    uint8_t i;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    set = (uint8_t*) reserve_words(MEM_SET_SIZE_W);

    assert_non_null(set);
    ptra = &set[0];
    ptrb = &set[8];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++) {
        set[i] = i;
    }

    my_memmove(ptra, ptrb, TEST_MEMMOVE_LENGTH);

    for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
    {
        assert_int_equal(set[i + 8],i);
    }

    free_words( (uint32_t*)set );
}

static void test_memmove3(void **state) {
    uint8_t i;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    set = (uint8_t*)reserve_words( MEM_SET_SIZE_W );
    assert_non_null(set);
    ptra = &set[8];
    ptrb = &set[0];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++)
    {
        set[i] = i;
    }

    my_memmove(ptra, ptrb, TEST_MEMMOVE_LENGTH);

    for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
    {
        assert_int_equal(set[i],i+8);
    }

    free_words( (uint32_t*)set );
}

```

```

}

static void test_memmove_null_Ptr(void **state) {
    uint8_t * set = (uint8_t*) malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * source= &set[0];
    uint8_t * dest= NULL; //Pass a NULL destination pointer
    uint8_t * status = NULL;
    status = my_memmove(source,dest,(MEM_SIZE_EVEN/2));
    //check whether the pointer returned is NULL
    assert_ptr_equal(cast_ptr_to_largest_integral_type(status),NULL);
}

static void test_memmove_no_overlap(void **state) {
    uint8_t * set = (uint8_t*) malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * source= &set[0];
    uint8_t * dest= &set[(MEM_SIZE_EVEN/2)];
    uint8_t * status = NULL;
    //Initialize the memory to test values
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++)
        set[i]=i;
    status = my_memmove(source,dest,(MEM_SIZE_EVEN/2));
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++) {
        //Check each of the 5 values against the source
        assert_int_equal(i,* (status+i));
    }
}

static void test_memmove_src_in_dst(void **state) {
    uint8_t * set = (uint8_t*) malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * source= &set[0];
    uint8_t * dest= &set[2];
    uint8_t * status = NULL;
    //Initialize the memory to test values
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++)
        set[i]=i;
    status = my_memmove(source,dest,(MEM_SIZE_EVEN/2));
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++) {
        //Check each of the 5 values against the source
        assert_int_equal(i,* (status+i));
    }
}

static void test_memmove_dst_in_src(void **state) {
    uint8_t * set = (uint8_t*) malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * source= &set[2];
    uint8_t * dest= &set[0];
    uint8_t * status = NULL;
    //Initialize the memory to test values
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++)
        set[i+2]=i;
    status = my_memmove(source,dest,(MEM_SIZE_EVEN/2));
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++) {

```

```

        //Check each of the 5 values against the source
        assert_int_equal(i,* (status+i));
    }

}

static void test_memmove_dst_equal_src(void **state) {
    uint8_t * set = (uint8_t*) malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * source= &set[0];
    uint8_t * dest= &set[0];
    uint8_t * status = NULL;
    //Initialize the memory to test values
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++)
        set[i]=i;
    status = my_memmove(source,dest,(MEM_SIZE_EVEN/2));
    for (uint8_t i=0; i<(MEM_SIZE_EVEN/2); i++) {
        //Check each of the 5 values against the source
        assert_int_equal(* (status+i),i);
    }
}
static void test_memcpy(void **state){
    uint8_t i;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    set = (uint8_t*) reserve_words(MEM_SET_SIZE_W);

    assert_non_null(set);
    ptra = &set[0];
    ptrb = &set[16];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++) {
        set[i] = i;
    }

    my_memcpy(ptra, ptrb, TEST_MEMMOVE_LENGTH);

    for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
    {
        assert_int_equal(set[i + 16], i);
    }

    free_words( (uint32_t*)set );
}

static void test_memset(void **state){
    uint8_t i;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    set = (uint8_t*)reserve_words(MEM_SET_SIZE_W);
    assert_non_null(set);
}

```

```

ptrA = &set[0];
ptrB = &set[16];

/* Initialize the set to test values */
for( i = 0; i < MEM_SET_SIZE_B; i++)
{
    set[i] = i;
}

my_memset(ptrA, MEM_SET_SIZE_B, 0xFF);
my_memzero(ptrB, MEM_ZERO_LENGTH);

/* Validate Set & Zero Functionality */
for (i = 0; i < MEM_ZERO_LENGTH; i++)
{
    assert_int_equal(set[i], 0xFF);
    assert_int_equal(set[i + 16], 0);
}

free_words( (uint32_t*)set );
}

static void test_reverse(void **state) {
    uint8_t i;
    uint8_t * copy;
    uint8_t set[MEM_SET_SIZE_B] = {0x3F, 0x73, 0x72, 0x33, 0x54, 0x43,
    0x72, 0x26,
                                0x48, 0x63, 0x20, 0x66, 0x6F, 0x00,
    0x20, 0x33,
                                0x72, 0x75, 0x74, 0x78, 0x21, 0x4D,
    0x20, 0x40,
                                0x20, 0x24, 0x7C, 0x20, 0x24, 0x69,
    0x68, 0x54
                                };

    copy = (uint8_t*)reserve_words(MEM_SET_SIZE_W);
    assert_non_null(copy);

    my_memcpy(set, copy, MEM_SET_SIZE_B);

    my_reverse(set, MEM_SET_SIZE_B);

    for (i = 0; i < MEM_SET_SIZE_B; i++)
    {
        assert_int_equal(set[i], copy[MEM_SET_SIZE_B - i - 1]);
    }

    free_words( (uint32_t*)copy );
}

static void test_big_to_little_valid_Ptr(void **state) {
    uint32_t * data = NULL;
    //uint32_t length =4;
    int8_t status = big_to_little(data); //check for null pointer
}

```

```

        assert_int_equal(status,1); //status = 1 , null pointer passed
    }

static void test_big_to_little_valid_Conv(void **state) {
    int8_t status;
    uint32_t data = {0x9ABCDEF0};
    //uint32_t length = sizeof(data);
    uint32_t * temp = (uint32_t *)malloc(sizeof(uint32_t)); // temp
variable to store original copy
    my_memcpy((uint8_t*)&data,(uint8_t*)temp,4); //length*4);
    status = big_to_little(&data);
    assert_int_equal(status,0);
    for(int8_t i =0; i<2;i++)
    {
        assert_int_equal(*(((uint8_t *)temp)+i),*((uint8_t *)&data)+3-i));
//checking byte wise for data conversion
    }
}

static void test_little_to_big_valid_Ptr(void **state) {
    //uint32_t length =4;
    uint32_t * data = NULL;
    int8_t status = little_to_big(data);
    assert_int_equal(status,1); // return value is 1 if null pointer is
passed
}

static void test_little_to_big_valid_Conv(void **state) {
    int8_t status;
    uint32_t data = {0x9ABCDEF0};
    //uint32_t length = sizeof(data);
    uint32_t * temp = (uint32_t *)malloc(sizeof(uint32_t)); // temp
variable to store original copy
    my_memcpy((uint8_t*)&data,(uint8_t*)temp,4); //length*4);
    status = big_to_little(&data);
    assert_int_equal(status,0);
    for(int8_t i =0; i<2;i++)
    {
        assert_int_equal(*(((uint8_t *)temp)+i),*((uint8_t *)&data)+3-i));
//checking byte wise for data conversion
    }
}

static void test_circbuf_init(void **state){
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb, 100);

    assert_int_equal(cb_status, CB_SUCCESS);
}

```

```

static void test_circbuf_is_full1(void **state) {
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb, 100);
    assert_int_equal(cb_status, CB_SUCCESS);

    assert_false(cb_is_full(&my_cb));
}

static void test_circbuf_is_full2(void **state) {
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb, 100);
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_buffer_add_item(&my_cb, 'a');
    assert_int_equal(cb_status, CB_SUCCESS);

    assert_false(cb_is_full(&my_cb));
}

static void test_circbuf_full(void **state) {
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);
    for (uint8_t i=0; i<BUFFER_LENGTH; i++)
    {
        cb_status = cb_buffer_add_item(ptr,i);
        assert_int_equal(cb_status,CB_SUCCESS);
    }
    assert_int_equal(cb_is_full(ptr),1); //check if buffer full true
}

static void test_circbuf_destroy(void **state) {
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb,100);
    assert_int_equal(cb_status,CB_SUCCESS);

    cb_status = cb_destroy(&my_cb);
    assert_int_equal(cb_status, CB_SUCCESS);
}

static void test_circbuf_empty1(void **state) {
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb,100);
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_buffer_add_item(&my_cb, 'a');
    assert_int_equal(cb_status, CB_SUCCESS);

    assert_false(cb_is_empty(&my_cb));
}

```

```

static void test_circbuf_empty2(void **state) {
    cb_struct my_cb;
    int8_t j;

    cb_enum cb_status = cb_init(&my_cb, 100);
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_buffer_add_item(&my_cb, 'a');
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_buffer_remove_item(&my_cb, &j);
    assert_int_equal(cb_status, CB_SUCCESS);

    assert_true(cb_is_empty(&my_cb));
}

static void test_circbuf_empty(void **state) {
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    cb_enum cb_status = cb_init(ptr, BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);
    assert_int_equal(cb_is_empty(ptr), 1);
}

static void test_circbuf_add_remove(void **state) {
    cb_struct my_cb;
    int8_t j;
    cb_enum cb_status = cb_init(&my_cb, BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);

    for (uint8_t i=0; i<BUFFER_LENGTH+2; i++)
    {
        cb_status = cb_buffer_add_item(&my_cb, i);
        cb_status = cb_buffer_remove_item(&my_cb, &j);
        assert_int_equal(i, (uint8_t)j);
    }

    assert_int_equal(cb_status, CB_SUCCESS);
}

static void test_circbuf_add_item(void **state) {
    cb_struct my_cb;

    cb_enum cb_status = cb_init(&my_cb, 100);
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_buffer_add_item(&my_cb, 'a');
    assert_int_equal(cb_status, CB_SUCCESS);

    cb_status = cb_is_empty(&my_cb);
    assert_false(cb_status);
}

```

```

static void test_circbuf_wrap_add(void **state) {
    //cb_struct my_cb;
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    int8_t j;
    cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);
    for (uint8_t i=0; i<BUFFER_LENGTH; i++)
    {
        cb_status = cb_buffer_add_item(ptr,i);
        assert_int_equal(cb_status,CB_SUCCESS);
    }

    cb_status = cb_buffer_remove_item(ptr, &j); //remove first item
    cb_status = cb_buffer_add_item(ptr,j); //add it again to the end for wrap
    add
    assert_ptr_equal((ptr->head), (ptr->tail)-1);
}

static void test_circbuf_wrap_remove(void **state) {
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    int8_t j;
    cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);
    for (uint8_t i=0; i<BUFFER_LENGTH; i++)
    {
        cb_status = cb_buffer_add_item(ptr,i);
        assert_int_equal(cb_status,CB_SUCCESS);
    }
    for (uint8_t i=0; i<BUFFER_LENGTH; i++)
    {
        cb_status = cb_buffer_remove_item(ptr,&j);
        assert_int_equal(cb_status,CB_SUCCESS);
    }
    assert_ptr_equal((ptr->head), (ptr->tail));
}

static void test_circbuf_overfill(void **state) {
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    //int8_t j;
    cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);
    assert_int_equal(cb_status, CB_SUCCESS);
    for (uint8_t i=0; i<BUFFER_LENGTH; i++)
    {
        cb_status = cb_buffer_add_item(ptr,i);
        assert_int_equal(cb_status,CB_SUCCESS);
    }
    cb_status = cb_buffer_add_item(ptr,(uint8_t)0 );
    assert_int_equal(cb_status, CB_FULL_ERROR);
}

static void test_circbuf_overempty(void **state) {
    cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
    int8_t j;

```

```

cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);
assert_int_equal(cb_status, CB_SUCCESS);
//remove item from empty buffer
cb_status = cb_buffer_remove_item(ptr,&j);
assert_int_equal(cb_status,CB_EMPTY_ERROR);
}

static void test_circbuf_allocate_free(void **state) {
cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
//int8_t j;
cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);//allocate
assert_int_equal(cb_status, CB_SUCCESS);
cb_status = cb_destroy(ptr);//free
assert_int_equal(cb_status, CB_SUCCESS);
}

static void test_circbuf_invalid_ptr(void **state) {
cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
//int8_t j;
cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);//allocate
assert_int_equal(cb_status, CB_SUCCESS);
assert_ptr_not_equal(ptr->buffer, NULL);
}

static void test_circbuf_initialised(void **state) {
cb_struct *ptr = (cb_struct*)malloc(sizeof(cb_struct));
int8_t j;
cb_enum cb_status = cb_init(ptr,BUFFER_LENGTH);//allocate
assert_int_equal(cb_status, CB_SUCCESS);
for(j=0;j<BUFFER_LENGTH;j++)
assert_int_equal(*((ptr->buffer)+j),0);
}

static void test_reverse_null_Ptr(void **state) {
uint8_t * array= NULL;
uint8_t * status = NULL;
status = my_reverse(array,MEM_SIZE_EVEN);
assert_ptr_equal(cast_ptr_to_largest_integral_type(status),NULL);
}

static void test_reverse_odd(void **state) {
uint8_t array[MEM_SIZE_ODD]= {10,11,12,13,14,15,16,17,18};
uint8_t *temp =(uint8_t *)malloc(MEM_SIZE_ODD*sizeof(uint8_t));
temp = my_memcpy(array,temp, MEM_SIZE_ODD);
uint8_t * status = NULL;
status = my_reverse(array,MEM_SIZE_ODD);
for (uint8_t i=0;i<MEM_SIZE_ODD;i++)
{
    assert_int_equal(temp[MEM_SIZE_ODD-1-i],status[i]);
}
}

static void test_reverse_even(void **state) {
uint8_t array[MEM_SIZE_EVEN]= {10,11,12,13,14,15,16,17};
uint8_t *temp =(uint8_t *)malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
}

```

```

        temp = my_memcpy(array,temp,MEM_SIZE_EVEN);
        uint8_t * status = NULL;
        status = my_reverse(array, MEM_SIZE_EVEN);
        for (uint8_t i=0;i<MEM_SIZE_EVEN;i++)
        {
            assert_int_equal(temp[MEM_SIZE_EVEN-1-i],status[i]);
        }
    }

static void test_reverse_char(void **state) {
    char a[MEM_SIZE_CHAR];
    for (uint16_t i=0; i<MEM_SIZE_CHAR; i++)
        a[i]=(char)i;
    uint8_t * status = NULL;
    status = my_reverse((uint8_t *)a, MEM_SIZE_CHAR);
    for (uint16_t i=0;i<MEM_SIZE_CHAR;i++)
        assert_int_equal(MEM_SIZE_CHAR-i-1,status[i]);
}

static void test_memzero_null_Ptr(void **state) {
    uint8_t * set = NULL;
    uint8_t * status = NULL;
    status = my_memzero(set, MEM_SIZE_EVEN);
    assert_ptr_equal(cast_ptr_to_largest_integral_type(status),NULL);
}

static void test_memzero_check_set(void **state) {
    uint8_t * set = (uint8_t *)malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * status = NULL;
    status = my_memzero(set, MEM_SIZE_EVEN);
    for (uint8_t i=0; i<MEM_SIZE_EVEN; i++)
        assert_int_equal(0,* (status+i));
}

static void test_memset_null_Ptr(void **state) {
    uint8_t * set = NULL;
    uint8_t * status = NULL;
    uint8_t data = 10;
    status = my_memset(set, MEM_SIZE_EVEN,data);
    assert_ptr_equal(cast_ptr_to_largest_integral_type(status),NULL);
}
static void test_memset_check_set(void **state) {
    uint8_t * set = (uint8_t *)malloc(MEM_SIZE_EVEN*sizeof(uint8_t));
    uint8_t * status = NULL;
    uint8_t data = 10;
    status = my_memset(set, MEM_SIZE_EVEN,data);
    for (uint8_t i=0; i<MEM_SIZE_EVEN; i++)
        assert_int_equal(data,* (status+i));
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_data1),
        cmocka_unit_test(test_data2),

```

```

        cmocka_unit_test(test_itoa_max_sized_integer),
        cmocka_unit_test(test_atoi_max_sized_integer),
        cmocka_unit_test(test_memmove1),
        cmocka_unit_test(test_memmove2),
        cmocka_unit_test(test_memmove3),
        cmocka_unit_test(test_memmove_null_Ptr),
        cmocka_unit_test(test_memmove_no_overlap),
        cmocka_unit_test(test_memmove_src_in_dst),
        cmocka_unit_test(test_memmove_dst_in_src),
        cmocka_unit_test(test_memmove_dst_equal_src),
        cmocka_unit_test(test_memzero_null_Ptr),
        cmocka_unit_test(test_memzero_check_set),
        cmocka_unit_test(test_memcpy),
        cmocka_unit_test(test_memset),
        cmocka_unit_test(test_memset_null_Ptr),
        cmocka_unit_test(test_memset_check_set),
        cmocka_unit_test(test_reverse),
        cmocka_unit_test(test_reverse_null_Ptr),
        cmocka_unit_test(test_reverse_odd),
        cmocka_unit_test(test_reverse_even),
        cmocka_unit_test(test_reverse_char),
        cmocka_unit_test(test_big_to_little_valid_Ptr),
        cmocka_unit_test(test_big_to_little_valid_Conv),
        cmocka_unit_test(test_little_to_big_valid_Ptr),
        cmocka_unit_test(test_little_to_big_valid_Conv),
        cmocka_unit_test(test_circbuf_init),
        cmocka_unit_test(test_circbuf_is_full1),
        cmocka_unit_test(test_circbuf_is_full2),
        cmocka_unit_test(test_circbuf_destroy),
        cmocka_unit_test(test_circbuf_empty1),
        cmocka_unit_test(test_circbuf_empty2),
        cmocka_unit_test(test_circbuf_add_remove),
        cmocka_unit_test(test_circbuf_add_item),
        cmocka_unit_test(test_circbuf_wrap_add),
        cmocka_unit_test(test_circbuf_wrap_remove),
        cmocka_unit_test(test_circbuf_overfill),
        cmocka_unit_test(test_circbuf_overempty),
        cmocka_unit_test(test_circbuf_allocate_free),
        cmocka_unit_test(test_circbuf_invalid_ptr),
        cmocka_unit_test(test_circbuf_full),
        cmocka_unit_test(test_circbuf_initialised),
        cmocka_unit_test(test_circbuf_empty),
        cmocka_unit_test(test_itoa_invalid_ptr),
        cmocka_unit_test(test_atoi_invalid_ptr),
        cmocka_unit_test(test_itoa_zero_integer),
        cmocka_unit_test(test_atoi_zero_integer)

    };

    return cmocka_run_group_tests(tests, NULL, NULL);
}
/***
 * @file debug.c

```

```

* @brief source files for debug.c
* @author Miles Frain
* @author Shreya Chakraborty
* @version 1
* @date 2018-02-10
*/
#ifndef __DEBUG_H__
#define __DEBUG_H__

#include <stdint.h>
#include "debug.h"
#include "platform.h"

void print_array(uint8_t* start, uint32_t length)
{
#ifdef VERBOSE
    int i = 0;
    if (start)
    {
        while(length != 0)
        {
            PRINTF("Address: %p and Data: %d\n ", (start+i),*(start+i)) ;
            i++; length--;
        }
    }
#endif
}
#endif // __DEBUG_H__
/** 
 * @file port.c
 * @brief GPIO management
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-03-04
*/
#include "port.h"
#include "MKL25Z4.h"

//Configures the RGB LEDs to be output with their initial values.
void GPIO_Configure()
{
    SIM_SCGC5 |= 0x00000400; //enable clock gating control for PORT B
    SIM_SCGC5 |= 0x00001000; //enable clock gating control for PORT D
    PTB->PDDR |= 0x00040000; //port B pin 18 RGB led red, set direction
as output
    PTB->PDDR |= 0x00080000; //port B pin 19 RGB led green, set direction
as output
    //PTD->PDDR |= 0x00000002; //port D pin 1 RGB led blue, set direction
as output
    SIM_SCGC5 |= SIM_SCGC5_PORTB(1);
    PORTB->PCR[18] |= (1 << 8);
}

```

```

PORTB->PCR[19] |= (1 << 8);
//PORTD->PCR[1] |= (1 << 8);
RGB_RED_OFF();
RGB_GREEN_OFF();
//RGB_BLUE_OFF();
}
*****
* Copyright (C) 2017 by Alex Fosdick - University of Colorado
*
* Redistribution, modification or use of this software in source or
binary
* forms is permitted as long as the files maintain this copyright. Users
are
* permitted to modify this and use it to learn about the field of
embedded
* software. Alex Fosdick and the University of Colorado are not liable
for any
* misuse of this material.
*

*****
*** /
/***
* @file project1_test.c
* @brief This file is to be used to project 1.
*
* @author Alex Fosdick
* @date April 2, 2017
*
*/
#include <stdio.h>
#include <stdint.h>
#include "platform.h"
#include "project1.h"
#include "memory.h"
#include "conversion.h"
#include "debug.h"

#define BASE_16 (16)
#define BASE_10 (10)

int8_t test_data1() {
    uint8_t * ptr;
    int32_t num = -4096;
    uint32_t digits;
    int32_t value;

    PRINTF("\ntest_data1();\n");
    ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

    if (! ptr )
    {

```

```

        return TEST_ERROR;
    }

digits = my_itoa( num, ptr, BASE_16);
value = my_atoi( ptr, digits, BASE_16);
#ifndef VERBOSE
PRINTF("  Initial number: %ld\n", num);
PRINTF("  Final Decimal number: %ld\n", value);
#endif
free_words( (uint32_t*)ptr );

if ( value != num )
{
    return TEST_ERROR;
}
return TEST_NO_ERROR;
}

int8_t test_data2() {
    uint8_t *ptr;
    int32_t num = 123456;
    uint32_t digits;
    int32_t value;

PRINTF("test_data2()\n");
ptr = (uint8_t*) reserve_words( DATA_SET_SIZE_W );

if (! ptr )
{
    return TEST_ERROR;
}

digits = my_itoa( num, ptr, BASE_10);
value = my_atoi( ptr, digits, BASE_10);
#ifndef VERBOSE
PRINTF("  Initial Decimal number: %ld\n", num);
PRINTF("  Final Decimal number: %ld\n", value);
#endif
free_words( (uint32_t*)ptr );

if ( value != num )
{
    return TEST_ERROR;
}
return TEST_NO_ERROR;
}

int8_t test_memmove1() {
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t *set;
    uint8_t *ptrA;
    uint8_t *ptrB;

```

```

PRINTF("test_memmove1() - NO OVERLAP\n");
set = (uint8_t*) reserve_words( MEM_SET_SIZE_W );

if (! set )
{
    return TEST_ERROR;
}

ptr_a = &set[0];
ptr_b = &set[16];

/* Initialize the set to test values */
for( i = 0; i < MEM_SET_SIZE_B; i++)
{
    set[i] = i;
}

print_array(set, MEM_SET_SIZE_B);
my_memmove(ptr_a, ptr_b, TEST_MEMMOVE_LENGTH);
print_array(set, MEM_SET_SIZE_B);

for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
{
    if (set[i + 16] != i)
    {
        ret = TEST_ERROR;
    }
}

free_words( (uint32_t*)set );
return ret;
}

int8_t test_memmove2() {
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t * set;
    uint8_t * ptr_a;
    uint8_t * ptr_b;

PRINTF("test_memmove2() -OVERLAP END OF SRC BEGINNING OF DST\n");
set = (uint8_t*) reserve_words(MEM_SET_SIZE_W);

if (! set )
{
    return TEST_ERROR;
}
ptr_a = &set[0];
ptr_b = &set[8];

/* Initialize the set to test values */
for( i = 0; i < MEM_SET_SIZE_B; i++) {
    set[i] = i;
}
}

```

```

print_array(set, MEM_SET_SIZE_B);
my_memmove(ptra, ptrb, TEST_MEMMOVE_LENGTH);
print_array(set, MEM_SET_SIZE_B);

for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
{
    if (set[i + 8] != i)
    {
        ret = TEST_ERROR;
    }
}

free_words( (uint32_t*)set );
return ret;
}

int8_t test_memmove3() {
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

PRINTF("test_memmove3() - OVERLAP END OF DEST BEGINNING OF SRC\n");
set = (uint8_t*)reserve_words( MEM_SET_SIZE_W );

if (!set)
{
    return TEST_ERROR;
}
ptra = &set[8];
ptrb = &set[0];

/* Initialize the set to test values */
for( i = 0; i < MEM_SET_SIZE_B; i++)
{
    set[i] = i;
}

print_array(set, MEM_SET_SIZE_B);
my_memmove(ptra, ptrb, TEST_MEMMOVE_LENGTH);
print_array(set, MEM_SET_SIZE_B);

for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
{
    if (set[i] != (i + 8))
    {
        ret = TEST_ERROR;
    }
}

free_words( (uint32_t*)set );

```

```

    return ret;
}

int8_t test_memcpy() {
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    PRINTF("test_memcpy()\n");
    set = (uint8_t*) reserve_words(MEM_SET_SIZE_W);

    if (!set)
    {
        return TEST_ERROR;
    }
    ptra = &set[0];
    ptrb = &set[16];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++) {
        set[i] = i;
    }

    print_array(set, MEM_SET_SIZE_B);
    my_memcpy(ptra, ptrb, TEST_MEMMOVE_LENGTH);
    print_array(set, MEM_SET_SIZE_B);

    for (i = 0; i < TEST_MEMMOVE_LENGTH; i++)
    {
        if (set[i+16] != i)
        {
            ret = TEST_ERROR;
        }
    }

    free_words( (uint32_t*)set );
    return ret;
}

int8_t test_memset()
{
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t * set;
    uint8_t * ptra;
    uint8_t * ptrb;

    PRINTF("test_memset()\n");
    set = (uint8_t*) reserve_words(MEM_SET_SIZE_W);
    if (!set)
    {

```

```

        return TEST_ERROR;
    }
    ptra = &set[0];
    ptrb = &set[16];

    /* Initialize the set to test values */
    for( i = 0; i < MEM_SET_SIZE_B; i++)
    {
        set[i] = i;
    }

    print_array(set, MEM_SET_SIZE_B);
    my_memset(ptra, MEM_SET_SIZE_B, 0xFF);
    print_array(set, MEM_SET_SIZE_B);
    my_memzero(ptrb, MEM_ZERO_LENGTH);
    print_array(set, MEM_SET_SIZE_B);

    /* Validate Set & Zero Functionality */
    for (i = 0; i < MEM_ZERO_LENGTH; i++)
    {
        if (set[i] != 0xFF)
        {
            ret = TEST_ERROR;
        }
        if (set[16 + i] != 0)
        {
            ret = TEST_ERROR;
        }
    }

    free_words( (uint32_t*)set );
    return ret;
}

int8_t test_reverse()
{
    uint8_t i;
    int8_t ret = TEST_NO_ERROR;
    uint8_t * copy;
    uint8_t set[MEM_SET_SIZE_B] = {0x3F, 0x73, 0x72, 0x33, 0x54, 0x43,
    0x72, 0x26,
                                0x48, 0x63, 0x20, 0x66, 0x6F, 0x00,
    0x20, 0x33,
                                0x72, 0x75, 0x74, 0x78, 0x21, 0x4D,
    0x20, 0x40,
                                0x20, 0x24, 0x7C, 0x20, 0x24, 0x69,
    0x68, 0x54
                                };

    PRINTF("test_reverse()\n");
    copy = (uint8_t*)reserve_words(MEM_SET_SIZE_W);
    if (!copy )
    {
        return TEST_ERROR;
    }
}

```

```

}

my_memcpy(set, copy, MEM_SET_SIZE_B);

print_array(set, MEM_SET_SIZE_B);
my_reverse(set, MEM_SET_SIZE_B);
print_array(set, MEM_SET_SIZE_B);

for (i = 0; i < MEM_SET_SIZE_B; i++)
{
    if (set[i] != copy[MEM_SET_SIZE_B - i - 1])
    {
        ret = TEST_ERROR;
    }
}

free_words( (uint32_t*)copy );
return ret;
}

void project1(void)
{
    uint8_t i;
    int8_t failed = 0;
    int8_t results[TESTCOUNT];

    results[0] = test_data1();
    results[1] = test_data2();
    results[2] = test_memmove1();
    results[3] = test_memmove2();
    results[4] = test_memmove3();
    results[5] = test_memcpy();
    results[6] = test_memset();
    results[7] = test_reverse();

    for (i = 0; i < TESTCOUNT; i++)
    {
        failed += results[i];
    }

    PRINTF("-----\n");
    PRINTF("Test Results:\n");
    PRINTF("  PASSED: %d / %d\n", (TESTCOUNT - failed), TESTCOUNT);
    PRINTF("  FAILED: %d / %d\n", failed, TESTCOUNT);
    PRINTF("-----\n");
}

/***
 * @file main.c
 * @brief Main program entry point
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-02-10
 */

```

```

#include "project1.h"
#include "profiler.h"
#include "data.h"
#include "platform.h"
#include "data_processing.h"
#include "circbuf.h"
#include <unistd.h>
#ifndef KL25Z
#include "uart.h"
#include "clock.h"
#include "port.h"
#include "profiler.h"
#include "nordic.h"
// Not sure exactly what state SystemInit() leaves the device in with
CLOCK_SETUP
// TODO - check register differences
#define CLOCK_SETUP 1
#endif

int main()
{
#if defined PROJECT2 || defined PROJECT3
#ifndef KL25Z
    clock_setup();
    GPIO_Configure();
    UART_configure(BAUD_115200);
    systick_init();
    SPI_init();
#else // platform not KL25Z
    cb_struct *rx_buffer; // defined by uart otherwise
#endif // platform

#ifndef 0
    rx_buffer = malloc(sizeof(cb_struct));
    cb_enum status = cb_init(rx_buffer, 256);
    if (status != CB_SUCCESS)
    {
        PRINTF("Error initializing rx buffer\n");
    }
    PRINTF("\nStarting Project 2 or 3\n");
#endif
#endif

#ifndef PROJECT3
    PRINTF("\nProject 3 print\n");
#endif
#ifndef KL25Z
    k125z_profile_option(MEMMOVE_DMA);
    k125z_profile_option(MEMSET_DMA);
    k125z_profile_option(MY_MEMMOVE);
    k125z_profile_option(MY_MEMSET);
    k125z_profile_option(MEMMOVE);
    k125z_profile_option(MEMSET);
#endif

    nordic_test();
}

```

```

#else
    bbb_profile_option(1);
    bbb_profile_option(2);
    bbb_profile_option(3);
    bbb_profile_option(4);
#endif // platform
#endif // project 3

#ifdef PROJECT2
    print_data_process_header();

#ifdef KL25Z
    while (1)
    {
        if (!cb_is_empty(rx_buffer))
        {
            data_process(rx_buffer);

            RGB_RED_TOGGLE();
        }
    }
#else // HOST or BBB
    char string[2000];
    while (1)
    {
        printf("\nEnter String: ");
        scanf("%s", string);
        for (int j = 0; string[j] != '\0'; j++)
        {
            cb_buffer_add_item(rx_buffer, string[j]);
        }
        data_process(rx_buffer);
    }
#endif // platform
#endif // PROJECT2

#endif // project 2 or 3

#ifdef PROJECT1
    project1();
    print_cstd_type_sizes();
    print_stdint_type_sizes();
    print_pointer_sizes();
    uint32_t i = 0x12345678;
    PRINTF("pre swap 0x%lx\n", i);
    determine_endianness();
    swap_data_endianness((uint8_t *)&i, sizeof(i));
    PRINTF("post swap 0x%lx\n", i);
#endif

    return 0;
}
/***

```

```

* @file clock.c
* @brief Clock setup functions
* @author Miles Frain
* @author Shreya Chakraborty
* @version 1
* @date 2018-03-04
*/
#include "MKL25Z4.h"
#include "MKL25Z4_extension.h"

void clock_setup()
{
    SIM_BWR_SOPT2_PLLFLLSEL(SIM, 1); // MCGPLLCLK clock with fixed divide
by two

    SIM_BWR_CLKDIV1_OUTDIV1(SIM, 1); // Divide by 2: core, platform, and
system clock
    SIM_BWR_CLKDIV1_OUTDIV4(SIM, 3); // Divide by 8 (4 here * 2 above):
bus and flash clock

    MCG_BWR_C1_IRCLKEN(MCG, 1); // MCGIRCLK internal reference clock is
enabled
    MCG_BWR_C1_IREFS(MCG, 0); // Not selecting slow internal clock
reference. Selecting external reference clock instead. Defaults to 1
(slow internal)
    MCG_BWR_C1_FRDIV(MCG, 3); // FLL external reference divider. Divide
factor of 256 (C2->RANGE = 2: very high frequency)

    MCG_BWR_C2_EREFS0(MCG, 1); // External reference select, select
Oscillator instead of external reference clock (naming logic seems
inverted)
    MCG_BWR_C2_RANGE0(MCG, 2); // very high frequency range

    MCG_BWR_C5_PRDIV0(MCG, 1); // Set PLL external refenece divider
factor to 9. Should probably set this before enabling PLL

    MCG_BWR_C6_PLLS(MCG, 1); // PLL selected. Make sure to set PRDIV0
correctly first.

    OSC_BWR_CR_ERCLKEN(OSC0, 1); // Enable external reference clock
}
/***
* @file data.c
* @brief source files for data.c
* @author Miles Frain
* @author Shreya Chakraborty
* @version 1
* @date 2018-02-10
*/
#endif defined (__GNUC__)
#pragma GCC diagnostic ignored "-Wunused-but-set-variable"
#endif

```

```

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <stddef.h>
#include "memory.h"
#include "platform.h"

#define LITTLE_ENDIANN (0)
#define BIG_ENDIANN (1)
#define SWAP_NO_ERROR (0)
#define SWAP_ERROR (-1)

/*reports on the type sizes of cstd*/
void print_cstd_type_sizes()
{
    int a;
    char b;
    short c;
    long d;
    double e;
    float f;
    unsigned char g;
    unsigned int h;
    unsigned long i;
    signed char j;
    signed int k;
    signed long l;

    size_t temp = 0; //stores size of each variable

    temp = sizeof(a);
    PRINTF("size of the integer a is %zu bytes\n", temp);
    temp = sizeof(b);
    PRINTF("size of the character b is %zu bytes\n", temp);
    temp = sizeof(c);
    PRINTF("size of the short c is %zu bytes\n", temp);
    temp = sizeof(d);
    PRINTF("size of the long d is %zu bytes\n", temp);
    temp = sizeof(e);
    PRINTF("size of the double e is %zu bytes\n", temp);
    temp = sizeof(f);
    PRINTF("size of the float f is %zu bytes\n", temp);
    temp = sizeof(g);
    PRINTF("size of the unsigned character g is %zu bytes\n", temp);
    temp = sizeof(h);
    PRINTF("size of the unsigned integer h is %zu bytes\n", temp);
    temp = sizeof(i);
    PRINTF("size of the unsigned long i is %zu bytes\n", temp);
    temp = sizeof(j);
    PRINTF("size of the signed character j is %zu bytes\n", temp);
    temp = sizeof(k);
    PRINTF("size of the signed integer k is %zu bytes\n", temp);
    temp = sizeof(l);
}

```

```

        PRINTF("size of the signed long l is %zu bytes\n", temp);
    }

/*reports on the type sizes of stdint*/
void print_stdint_type_sizes()
{
    int8_t a;
    uint8_t b;
    int16_t c;
    uint16_t d;
    int32_t e;
    uint32_t f;
    uint_fast8_t g;
    uint_fast16_t h;
    uint_fast32_t i;
    uint_least8_t j;
    uint_least16_t k;
    uint_least32_t l;
    size_t m;
    ptrdiff_t n;

    size_t temp = 0; //stores size of each variable

    temp = sizeof(a);
    PRINTF("size of the int8_t a is %zu bytes\n", temp);
    temp = sizeof(b);
    PRINTF("size of the uint8_t b is %zu bytes\n", temp);
    temp = sizeof(c);
    PRINTF("size of the int16_t c is %zu bytes\n", temp);
    temp = sizeof(d);
    PRINTF("size of the uint16_t d is %zu bytes\n", temp);
    temp = sizeof(e);
    PRINTF("size of the int32_t e is %zu bytes\n", temp);
    temp = sizeof(f);
    PRINTF("size of the uint32_t f is %zu bytes\n", temp);
    temp = sizeof(g);
    PRINTF("size of the uint_fast8_t g is %zu bytes\n", temp);
    temp = sizeof(h);
    PRINTF("size of the uint_fast16_t h is %zu bytes\n", temp);
    temp = sizeof(i);
    PRINTF("size of the uint_fast32_t i is %zu bytes\n", temp);
    temp = sizeof(j);
    PRINTF("size of the uint_least8_t j is %zu bytes\n", temp);
    temp = sizeof(k);
    PRINTF("size of the uint_least16_t k is %zu bytes\n", temp);
    temp = sizeof(l);
    PRINTF("size of the uint_least32_t l is %zu bytes\n", temp);
    temp = sizeof(m);
    PRINTF("size of the size_t m is %zu bytes\n", temp);
    temp = sizeof(n);
    PRINTF("size of the ptrdiff_t n is %zu bytes\n", temp);
}

/*reports on the type sizes of pointers*/

```

```

void print_pointer_sizes()
{
    char *a;
    short *b;
    int *c;
    long *d;
    double *e;
    float *f;
    void *g;
    int8_t *h;
    int16_t *i;
    int32_t *j;
    char **k;
    int **l;
    void **m;

    size_t temp = 0; //stores size of each variable

    temp = sizeof(a);
    PRINTF("size of the char ptr a is %zu bytes\n", temp);
    temp = sizeof(b);
    PRINTF("size of the short ptr b is %zu bytes\n", temp);
    temp = sizeof(c);
    PRINTF("size of the int ptr c is %zu bytes\n", temp);
    temp = sizeof(d);
    PRINTF("size of the long ptr d is %zu bytes\n", temp);
    temp = sizeof(e);
    PRINTF("size of the double ptr e is %zu bytes\n", temp);
    temp = sizeof(f);
    PRINTF("size of the float ptr f is %zu bytes\n", temp);
    temp = sizeof(g);
    PRINTF("size of the void ptr g is %zu bytes\n", temp);
    temp = sizeof(h);
    PRINTF("size of the int8_t ptr h is %zu bytes\n", temp);
    temp = sizeof(i);
    PRINTF("size of the int16_t ptr i is %zu bytes\n", temp);
    temp = sizeof(j);
    PRINTF("size of the int32_t ptr j is %zu bytes\n", temp);
    temp = sizeof(k);
    PRINTF("size of the char double pointer k is %zu bytes\n", temp);
    temp = sizeof(l);
    PRINTF("size of the int double pointer l is %zu bytes\n", temp);
    temp = sizeof(m);
    PRINTF("size of the void double pointer m is %zu bytes\n", temp);
}

/* function to determine the endianness */
uint32_t determine_endianness()
{
    unsigned int x = 0x01234567;
    char *c = (char*) &x;
    PRINTF ("*c is: 0x%x\n", *c);
    if (*c == 0x67)
    {
        PRINTF("little endian. \n");
    }
}

```

```

        return LITTLE_ENDIAN;
    }
else
{
    PRINTF("big endian. \n");
    return BIG_ENDIAN;
}
}

/*function to swap endianess*/
int32_t swap_data_endianness(uint8_t * data, size_t type_length)
{
    if(my_reverse(data, type_length)) //changing the order of bytes
    {
        return SWAP_NO_ERROR;
    }
else
{
    return SWAP_ERROR;
}
}

/*function to change from little to big*/
int8_t little_to_big(uint32_t * data)
{
    int8_t *a= NULL;
    int8_t result = 1;
    a = (int8_t*)my_reverse((uint8_t*)data, 4);
    if(a)
    {
        result = 0;
    }
    return result;
}

/*function to change from big to little*/
int8_t big_to_little(uint32_t * data)
{
    int8_t *a= NULL;
    int8_t result = 1;
    a = (int8_t*)my_reverse((uint8_t*)data, 4);
    if(a)
    {
        result = 0;
    }
    return result;
}

/**
 * @file circbuf.c
 * @brief Circular buffer functions
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1

```

```

* @date 2018-03-04
*/
#include <circbuf.h>
#include <stdio.h>
#include <stdint.h>
#include "memory.h"
#include "platform.h"

/*initialisation and allocation of dynamic memory*/
cb_enum cb_init(cb_struct *ptr, size_t length)
{
    cb_enum status;
    if (ptr == NULL) //check for null and negative length
    {
        status = CB_NULL_ERROR;
    }
    else if (length <= 0)
    {
        status = CB_LENGTH_ERROR;
    }
    else
    {
        ptr->buffer = (int8_t *)malloc(length * sizeof(int8_t)); // allocates the length no of items for the buffer memory
        if(ptr->buffer == NULL)
            //reinitialise size to 0, current item count in the buffer to 0, head and tail to null
            ptr->size = 0;
            ptr->head = NULL;
            ptr->tail = NULL;
            ptr->count = 0;
            status = CB_ALLOCATION_FAILURE;
        }
        else
        { //initialise size to given length, current item count in the buffer to 0, head and tail to null
            ptr->size = length;
            ptr->head = ptr->buffer;
            ptr->tail = ptr->buffer;
            ptr->count = 0;
            my_memzero((uint8_t*)ptr->buffer, length);
            status = CB_SUCCESS;
        }
    }
    return status;
}

/*The functions takes in a pointer to the circular buffer to be destroyed by deallocating using free*/
cb_enum cb_destroy(cb_struct *ptr)
{
    BEGIN_CRITICAL;

```

```

cb_enum status;
if (ptr == NULL) //check for null pointer
{
    status = CB_NULL_ERROR;
}
else
{
    if (ptr ->buffer != NULL) //check if pointer not null
    {
        free(ptr->buffer); //deallocate the memory
        ptr->buffer = NULL; //make pointer null
    }
    ptr->size = 0;
    ptr->head = NULL;
    ptr->tail = NULL;
    ptr->count = 0;
    status = CB_SUCCESS;
}
END_CRITICAL;
return status;

}

/*function to add items to the buffer*/
cb_enum cb_buffer_add_item(cb_struct *ptr, int8_t data_add)
{
BEGIN_CRITICAL;
cb_enum status = CB_SUCCESS;
if (ptr == NULL || ptr->head == NULL || ptr->tail == NULL || ptr-
>buffer == NULL) //check for null pointer
{
    status = CB_NULL_ERROR;
}
else if (cb_is_full(ptr) == 1)
{
    status = CB_FULL_ERROR;
}
else if(ptr->head == ptr->buffer + ptr->size -1 ) //wrap around
situation, head to wrap around to base when at the end
{
    *(ptr->head) = data_add;
    ptr->head = ptr->buffer;
    ptr->count++;
    status = CB_SUCCESS;
}
else
{// buffer has empty spaces, hence increment the head to the next
buffer location
    *ptr->head = data_add;
    ptr->head++;
    ptr->count++;
    status = CB_SUCCESS;
}
END_CRITICAL;
}

```

```

    return status;
}

/*function to remove items from buffer*/
cb_enum cb_buffer_remove_item(cb_struct *ptr, int8_t *data_remove)
{
    BEGIN_CRITICAL;
    cb_enum status = CB_SUCCESS;
    if (ptr == NULL || ptr->head == NULL || ptr->tail == NULL || ptr-
>buffer == NULL) //check for null pointer
    {
        status = CB_NULL_ERROR;
    }

    if (cb_is_empty(ptr))
    {
        status = CB_EMPTY_ERROR;
    }
    else
    {
        if(ptr->tail == ptr->buffer + ptr->size -1) //wrap around
situation, tail to wrap around to base when at the end
        {
            *data_remove = *(ptr->tail);
            *(ptr->tail) = 0;
            ptr->tail = ptr-> buffer;
            ptr->count--;
            status = CB_SUCCESS;
        }
        else
        {// buffer has full spaces, hence increment the tail to the next
buffer location
            *data_remove = *(ptr->tail);
            *(ptr->tail) = 0;
            ptr->tail++;
            ptr->count--;
            status = CB_SUCCESS;
        }
    }
    END_CRITICAL;
    return status;
}

/*function to peek into buffer*/
cb_enum cb_peek(cb_struct *ptr, int8_t position, int8_t *data)
{
    cb_enum status = CB_SUCCESS;
    if (ptr == NULL || ptr->head == NULL || ptr->tail == NULL || ptr -
>buffer == NULL || data == NULL) //check for null pointer
    {
        status = CB_NULL_ERROR;
    }
    else if (status == CB_SUCCESS || status == CB_EMPTY_ERROR)

```

```

    {
        status = CB_EMPTY_ERROR;
    }
    else if (position <= 0 || position > ptr->size)
    {
        status = CB_POSITION_ERROR;
    }

    else
    {
        *data = *(ptr->buffer + position );
        status = CB_SUCCESS;
    }
    return status;
}

#include "nordic.h"
#include "platform.h"

/*Read the register and return the value*/
uint8_t nrf_read_register(uint8_t reg)
{
    uint8_t value;
    nrf_chip_enable();
    SPI_write_byte(reg | READ_INST);
    SPI_read_byte(&value); // STATUS returned for first byte
    SPI_write_byte(nrf_NOP); // Write any value to read next byte
    SPI_read_byte(&value); // Value of register we are interested in
    nrf_chip_disable();
    return value;
}

/*Write to the given register with the data.*/
void nrf_write_register(uint8_t reg, uint8_t value)
{
    uint8_t dummy;
    nrf_chip_enable();
    SPI_write_byte(reg | WRITE_INST);
    SPI_read_byte(&dummy); // STATUS returned
    SPI_write_byte(value);
    SPI_read_byte(&dummy); // Need to wait until byte is written before
disabling csn. Easiest way to do this is to read dummy value.
    nrf_chip_disable();
}
/*Reads the STATUS register*/
uint8_t nrf_read_status()
{
    // This could be optimized by just sending NOP command, which returns
status.
    // Otherwise status returned twice for traditional read.
    return nrf_read_register(nrf_STATUS);
}
/*Write to CONFIG register*/
void nrf_write_config(uint8_t config)

```

```

{
    nrf_write_register(nrf_CONFIG, config);
}
/*Read the CONFIG register*/
uint8_t nrf_read_config()
{
    return nrf_read_register(nrf_CONFIG);
}

/* Reads RF_SETUP register*/
uint8_t nrf_read_rf_setup()
{
    return nrf_read_register(nrf_RF_SETUP);
}
/*Writes to the RF_SETUP register*/
void nrf_write_rf_setup(uint8_t config)
{
    nrf_write_register(nrf_RF_SETUP, config);
}

/* Reads RF_CH register*/
uint8_t nrf_read_rf_ch()
{
    return nrf_read_register(nrf_RF_chregister);
}

/*Writes to the RF_CH register*/
void nrf_write_rf_ch(uint8_t channel)
{
    nrf_write_register(nrf_RF_chregister, channel);
}

/*Reads the 5 bytes of the TX_ADDR register*/
void nrf_read_TX_ADDR(uint8_t *address)
{
    nrf_chip_enable();
    SPI_write_byte(nrf_TX_ADDR | READ_INST);
    SPI_read_byte(address); // Read status byte returned from command
    for (int i = 0; i < 5; i++) // Reads the 5 bytes of the TX_ADDR
register
    {
        SPI_write_byte(0xAA); // Must write a byte to receive a byte, but
this written byte can be anything.
        SPI_read_byte(address);
        address++;
    }
    nrf_chip_disable();
}

/* Writes the 5 byte TX_ADDR register*/
void nrf_write_TX_ADDR(uint8_t *tx_addr)
{
    if (!tx_addr)
    {

```

```

        return;
    }
    uint8_t dummy;
    nrf_chip_enable();
    SPI_write_byte(nrf_TX_ADDR | WRITE_INST);
    SPI_read_byte(&dummy);
    for (int i = 0; i < 5; i++) // Writes to the 5 byte TX_ADDR register
    {
        SPI_write_byte(*tx_addr);
        tx_addr++;
        SPI_read_byte(&dummy); // Need to wait until byte is written
        before disabling csn. Easiest way to do this is to read dummy value.
    }
    nrf_chip_disable();
}
/* Reads FIFO_STATUS register*/
uint8_t nrf_read_fifo_status()
{
    return nrf_read_register(nrf_FIFO_STATUS);
}

/*Sends the command FLUSH_TX*/
void nrf_flush_tx_fifo()
{
    uint8_t dummy;
    nrf_chip_enable();
    SPI_write_byte(nrf_TXFIFO_FLUSH_CMD);
    SPI_read_byte(&dummy); // Need to wait until byte is written before
    disabling csn. Easiest way to do this is to read dummy value.
    nrf_chip_disable();
}

/*Sends the command FLUSH_RX*/
void nrf_flush_rx_fifo()
{
    uint8_t dummy;
    nrf_chip_enable();
    SPI_write_byte(nrf_RXFIFO_FLUSH_CMD);
    SPI_read_byte(&dummy); // Need to wait until byte is written before
    disabling csn. Easiest way to do this is to read dummy value.
    nrf_chip_disable();
}

void nordic_test()
{
    uint8_t spi_data[5];
    uint8_t spi_value;

    /*Reads the STATUS register*/
    spi_value = nrf_read_status();

    if (spi_value == 0x0E)
    {
        PRINTF("SPI status okay\n");
    }
}
```

```

    }
else
{
    PRINTF("SPI status ERROR\n");
}

/*Write to CONFIG register*/
nrf_write_config(0x0A);

/*Read the CONFIG register*/
spi_value = nrf_read_config();

if (spi_value == 0x0A)
{
    PRINTF("SPI config okay\n");
}
else
{
    PRINTF("SPI config ERROR\n");
}

/*Writes to the RF_SETUP register*/
nrf_write_rf_setup(0x0B);

/* Reads RF_SETUP register*/
spi_value = nrf_read_rf_setup();

if (spi_value == 0x0B)
{
    PRINTF("SPI RF_SETUP okay\n");
}
else
{
    PRINTF("SPI RF_SETUP ERROR\n");
}

/*Writes to the RF_CH register*/
nrf_write_rf_ch(0x1B);

/* Reads RF_CH register*/
spi_value = nrf_read_rf_ch();

if (spi_value == 0x1B)
{
    PRINTF("SPI RF_CH okay\n");
}
else
{
    PRINTF("SPI RF_CH ERROR\n");
}

for (int i = 0; i < 5; i++)
{
    spi_data[i] = i;
}

```

```

}

/* Writes the 5 byte TX_ADDR register*/
nrf_write_TX_ADDR(spi_data);

/*Reads the 5 bytes of the TX_ADDR register*/
nrf_read_TX_ADDR(spi_data);

for (int i = 0; i < 5; i++)
{
    if (spi_data[i] == i)
    {
        PRINTF("SPI TX_ADDR[%d] okay\n", i);
    }
    else
    {
        PRINTF("SPI TX_ADDR[%d] ERROR\n", i);
    }
}

/* Reads FIFO_STATUS register*/
spi_value = nrf_read_fifo_status();

if (spi_value == 0x11)
{
    PRINTF("SPI FIFO_STATUS okay\n");
}
else
{
    PRINTF("SPI FIFO_STATUS ERROR\n");
}

/*Sends the command FLUSH_TX*/
nrf_flush_tx_fifo();

/*Sends the command FLUSH_RX*/
nrf_flush_rx_fifo();
#endif include "spi.h"

/*Initializes the SPI controller*/
void SPI_init()
{
    SIM->SCGC4 |= SIM_SCGC4_SPI0_MASK; // clock gate enable for spi
    SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK; //gpio port d enable
                                            //SIM->SCGC5 |=
    SIM_SCGC5_PORTA_MASK; //gpio port a enable
    //selecting alternate pin functions for spi
    PORTD_PCR0 = PORT_PCR_MUX(0x1); //Enable chip select
    PORTD_PCR1 = PORT_PCR_MUX(0x2); //Enable the SPI_SCK function on
    PTA15
    PORTD_PCR2 = PORT_PCR_MUX(0x2); // Enable the SPI_MOSI function on
    PTA16
    PORTD_PCR3 = PORT_PCR_MUX(0x2); // Enable the SPI_MISO function on
    PTA17
}

```

```

GPIOD_PDDR |= (1 << 0); // Set csn (PORTD.0) as output

SPI0_C1 = SPI_C1_MSTR(1) | SPI_C1_SPE(1); //configured device as
master- MSTR=1, enabled spi SPE = 1
    SPI0_BR |= SPI_BR_SPPR(0) | SPI_BR_SPR(2); //Baud Rate Prescalar as 1
and the Baud Rate Divisor as 4
}

/*Reads a single byte from the SPI bus*/
void SPI_read_byte(uint8_t *byte)
{
    if (byte != NULL)
    {
        // Wait until receive data buffer is full
        while ((SPI0_S & SPI_S_SPRF_MASK) == 0);
        *byte = SPI0_D; // Read data from SPI Data register
    }
}

/*Sends a single byte on the SPI bus*/
void SPI_write_byte(uint8_t byte)
{
    // Wait until data buffer is empty
    while ((SPI0_S & SPI_S_SPTEF_MASK) == 0);
    SPI0_D = byte; // Write data to SPI Data register
}

/*Sends numerous SPI Bytes given a pointer to a byte array and a length
of how many
bytes to send.*/
void SPI_send_packet(uint8_t *p, size_t length)
{
    int i = 0;
    if (p != NULL && length > 0)
    {
        while (i < length)
        {
            SPI_write_byte((uint8_t)(*p));
            p++;
        }
    }
}

/*Blocks until SPI transmit buffer has completed transmitting*/
void SPI_flush()
{
    //Flag is set when transmit data buffer is empty
    while ((SPI0_S & SPI_S_SPTEF_MASK) == 0);
}/***
 * @file data_processing.c
 * @brief Character counting and statistics functions
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-03-04
 */

```

```

*/



#include <stdint.h>
#include "circbuf.h"
#include "platform.h"
#ifndef KL25Z
#include "uart.h"
#endif

uint8_t alphabetical(uint8_t value) // to check if alphabet
{
    if ((value >= 'A' && value <= 'Z') || (value >= 'a' && value <= 'z'))
        return 1;
    else
        return 0;
}

uint8_t numerical(uint8_t value)// to check if number
{
    if (value >= '0' && value <= '9')
        return 1;
    else
        return 0;
}

uint8_t punctuation(uint8_t value) // to check if punctuation
{
    switch ((unsigned char)value)
    {
        case '.':
        case '\'':
        case '\"':
        case ':':
        case ';':
        case ',':
        case '?':
        case '!': return 1;
        default: return 0;
    }
}

void print_data_process_header(void)
{
    PRINTF("----- Character Statistics ----- \n");
    PRINTF("----- \n");
    PRINTF("Letter\t | Number\t | Punctuation\t | Misc\t | Total\n");
    PRINTF("----- \n");
}

void data_process(cb_struct *buf)
{
    //variables to keep count
    static uint32_t alphabet= 0;
    static uint32_t number= 0;
}

```

```

static uint32_t punctuations= 0;
static uint32_t miscell= 0;
static uint32_t current_data= 0;
static uint32_t count = 0;
while (!cb_is_empty(buf))
{
    cb_enum status = cb_buffer_remove_item(buf,
(int8_t*)&current_data);
    if (status == CB_SUCCESS)
    {
        if(alphabetical(current_data))
        {
            alphabet++;
        }
        else if(numerical(current_data))
        {
            number++;
        }
        else if(punctuation(current_data))
        {
            punctuations++;
        }
        else
        {
            miscell++;
        }
        count++;
    }
}

#endif KL25Z
char line[200];
char len = sprintf(line,
" %lu\t | %lu\t\t | %lu\t\t | %lu\t | %lu\r",
alphabet, number, punctuations, miscell, count);
UART_send_n((uint8_t*)line, len);
#else
// Built-in printf forces line buffering before flush
PRINTF(" %lu\t | %lu\t\t | %lu\t\t | %lu\t | %lu\r",
alphabet, number, punctuations, miscell, count);
#endif
/***
 * @file conversion.c
 * @brief source files for conversion.c functions
 * @author Miles FRAIN
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-02-10
 */
#include "conversion.h"
#include "memory.h"

```

```

/*function to convert data back from ascii to integer*/
int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base)
{
    int32_t sign = 0, i = 0, result = 0;

    if (ptr != NULL && digits >= 0 && base != 0)
    {
        if(*ptr == '-')
        {
            sign = 1; //set sign flag
            i++;
        }
        while((*(ptr+i) != '\0') && (i < digits))
        {
            if((*(ptr+i)>= '0') && (*(ptr+i) <= '9')) //if the character
is within 0 to 9
            {
                result = result * base + *(ptr+i) - 48;
                i++;
            }
            else { //the character is between A to F
                result = (result * base) + *(ptr+i) - 55;
                i++;
            }
        }
    }
    if (sign == 1) //if the number is negative
        result = result * -1;
    return result;
}

```

```

/*function to convert data from an integer to ascii*/
uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base)
{
    uint32_t sign = 0, i = 0, remainder = 0; // flag for sign
    uint8_t length = 0;

    if (data < 0) //data is negative
    {
        data = data * -1; // takes the absolute value of data
        sign = 1; // set sign flag
    }
    if (ptr != 0 && data > 0 && base != 0)
    {
        while(data)
        {
            remainder = data % base; //stores the remainder

```

```

        *(ptr+i) = (remainder > 9)?(remainder -10) + 'A' : remainder
+ '0';
        data = data/base; //stores the quotient
        i++;length++;
    }
    if(sign == 1){
        *(ptr+i) = 45; //ascii of '-' is 45
        i++; length++;
    }
    // the reversal of string to get the final ouput as the loop
gives us values from the last ie from lsb
    ptr = my_reverse(ptr,length);
    *(ptr+i) ='\\0'; //null character for the end of string
    //return length;//returns length of array
}
else if(data == 0)
{
    remainder = data % base; //stores the remainder
    *(ptr+i) = (remainder > 9)?(remainder -10) + 'A' : remainder
+ '0';
    data = data/base; //stores the quotient
    i++;length++;
}
return length;
}

```

```

#include "dma.h"
#include "memory_dma.h"
#include "MKL25Z4.h"
#include <stdint.h>

/*mem_enum memmove(uint8_t * src, uint8_t * dst, size_t length)
{
    mem_enum return_status = NO_ERROR;
    int8_t i;

    if ( src == NULL || dst == NULL || length <= 0)           // if
pointers are NULL
    {
        return_status = INVALID_POINTER;
    }
    else if ((dst + length < src) && (src + length < dst))   //if no
overlap
    {
        return_status = NO_OVERLAP;
    }
    else
//destination overlaps source
    {
        for(i = 0; i<length; i++)

```

```

    {
        *(dst+i) = *(src+i);
    }
}

return return_status;
}

mem_enum memset(uint8_t * dst, size_t length, uint8_t value)
{
    mem_enum return_status = NO_ERROR;
    int8_t i;

    if (dst == NULL || length <= 0) // if null pointer
    {
        return_status = INVALID_POINTER;
    }
    else
    {
        for (i = 0; i <length; i++) //copying value to destination
        {
            *(dst + i) = value;
        }
    }
    return return_status;
}

mem_enum memmove_overlap(uint8_t * src, uint8_t * dst, size_t length)
{
    mem_enum return_status = NO_ERROR;
    uint8_t *a = (uint8_t *)malloc(sizeof(uint8_t)); // a is a temp pointer which holds the source address

    for(int i = 0; i < length; i++)
    {
        *(a+i) = *(src+i);
    }

    {
        //order for dma transfer
        //set source address
        DMA_SAR0 = a;
        // Set Destination Address
        DMA_DAR0 = dst;
        // Set BCR for the no of bytes to be transferred
        DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length);
        // Enable interrupt on completion of transfer, Source and destination address increment after every transfer & Set Source and Destination size as 8bit
        DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_SINC_MASK | DMA_DCR_SSIZ(1) | DMA_DCR_DSIZ(1));
        // Start DMA transfer
        DMA_DCR0 |= DMA_DCR_START_MASK;
    }
}

```

```

        //return_status = DST_IN_SRC_OVERLAP;
    }
    return return_status;
}
*/
mem_enum memmove_dma(uint8_t * src, uint8_t * dst, size_t length,
dma_block_size size)
{
    #if 1
    // Working

    //set source address
    DMA_SAR0 = (intptr_t)src;
    // Set Destination Address
    DMA_DAR0 = (intptr_t)dst;
    // Set BCR for the no of bytes to be transferred
    DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length);
    // Enable interrupt on completion of transfer, Source and destination
    address increment after every transfer & Set Source and Destination size
    as 8bit
    //DMA_DCR0 |= (DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
    DMA_DCR_SINC_MASK | DMA_DCR_SSIZE(1) | DMA_DCR_DSIZ(1));
    DMA_DCR0 |= (DMA_DCR_DINC_MASK | DMA_DCR_SINC_MASK |
    DMA_DCR_SSIZE(size) | DMA_DCR_DSIZ(size));
    //return NO_ERROR;
    // Start DMA transfer
    DMA_DCR0 |= DMA_DCR_START_MASK;

    return NO_ERROR;
#endif

    __disable_irq();
    mem_enum return_status;
    int8_t i;

    if ( !src || !dst || !length || !size ) // NULL pointer or zero size
    {
        return ERROR;
    }

    // No overlap case
    if ((dst + length < src) ||
        (src + length < dst))
    {
        //set source address
        DMA_SAR0 = (intptr_t) src;
        // Set Destination Address
        DMA_DAR0 = (intptr_t) dst;
        // Set BCR for the no of bytes to be transferred
        DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length);
    }
}
```

```

        // Enable interrupt on completion of transfer, Source and
        destination address increment after every transfer & Set Source and
        Destination size as 8bit
        DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_SINC_MASK | DMA_DCR_SSIZEx(size) | DMA_DCR_DSIZEx(size));
        // Start DMA transfer
        DMA_DCR0 |= DMA_DCR_START_MASK;

        if (length%size != 0)
        {
            for(i = 0; i<= length%size; i++)
            {
                *(dst + (length - length%size))+i) = *((src + (length -
length%size))+i);
            }

            //return_status = NO_OVERLAP;
            return_status = NO_ERROR;
        }
    }

    else if ((dst > src) && (src + length > dst))           // if source
overlaps destination
    {
        int overlap = src + length - dst; // overlap
        //memmove_overlap(dst, dst + length - overlap, overlap );
//memmove of overlap data
        DMA_SAR0 = (intptr_t) dst;
        // Set Destination Address
        DMA_DAR0 = (intptr_t) (dst + length - overlap);
        // Set BCR for the no of bytes to be transferred
        DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(overlap);
        // Enable interrupt on completion of transfer, Source and
destination address increment after every transfer & Set Source and
Destination size as 8bit
        DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_SINC_MASK | DMA_DCR_SSIZEx(size) | DMA_DCR_DSIZEx(size));
        // Start DMA transfer
        DMA_DCR0 |= DMA_DCR_START_MASK;

        //memmove_overlap(src, dst, length-overlap ); //memmove of the
remaining data
        DMA_SAR0 = (intptr_t) src;
        // Set Destination Address
        DMA_DAR0 =(intptr_t) dst;
        // Set BCR for the no of bytes to be transferred
        DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length-overlap );
        // Enable interrupt on completion of transfer, Source and
destination address increment after every transfer & Set Source and
Destination size as 8bit
        DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_SINC_MASK | DMA_DCR_SSIZEx(size) | DMA_DCR_DSIZEx(size));
        // Start DMA transfer
        DMA_DCR0 |= DMA_DCR_START_MASK;

```

```

        //return_status = SRC_IN_DST_OVERLAP;
        return_status = NO_ERROR;
    }
    else
//destination overlaps source
{
    //set source address
    DMA_SAR0 = (intptr_t) src;
    // Set Destination Address
    DMA_DAR0 = (intptr_t) dst;
    // Set BCR for the no of bytes to be transferred
    DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length);
    // Enable interrupt on completion of transfer, Source and
destination address increment after every transfer & Set Source and
Destination size as 8bit
    DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_SINC_MASK | DMA_DCR_SSIZE(size) | DMA_DCR_DSIZ
e(size));
    // Start DMA transfer
    DMA_DCR0 |= DMA_DCR_START_MASK;

    if (length%size != 0)
    {
        for(i = 0; i<= length%size; i++)
        {
            *((dst + (length - length%size))+i) = *((src + (length -
length%size))+i);
        }
    }

    //return_status = DST_IN_SRC_OVERLAP;
    return_status = NO_ERROR;
}

__enable_irq();
return return_status;
}

mem_enum memset_dma(uint8_t * dst, size_t length, uint8_t data,
dma_block_size size)
{
    uint8_t * b; //temp pointer to store src address
    b = &data;
    mem_enum return_status = NO_ERROR;
    int8_t i;

    if (dst == NULL || length <= 0) //null pointer
    {
        return_status = INVALID_POINTER;
    }
    else if ( size != 1 && size != 2 && size != 4)
    {
        return_status = ERROR;
    }
}

```

```

    else
    {

        // Source Address
        DMA_SAR0 =(intptr_t) b;
        // Destination Address
        DMA_DAR0 = (intptr_t) dst;
        // BCR for the length of bytes to transfer
        DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(length);
        // Enable interrupt on completion of transfer, Source and
        destination address increment after every transfer & Set Source and
        Destination size as 8bit
        DMA_DCR0 |= ( DMA_DCR_EINT_MASK | DMA_DCR_DINC_MASK |
DMA_DCR_DSIZEx(size));
        // Start DMA transfer
        DMA_DCR0 |= DMA_DCR_START_MASK;

        if (length%size != 0)
        {
            for (i = 0; i<= length%size; i++)
            {
                *((dst + (length - length%size ))+ i) = data;
            }
        }
        //else return_status = ERROR;
    }
    return return_status;
}

#include "profiler.h"
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <sys/time.h>
#include "memory.h"

const size_t byte_length[4] = {10, 100, 1000, 5000};
uint32_t start, end, diff;
#ifdef KL25Z

#include "dma.h"
#include "memory_dma.h"
#include "MKL25Z4.h"
#include "MKL25Z4_extension.h"
#include "uart.h"
#include "platform.h"
const dma_block_size size[3] = {ONE_BYTE, TWO_BYTE, FOUR_BYTE};

void systick_init()
{
    SysTick->LOAD |= SysTick_LOAD_RELOAD_Msk;
    //load value is 0X00FFFFFF which is the maximum
}

```

```

    SysTick->CTRL |= (SysTick_CTRL_ENABLE_Msk |
SysTick_CTRL_CLKSOURCE_Msk); //counter enabled, processor clock
    SysTick->VAL = 0;
    //begin = SysTick->VAL;
}

void kl25z_profile_option(profile_test test_type)
{
    uint8_t *source, *dest;
    uint8_t x[10000];
    for (int k = 0; k < 5000; k++)
    {
        x[k] = 1;
    }
    for (int k = 5000; k < 10000; k++)
    {
        x[k] = 2;
    }

    source = &x[0];
    dest = &x[4999];

    // DMA clock setup
    SIM_BWR_SCGC7_DMA(SIM, 1);
    SIM_BWR_SCGC6_DMAMUX(SIM, 1);

    // Enables the DMA channel and select the DMA Channel Source
    DMAMUX0->CHCFG[0] |= DMAMUX_CHCFG_ENBL_MASK | DMAMUX_CHCFG_SOURCE(2);

    // uint8_t *src = &x[0];
    // uint8_t *dst = &x[1000];
    // size_t length = 3;

    #if 0
    for (int j = 1; j < 7; j++)
    {
        length = 16 * j;

        // Setup transfer before and after
        for (int i = 0; i < length; i++)
        {
            src[i] = i * j;
            dst[i] = 0;
        }

        start = SysTick->VAL;
        // For reference, this works fine, not sure why later version
does not work.
        memmove_dma(src, dst, length, FOUR_BYTE);
        end = SysTick->VAL;
        diff = start - end;
        while (DMA_DSR_BCR0 & DMA_DSR_BCR_BCR_MASK) ;
        uint32_t diff2 = start - SysTick->VAL;
        // print report on sizes
    }

```

```

        PRINTF("Time taken for std memmove for transfer is %lu %lu\n",
diff, diff2);

        PRINTF("Transfer complete %d\n", dst[1]);
        // Verify transfer successful
        for (int i = 0; i < length; i++)
        {
            if (dst[i] != src[i])
            {
                PRINTF("Bad transfer\n");
                break;
            }
            //PRINTF("dst[%d] = %d\n", i, dst[i]);
        }
    }

return;
#endif

if (test_type == MEMMOVE_DMA)
{
    PRINTF("trying memmove_dma\n");
    for (int j = 0; j < 4; j++)
    {
        for (int m = 0; m < 3; m++)
        {
            #if 0
            start = SysTick->VAL;
            //memmove_dma(src, dst, length, FOUR_BYTE);
            //memmove_dma(source, dest, byte_length[j], size[m]);
            //memmove_dma(source, dest, 16, FOUR_BYTE);
            memmove_dma(src, dst, length, FOUR_BYTE);
            end = SysTick->VAL;
            diff = start - end;
            while (DMA_DSR_BCR0 & DMA_DSR_BCR_BCR_MASK)
            ;
            #endif

            start = SysTick->VAL;
            /* No idea why this version does not work, but the above
"for reference" version works fine.
What is different between the two calls?
*/
            // memmove_dma(src, dst, length, FOUR_BYTE);
            memmove_dma(source, dest, byte_length[j], size[m]);
            while(DMA_DSR_BCR0 & DMA_DSR_BCR_BSY(1));
            end = SysTick->VAL;
            diff = start - end;

            PRINTF("Time taken for memmove dma for %d byte transfer
is %ld\n", byte_length[j], diff);
        }
    }
}

```

```

if (test_type == MEMSET_DMA)
{
    for (int j = 0; j < 4; j++)
    {
        for (int m = 0; m < 3; m++)
        {
            start = SysTick->VAL;
            memset_dma(dest, byte_length[j], 5, size[m]);
            while(DMA_DSR_BCR0 & DMA_DSR_BCR_BSY(1));
            end = SysTick->VAL;
            diff = start - end;
            // print report on sizes
            PRINTF("Time taken for memset dma for %d byte transfer
is %ld\n", byte_length[j], diff);
        }
    }
}

if (test_type == MY_MEMMOVE)
{
    for (int j = 0; j < 4; j++)
    {
        start = SysTick->VAL;
        my_memmove(source, dest, byte_length[j]);
        end = SysTick->VAL;
        diff = start - end;
        // print report on sizes
        PRINTF("Time taken for my memmove for %d byte transfer is
%ld\n", byte_length[j], diff);
    }
}
if (test_type == MY_MEMSET)
{
    for (int j = 0; j < 4; j++)
    {

        start = SysTick->VAL;
        my_memset(dest, byte_length[j], 10);
        end = SysTick->VAL;
        diff = start - end;
        // print report on sizes
        PRINTF("Time taken for my memset for %d byte transfer is
%ld\n", byte_length[j], diff);
    }
}
if (test_type == MEMMOVE)
{
    for (int j = 0; j < 4; j++)
    {

        start = SysTick->VAL;
        memmove(dest, source, byte_length[j]);
        end = SysTick->VAL;
    }
}

```

```

        diff = start - end;
        // print report on sizes
        PRINTF("Time taken for std memmove for %d byte transfer
is %ld\n", byte_length[j], diff);
    }
}
if (test_type == MEMSET)
{
    for (int j = 0; j < 4; j++)
    {

        start = SysTick->VAL;
        memset(dest, 10, byte_length[j]);
        end = SysTick->VAL;
        diff = start - end;
        // print report on sizes
        PRINTF("Time taken for std memset for %d byte transfer
is %ld\n", byte_length[j], diff);
    }
}
#else

void bbb_profile_option(uint8_t number)
{
    uint8_t *source, *dest;
    uint8_t x[10000];
    for (int k = 0; k < 5000; k++)
    {
        x[k] = 1;
    }
    for (int k = 5000; k < 10000; k++)
    {
        x[k] = 2;
    }

    source = &x[0];
    dest = &x[4999];
    //int i = 0;
    int ret_stat = 0;
    struct timespec start_time, end_time;
    double diff;
    if (number == 1)
    {
        for (int j = 0; j < 4; j++)
        {
            ret_stat = clock_gettime(CLOCK_REALTIME, &start_time);
            memmove(source, dest, byte_length[j]);
            ret_stat = clock_gettime(CLOCK_REALTIME, &end_time);
            diff = ((end_time.tv_sec - start_time.tv_sec) +
(end_time.tv_nsec - start_time.tv_nsec));
            printf("Time taken for std memmove for %d byte transfer is
%lf\n", byte_length[j], diff);
        }
    }
}

```

```

}

if (number == 2)
{
    for (int j = 0; j < 4; j++)
    {
        ret_stat = clock_gettime(CLOCK_REALTIME, &start_time);
        memset(source, 10, byte_length[j]);
        ret_stat = clock_gettime(CLOCK_REALTIME, &end_time);
        diff = ((end_time.tv_sec - start_time.tv_sec) +
(end_time.tv_nsec - start_time.tv_nsec));
        printf("Time taken for std memset for %d byte transfer is
%lf\n", byte_length[j], diff);
    }
}

if (number == 3)
{
    for (int j = 0; j < 4; j++)
    {
        ret_stat = clock_gettime(CLOCK_REALTIME, &start_time);
        my_memset(source, byte_length[j], 10);
        ret_stat = clock_gettime(CLOCK_REALTIME, &end_time);
        diff = ((end_time.tv_sec - start_time.tv_sec) +
(end_time.tv_nsec - start_time.tv_nsec));
        printf("Time taken for my_memset for %d byte transfer is
%lf\n", byte_length[j], diff);
    }
}

if (number == 4)
{
    for (int j = 0; j < 4; j++)
    {
        ret_stat = clock_gettime(CLOCK_REALTIME, &start_time);
        my_memmove(source, dest, byte_length[j]);
        ret_stat = clock_gettime(CLOCK_REALTIME, &end_time);
        diff = ((end_time.tv_sec - start_time.tv_sec) +
(end_time.tv_nsec - start_time.tv_nsec));
        printf("Time taken for my_memmove for %d byte transfer is
%lf\n", byte_length[j], diff);
    }
}

if (number == 5)
{
    // show statistics of all bbb execution times
}
if (ret_stat)
;
// Not used
}

#endif
/*
** ##########
** Processors:          MKL25Z128FM4
**                      MKL25Z128FT4
**                      MKL25Z128LH4
**                      MKL25Z128VLK4

```

```
**
**      Compilers:          Keil ARM C/C++ Compiler
**                           Freescale C/C++ for Embedded ARM
**                           GNU C Compiler
**                           GNU C Compiler - CodeSourcery Sourcery G++
**                           IAR ANSI C/C++ Compiler for ARM
**
**      Reference manual:   KL25P80M48SF0RM, Rev.3, Sep 2012
**      Version:            rev. 2.5, 2015-02-19
**      Build:              b150224
**
**      Abstract:
**                  Provides a system configuration function and a global variable
that
**                  contains the system frequency. It configures the device and
initializes
**                  the oscillator (PLL) that is part of the microcontroller
device.
**
**      Copyright (c) 2015 Freescale Semiconductor, Inc.
**      All rights reserved.
**
**      Redistribution and use in source and binary forms, with or without
modification,
**      are permitted provided that the following conditions are met:
**
**          o Redistributions of source code must retain the above copyright
notice, this list
**              of conditions and the following disclaimer.
**
**          o Redistributions in binary form must reproduce the above
copyright notice, this
**              list of conditions and the following disclaimer in the
documentation and/or
**              other materials provided with the distribution.
**
**          o Neither the name of Freescale Semiconductor, Inc. nor the names
of its
**              contributors may be used to endorse or promote products derived
from this
**              software without specific prior written permission.
**
**      THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
**      ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED
**      WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE
**      DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
BE LIABLE FOR
**      ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES
**      (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES;
```

```

**      LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
** CAUSED AND ON
**      ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
**      (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS
**      SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**

**      http:          www.freescale.com
**      mail:          support@freescale.com
**

**      Revisions:
**      - rev. 1.0 (2012-06-13)
**          Initial version.
**      - rev. 1.1 (2012-06-21)
**          Update according to reference manual rev. 1.
**      - rev. 1.2 (2012-08-01)
**          Device type UARTLP changed to UART0.
**      - rev. 1.3 (2012-10-04)
**          Update according to reference manual rev. 3.
**      - rev. 1.4 (2012-11-22)
**          MCG module - bit LOLS in MCG_S register renamed to LOLSO.
**          NV registers - bit EZPORT_DIS in NV_FOPT register removed.
**      - rev. 1.5 (2013-04-05)
**          Changed start of doxygen comment.
**      - rev. 2.0 (2013-10-29)
**          Register accessor macros added to the memory map.
**          Symbols for Processor Expert memory map compatibility added to
the memory map.
**          Startup file for gcc has been updated according to CMSIS 3.2.
**          System initialization updated.
**      - rev. 2.1 (2014-07-16)
**          Module access macro module_BASES replaced by module_BASE_PTRS.
**          System initialization and startup updated.
**      - rev. 2.2 (2014-08-22)
**          System initialization updated - default clock config changed.
**      - rev. 2.3 (2014-08-28)
**          Update of startup files - possibility to override DefaultISR
added.
**      - rev. 2.4 (2014-10-14)
**          Interrupt INT_LPTimer renamed to INT_LPTMR0.
**      - rev. 2.5 (2015-02-19)
**          Renamed interrupt vector LLW to LLWU.
**
** ##########
*/
/* !
 * @file MKL25Z4
 * @version 2.5
 * @date 2015-02-19
 * @brief Device specific configuration file for MKL25Z4 (implementation
file)
*

```

```

 * Provides a system configuration function and a global variable that
contains
 * the system frequency. It configures the device and initializes the
oscillator
 * (PLL) that is part of the microcontroller device.
 */

#include <stdint.h>
#include "fsl_device_registers.h"

/*
-----
-- Core clock
-----
*/
uint32_t SystemCoreClock = DEFAULT_SYSTEM_CLOCK;

/*
-----
-- SystemInit()
-----
*/
void SystemInit (void) {
#if (DISABLE_WDOG)
/* SIM_COPC: COPT=0,COPCLKS=0,COPW=0 */
SIM->COPC = (uint32_t)0x00u;
#endif /* (DISABLE_WDOG) */
#ifndef CLOCK_SETUP
if((RCM->SRS0 & RCM_SRS0_WAKEUP_MASK) != 0x00U)
{
    if((PMC->REGSC & PMC_REGSC_ACKISO_MASK) != 0x00U)
    {
        PMC->REGSC |= PMC_REGSC_ACKISO_MASK; /* Release hold with ACKISO:
Only has an effect if recovering from VLLSx.*/
    }
}

/* Power mode protection initialization */
#ifndef SYSTEM_SMC_PMPROT_VALUE
SMC->PMPROT = SYSTEM_SMC_PMPROT_VALUE;
#endif

/* System clock initialization */
/* Internal reference clock trim initialization */
#ifndef SLOW_TRIM_ADDRESS
if ( *((uint8_t*)SLOW_TRIM_ADDRESS) != 0xFFU) {
/* Skip if non-volatile flash memory is erased */
MCG->C3 = *((uint8_t*)SLOW_TRIM_ADDRESS);
#endif /* defined(SLOW_TRIM_ADDRESS) */
#ifndef SLOW_FINE_TRIM_ADDRESS

```

```

    MCG->C4 = (MCG->C4 & ~ (MCG_C4_SCFTRIM_MASK)) | ((*(uint8_t*)
SLOW_FINE_TRIM_ADDRESS)) & MCG_C4_SCFTRIM_MASK;
#endif
#if defined(FAST_TRIM_ADDRESS)
    MCG->C4 = (MCG->C4 & ~ (MCG_C4_FCTRIM_MASK)) | ((*(uint8_t*)
FAST_TRIM_ADDRESS)) & MCG_C4_FCTRIM_MASK;
#endif
#if defined(SLOW_TRIM_ADDRESS)
}
#endif /* defined(SLOW_TRIM_ADDRESS) */

/* Set system prescalers and clock sources */
SIM->CLKDIV1 = SYSTEM_SIM_CLKDIV1_VALUE; /* Set system prescalers */
SIM->SOPT1 = ((SIM->SOPT1) & (uint32_t)(~(SIM_SOPT1_OSC32KSEL_MASK))) | 
((SYSTEM_SIM_SOPT1_VALUE) & (SIM_SOPT1_OSC32KSEL_MASK)); /* Set 32 kHz
clock source (ERCLK32K) */
SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_PLLFLLSEL_MASK))) | 
((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_PLLFLLSEL_MASK)); /* Selects the
high frequency clock for various peripheral clocking options. */
SIM->SOPT2 = ((SIM->SOPT2) & (uint32_t)(~(SIM_SOPT2_TPMSRC_MASK))) | 
((SYSTEM_SIM_SOPT2_VALUE) & (SIM_SOPT2_TPMSRC_MASK)); /* Selects the
clock source for the TPM counter clock. */
#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FBI) ||
(MCG_MODE == MCG_MODE_BLPI))
/* Set MCG and OSC */
#if (((SYSTEM_OSC0_CR_VALUE) & OSC_CR_ERCLKEN_MASK) != 0x00U) ||
(((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKENO_MASK) != 0x00U)
/* SIM_SCGC5: PORTA=1 */
SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;
/* PORTA_PCR18: ISF=0,MUX=0 */
PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK |
PORT_PCR_MUX(0x07)));
if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFS0_MASK) != 0x00U) {
/* PORTA_PCR19: ISF=0,MUX=0 */
PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK |
PORT_PCR_MUX(0x07)));
}
#endif
#endif
MCG->SC = SYSTEM_MCG_SC_VALUE; /* Set SC (fast clock internal
reference divider) */
MCG->C1 = SYSTEM_MCG_C1_VALUE; /* Set C1 (clock source selection,
FLL ext. reference divider, int. reference enable etc.) */
/* Check that the source of the FLL reference clock is the requested
one. */
if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {
    while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {
    }
} else {
    while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {
    }
}
MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set
C2 (freq. range, ext. and int. reference selection etc.; low power bit is
set later) */

```

```

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK)); /* Set C4 (FLL output; trim values not changed) */
OSC0->CR = SYSTEM_OSC0_CR_VALUE; /* Set OSC_CR (OSCERCLK enable,
oscillator capacitor load) */
#if (MCG_MODE == MCG_MODE_BLPI)
/* BLPI specific */
MCG->C2 |= (MCG_C2_LP_MASK); /* Disable FLL and PLL in bypass
mode */
#endif

#else /* MCG_MODE */
/* Set MCG and OSC */
/* SIM_SCGC5: PORTA=1 */
SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;
/* PORTA_PCR18: ISF=0,MUX=0 */
PORTA_PCR18 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK |
PORT_PCR_MUX(0x07)));
if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFSD0_MASK) != 0x00U) {
/* PORTA_PCR19: ISF=0,MUX=0 */
PORTA_PCR19 &= (uint32_t)~(uint32_t)((PORT_PCR_ISF_MASK |
PORT_PCR_MUX(0x07)));
}
MCG->SC = SYSTEM_MCG_SC_VALUE; /* Set SC (fast clock internal
reference divider) */
MCG->C2 = (SYSTEM_MCG_C2_VALUE) & (uint8_t)(~(MCG_C2_LP_MASK)); /* Set
C2 (freq. range, ext. and int. reference selection etc.; low power bit is
set later) */
OSC0->CR = SYSTEM_OSC0_CR_VALUE; /* Set OSC_CR (OSCERCLK enable,
oscillator capacitor load) */
#if (MCG_MODE == MCG_MODE_PEE)
MCG->C1 = (SYSTEM_MCG_C1_VALUE) | MCG_C1_CLKS(0x02); /* Set C1 (clock
source selection, FLL ext. reference divider, int. reference enable etc.)
- PBE mode*/
#else
MCG->C1 = SYSTEM_MCG_C1_VALUE; /* Set C1 (clock source selection,
FLL ext. reference divider, int. reference enable etc.) */
#endif
if (((SYSTEM_MCG_C2_VALUE) & MCG_C2_EREFSD0_MASK) != 0x00U) {
while((MCG->S & MCG_S_OSCINIT0_MASK) == 0x00U) { /* Check that the
oscillator is running */
}
}
/* Check that the source of the FLL reference clock is the requested
one. */
if (((SYSTEM_MCG_C1_VALUE) & MCG_C1_IREFS_MASK) != 0x00U) {
while((MCG->S & MCG_S_IREFST_MASK) == 0x00U) {
}
} else {
while((MCG->S & MCG_S_IREFST_MASK) != 0x00U) {
}
}
}
}

```

```

MCG->C4 = ((SYSTEM_MCG_C4_VALUE) & (uint8_t)(~(MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK))) | (MCG->C4 & (MCG_C4_FCTRIM_MASK |
MCG_C4_SCFTRIM_MASK)); /* Set C4 (FLL output; trim values not changed) */
#endif /* MCG_MODE */

/* Common for all MCG modes */

/* PLL clock can be used to generate clock for some devices regardless
of clock generator (MCGOUTCLK) mode. */
MCG->C5 = (SYSTEM_MCG_C5_VALUE) & (uint8_t)(~(MCG_C5_PLLCLKEN0_MASK));
/* Set C5 (PLL settings, PLL reference divider etc.) */
MCG->C6 = (SYSTEM_MCG_C6_VALUE) & (uint8_t)~(MCG_C6_PLLS_MASK); /* Set
C6 (PLL select, VCO divider etc.) */
if ((SYSTEM_MCG_C5_VALUE) & MCG_C5_PLLCLKEN0_MASK) {
    MCG->C5 |= MCG_C5_PLLCLKEN0_MASK; /* PLL clock enable in mode other
than PEE or PBE */
}
/* BLPE, PEE and PBE MCG mode specific */

#if (MCG_MODE == MCG_MODE_BLPE)
    MCG->C2 |= (MCG_C2_LP_MASK); /* Disable FLL and PLL in bypass
mode */
#elif ((MCG_MODE == MCG_MODE_PBE) || (MCG_MODE == MCG_MODE_PEE))
    MCG->C6 |= (MCG_C6_PLLS_MASK); /* Set C6 (PLL select, VCO divider
etc.) */
    while((MCG->S & MCG_S_LOCK0_MASK) == 0x00U) { /* Wait until PLL is
locked*/
}
#endif
#if (MCG_MODE == MCG_MODE_PEE)
    MCG->C1 &= (uint8_t)~(MCG_C1_CLKS_MASK);
#endif
#endif
#if ((MCG_MODE == MCG_MODE_FEI) || (MCG_MODE == MCG_MODE_FEE))
    while((MCG->S & MCG_S_CLKST_MASK) != 0x00U) { /* Wait until output of
the FLL is selected */
}
/* Use LPTMR to wait for 1ms for FLL clock stabilization */
SIM_SCGC5 |= SIM_SCGC5_LPTMR_MASK; /* Allow software control of LPMTR
*/
LPTMR0->CMR = LPTMR_CMR_COMPARE(0); /* Default 1 LPO tick */
LPTMR0->CSR = (LPTMR_CSR_TCF_MASK | LPTMR_CSR_TPS(0x00));
LPTMR0->PSR = (LPTMR_PSR_PCS(0x01) | LPTMR_PSR_PBYP_MASK); /* Clock
source: LPO, Prescaler bypass enable */
LPTMR0->CSR = LPTMR_CSR_TEN_MASK; /* LPMTR enable */
while((LPTMR0_CSR & LPTMR_CSR_TCF_MASK) == 0u) {
}
LPTMR0_CSR = 0x00; /* Disable LPTMR */
SIM_SCGC5 &= (uint32_t)~(uint32_t)SIM_SCGC5_LPTMR_MASK;
#endif
#if ((MCG_MODE == MCG_MODE_FBI) || (MCG_MODE == MCG_MODE_BLPI))
    while((MCG->S & MCG_S_CLKST_MASK) != 0x04U) { /* Wait until internal
reference clock is selected as MCG output */
}
#endif
#if ((MCG_MODE == MCG_MODE_FBE) || (MCG_MODE == MCG_MODE_PBE) ||
(MCG_MODE == MCG_MODE_BLPE))

```

```

while((MCG->S & MCG_S_CLKST_MASK) != 0x08U) { /* Wait until external
reference clock is selected as MCG output */
}
#endif
#if ((SYSTEM_SMC_PMCTRL_VALUE) & SMC_PMCTRL_RUNM_MASK) == (0x02U <<
SMC_PMCTRL_RUNM_SHIFT)
    SMC->PMCTRL = (uint8_t)((SYSTEM_SMC_PMCTRL_VALUE) &
(SMC_PMCTRL_RUNM_MASK)); /* Enable VLPR mode */
    while(SMC->PMSTAT != 0x04U) { /* Wait until the system is in
VLPR mode */
}
#endif

/* PLL loss of lock interrupt request initialization */
if (((SYSTEM_MCG_C6_VALUE) & MCG_C6_LOLIE0_MASK) != 0U) {
    NVIC_EnableIRQ(MCG_IRQn); /* Enable PLL loss of lock
interrupt request */
}
#endif
}

/*
-----
-- SystemCoreClockUpdate()
-----
*/
void SystemCoreClockUpdate (void) {
    uint32_t MCGOUTClock; /* Variable to store output clock
frequency of the MCG module */
    uint16_t Divider;

    if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x00U) {
        /* Output of FLL or PLL is selected */
        if ((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U) {
            /* FLL is selected */
            if ((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U) {
                /* External reference clock is selected */
                MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG
clock */
            }
            if ((MCG->C2 & MCG_C2_RANGE0_MASK) != 0x00U) {
                switch (MCG->C1 & MCG_C1_FRDIV_MASK) {
                    case 0x38U:
                        Divider = 1536U;
                        break;
                    case 0x30U:
                        Divider = 1280U;
                        break;
                    default:
                }
            }
        }
    }
}

```

```

        Divider = (uint16_t) (32LU << ((MCG->C1 & MCG_C1_FRDIV_MASK)
>> MCG_C1_FRDIV_SHIFT));
        break;
    }
} else /* ((MCG->C2 & MCG_C2_RANGE_MASK) != 0x00U) */
    Divider = (uint16_t) (1LU << ((MCG->C1 & MCG_C1_FRDIV_MASK) >>
MCG_C1_FRDIV_SHIFT));
}
MCGOUTClock = (MCGOUTClock / Divider); /* Calculate the divided
FLL reference clock */
} else { /* (!((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U)) */
    MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* The slow internal reference
clock is selected */
} /* (!((MCG->C1 & MCG_C1_IREFS_MASK) == 0x00U)) */
/* Select correct multiplier to calculate the MCG output clock */
switch (MCG->C4 & (MCG_C4_DMX32_MASK | MCG_C4_DRST_DRS_MASK)) {
    case 0x00U:
        MCGOUTClock *= 640U;
        break;
    case 0x20U:
        MCGOUTClock *= 1280U;
        break;
    case 0x40U:
        MCGOUTClock *= 1920U;
        break;
    case 0x60U:
        MCGOUTClock *= 2560U;
        break;
    case 0x80U:
        MCGOUTClock *= 732U;
        break;
    case 0xA0U:
        MCGOUTClock *= 1464U;
        break;
    case 0xC0U:
        MCGOUTClock *= 2197U;
        break;
    case 0xE0U:
        MCGOUTClock *= 2929U;
        break;
    default:
        break;
}
} else { /* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */
/* PLL is selected */
Divider = (((uint16_t)MCG->C5 & MCG_C5_PRDIV0_MASK) + 0x01U);
MCGOUTClock = (uint32_t) (CPU_XTAL_CLK_HZ / Divider); /* Calculate
the PLL reference clock */
Divider = (((uint16_t)MCG->C6 & MCG_C6_VDIV0_MASK) + 24U);
MCGOUTClock *= Divider; /* Calculate the MCG output clock
*/
} /* (!((MCG->C6 & MCG_C6_PLLS_MASK) == 0x00U)) */
} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x40U) {
/* Internal reference clock is selected */

```

```

    if (((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U) {
        MCGOUTClock = CPU_INT_SLOW_CLK_HZ; /* Slow internal reference clock
selected */
    } else { /* (((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U)) */
        Divider = (uint16_t)(0x01LU << ((MCG->SC & MCG_SC_FCRDIV_MASK) >>
MCG_SC_FCRDIV_SHIFT));
        MCGOUTClock = (uint32_t) (CPU_INT_FAST_CLK_HZ / Divider); /* Fast
internal reference clock selected */
    } /* (((MCG->C2 & MCG_C2_IRCS_MASK) == 0x00U)) */
} else if ((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U) {
    /* External reference clock is selected */
    MCGOUTClock = CPU_XTAL_CLK_HZ; /* System oscillator drives MCG
clock */
} else { /* !((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U)) */
    /* Reserved value */
    return;
} /* !((MCG->C1 & MCG_C1_CLKS_MASK) == 0x80U)) */
SystemCoreClock = (MCGOUTClock / (0x01U + ((SIM->CLKDIV1 &
SIM_CLKDIV1_OUTDIV1_MASK) >> SIM_CLKDIV1_OUTDIV1_SHIFT)));
}
/***
 * @file memory.c
 * @brief Custom memory functions
 *
 * - my_memmove
 * - my_memcpy
 * - my_memset
 * - my_memzero
 * - my_reverse
 * - reserve_words
 * - free_words
 *
 * @author Shreya Chakraborty
 * @author Miles Frain
 * @version 1.0
 * @date 2018-02-07
 */

```

```

#include <stdio.h>
#include <stdint.h>
#include <malloc.h>

#include "memory.h"

/*copies a length of bytes from source to destination with no data
corruption in case of overlap*/
uint8_t *my_memmove(uint8_t *src, uint8_t *dst, size_t length)
{
    if((src != NULL) && (dst != NULL) && (length > 0))
    {
        if((src + length > dst) && (dst > src)) //overlap, source has
lower memory
            //copy from opposite end
    }
}

```

```

        for(int32_t i = length -1; i >= 0; i--)
        {
            *(dst+i) = *(src+i);
        }
    }
    else if ((dst + length > src) && (src > dst)) //overlap,
destination has lower memory than source

        //copy from front end
    {
        for(int32_t i = 0; i < length; i++)
        {
            *(dst+i) = *(src+i);
        }
    }
    else if ((src + length <= dst) && (dst > src)) // no overlap,
destination has higher memory than source

        //copy from front end
    {
        for(int32_t i = 0; i < length; i++)
        {
            *(dst+i) = *(src+i);
        }
    }
    else if ((dst + length <= src) && (src > dst)) // no overlap,
destination has lower memory than source

        //copy from front end
    {
        for(int32_t i = 0; i < length; i++)
        {
            *(dst+i) = *(src+i);
        }
    }
    return dst;
}
else
{
    return 0;
}
}

/*copies a length of bytes from source to destination with likely data
corruption in case of overlap*/
uint8_t *my_memcpy(uint8_t *src, uint8_t *dst, size_t length)
{
    if((src != NULL) && (dst != NULL) && (length > 0))
    {
        uint8_t * dst_copy = dst;
        while(length--)
        {
            *dst++ = *src++;
        }
    }
}

```

```

        }
        return dst_copy;
    }
    else return 0;
}

/*sets all location of memory to a given value*/
uint8_t *my_memset(uint8_t *src, size_t length, uint8_t value)
{
    if(src != NULL && (length >0))
    {
        uint8_t *src_copy = src;
        while(length!=0)
        {
            *src = value;
            src++;
            length--;
        }
        return src_copy;
    }
    else return 0;
}

/*sets all the elements of the memory space to 0*/
uint8_t *my_memzero(uint8_t *src, size_t length)
{
    if(src != NULL && (length > 0))
    {
        uint8_t *src_copy = src;
        while(length!=0) //while length is not zero, execute loop
        {
            *src = 0;
            src++;
            length--;
        }
        return src_copy;
    }
    else return 0;
}

/*reverses the order of all bytes*/
uint8_t *my_reverse(uint8_t *src, size_t length)
{
    if(src != NULL && (length > 0))
    {
        uint8_t a, temp, i = 0;

        if(length%2 == 0) //check is length is even
        {
            a = length/2;//for even string length

```

```

        }
    else
    {
        a = (length-1)/2;//for odd string length
    }
    for(i=0;i<a;i++)
    {//swapping to reverse the order of bytes
        temp = *(src+i);
        *(src+i) = *(src+length-1-i);
        *(src+length-1-i) = temp;
    }
    return src;
}

else return 0;
}

/*takes a number of words to allocate in dynamic memory*/
uint32_t *reserve_words(size_t length)
{
    return (uint32_t*)malloc(sizeof(uint32_t)*length);
}

/*frees dynamic memory allocation*/
uint8_t free_words(uint32_t *src)
{
    if(src != NULL)
    {
        free(src);
        return 0;
    }
    else return 1;
}

/*gives the size of string*/
uint32_t string_size(uint8_t *src)
{
    uint32_t count = 0;
    while(*src != '\0') //checking the string and incrementing the count
until null is found
    {
        count++;
        src++;
    }
    return count;
}

#include "dma.h"
#include "MKL25Z4.h"

void dma_clockenable()
{

```

```

        SIM->SCGC7 |= SIM_SCGC7_DMA(1);
        SIM->SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
        SIM->SOPT2 |= SIM_SOPT2_MCGFLLCLK_MASK;
    }

void DMA_config()
{
    dma_clockenable();

    //Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel
    DMAMUX0->CHCFG[0] = DMAMUX_CHCFG_ENBL(0) | DMAMUX_CHCFG_TRIG(0) ;

    // Clearing Source size and Destination size fields.
    DMA_DCR0 &= ~(DMA_DCR_SSIZEMASK | DMA_DCR_DSIZEMASK);

    // Enables the DMA channel and select the DMA Channel Source
    DMAMUX0->CHCFG[0] |= DMAMUX_CHCFG_ENBL_MASK |
    DMAMUX_CHCFG_SOURCE(2);

    NVIC_EnableIRQ(DMA2 IRQn);

    DMA_DCR0 |= DMA_DCR_EINT_MASK;
}

void DMA2_IRQHandler()
{
}

/*
 * Copyright (c) 2013 - 2014, Freescale Semiconductor, Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
modification,
 * are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright
notice, this list
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright
notice, this
 *   list of conditions and the following disclaimer in the documentation
and/or
 *   other materials provided with the distribution.
 *
 * o Neither the name of Freescale Semiconductor, Inc. nor the names of
its
 *   contributors may be used to endorse or promote products derived from
this
 *   software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND

```

```

* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR
* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON
* ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/

```

```

#include "startup.h"
#include "fsl_device_registers.h"

#if (defined(__ICCARM__))
    #pragma section = ".data"
    #pragma section = ".data_init"
    #pragma section = ".bss"
#endif

/***********************
*****
 * Code
*****
***********************/

/*FUNCTION*****
*****
* Function Name : init_data_bss
* Description   : Make necessary initializations for RAM.
* - Copy initialized data from ROM to RAM.
* - Clear the zero-initialized data section.
* - Copy the vector table from ROM to RAM. This could be an option.
*
* Tool Chians:
*   __GNUC__      : GCC
*   __CC_ARM       : KEIL
*   __ICCARM__     : IAR
*
*****
*END*****
*****
void init_data_bss(void)
{
    uint32_t n;

```

```

/* Addresses for VECTOR_TABLE and VECTOR_RAM come from the linker
file */

#if defined(__CC_ARM)
    extern uint32_t Image$$VECTOR_ROM$$Base[];
    extern uint32_t Image$$VECTOR_RAM$$Base[];
    extern uint32_t Image$$RW_m_data$$Base[];

#define __VECTOR_TABLE Image$$VECTOR_ROM$$Base
#define __VECTOR_RAM Image$$VECTOR_RAM$$Base
#define __RAM_VECTOR_TABLE_SIZE (((uint32_t)Image$$RW_m_data$$Base -
(uint32_t)Image$$VECTOR_RAM$$Base))
#elif defined(__ICCARM__)
    extern uint32_t __RAM_VECTOR_TABLE_SIZE[];
    extern uint32_t __VECTOR_TABLE[];
    extern uint32_t __VECTOR_RAM[];
#elif defined(__GNUC__)
    extern uint32_t __VECTOR_TABLE[];
    extern uint32_t __VECTOR_RAM[];
    extern uint32_t __RAM_VECTOR_TABLE_SIZE_BYTES[];
    uint32_t __RAM_VECTOR_TABLE_SIZE =
(uint32_t)(__RAM_VECTOR_TABLE_SIZE_BYTES);
#endif

if (__VECTOR_RAM != __VECTOR_TABLE)
{
    /* Copy the vector table from ROM to RAM */
    for (n = 0; n <
((uint32_t) __RAM_VECTOR_TABLE_SIZE)/sizeof(uint32_t); n++)
    {
        __VECTOR_RAM[n] = __VECTOR_TABLE[n];
    }
    /* Point the VTOR to the position of vector table */
    SCB->VTOR = (uint32_t) __VECTOR_RAM;
}
else
{
    /* Point the VTOR to the position of vector table */
    SCB->VTOR = (uint32_t) __VECTOR_TABLE;
}

#if !defined(__CC_ARM) && !defined(__ICCARM__)

/* Declare pointers for various data sections. These pointers
 * are initialized using values pulled in from the linker file */
uint8_t * data_ram, * data_rom, * data_rom_end;
uint8_t * bss_start, * bss_end;

/* Get the addresses for the .data section (initialized data section)
*/
#if defined(__GNUC__)
    extern uint32_t __DATA_ROM[];
    extern uint32_t __DATA_RAM[];
    extern char __DATA_END[];

```

```

    data_ram = (uint8_t *)__DATA_RAM;
    data_rom = (uint8_t *)__DATA_ROM;
    data_rom_end = (uint8_t *)__DATA_END;
    n = data_rom_end - data_rom;
#endif

/* Copy initialized data from ROM to RAM */
while (n--)
{
    *data_ram++ = *data_rom++;
}

/* Get the addresses for the .bss section (zero-initialized data) */
#if defined(__GNUC__)
    extern char __START_BSS[];
    extern char __END_BSS[];
    bss_start = (uint8_t *)__START_BSS;
    bss_end = (uint8_t *)__END_BSS;
#endif

/* Clear the zero-initialized data section */
n = bss_end - bss_start;
while(n--)
{
    *bss_start++ = 0;
}
#endif /* !__CC_ARM && !__ICCARM__*/
}

/******************
*****
* EOF
*****
*****************/

/***
 * @file uart.c
 * @brief UART functions for transmit and receive
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-03-04
 */
#include "uart.h"
#include "MKL25Z4.h"
#include "MKL25Z4_extension.h"
#include "circbuf.h"
#include "port.h"

cb_struct *rx_buffer = NULL;
cb_struct *tx_buffer = NULL;

```

```

int8_t UART_configure(BAUDRATE baudselect)
{
    if (baudselect != BAUD_115200 &&
        baudselect != BAUD_38400 &&
        baudselect != BAUD_57200 &&
        baudselect != BAUD_9600)
    {
        return BAUDRATE_ERROR;
    }

    // Select the FLL clock source for UART0
    SIM_BWR_SOPT2_UART0SRC(SIM, 1); // UART0 clock source as MCGFLLCLK
    clock or MCGPLLCLK/2 clock (div2 in our case from PLLFLLSEL = 1)

    // uart0 clock gate enable
    SIM_BWR_SCGC4_UART0(SIM, 1); // enable uart0 clock in system clock
    gating control reg

    UART0_BWR_C4_OSR(UART0, UART_OVERSAMPLING - 1); // Write one less
    than oversampling value to OSR

    // Calculation for baud rate selection
    uint16_t baud_rate_divider = 47939584 / (baudselect *
    UART_OVERSAMPLING);
    UART0_WR_BDL(UART0, baud_rate_divider); // LSB of SBR in BDL reg
    UART0_BWR_BDH_SBR(UART0, baud_rate_divider >> 8); // Set remaining 4
    high bits of SBR

    //Enabling RIE Interrupt and the TCIE interrupt now.
    //UART0->C2 |= UART_C2_RIE(1) | UART_C2_TCIE(1);
    UART0->C2 |= UART_C2_RIE(1);

    // Enable UART transmitter and receiver
    UART0_BWR_C2_TE(UART0, 1);
    UART0_BWR_C2_RE(UART0, 1);

    SIM_BWR_SCGC5_PORTA(SIM, 1); // Enable clock on PORTA for UART

    //Enabling the NVIC Interrupt for UART0
    NVIC_EnableIRQ(UART0_IRQn);

    PORT_BWR_PCR_MUX(PORTA, 1, 2); // Alternate function 2, UART0_RX
    PORT_BWR_PCR_MUX(PORTA, 2, 2); // Alternate function 2, UART0_TX

    return 0;
}

void UART_send(uint8_t data)
{
    disable_irq();
    while(!UART0_BRD_S1_TDRE(UART0)); //Waiting for the buffer to get
    empty ie bit TDRE = 1
    UART0->D = data; //write data
}

```

```

// Alternate data write macro
//UART0_WR_D(UART0, data);
// Probably don't need to wait for transmit to complete
//while(!(UART0->S1 & UART_S1_TC_MASK)); //Waiting for transmission
to get complete
    enable_irq();
    //RGB_BLUE_TOGGLE();
}

int _write(int fd, const void *buf, size_t len)
{
    (void)fd;

    // Don't think interrupts need to be disabled for this
    //UART_send_n((uint8_t*)buf, len);

    // Improved version where carriage return added to every newline
    size_t count = 0; // actual number of transmitted chars
    while (len--)
    {
        if (*((uint8_t*)buf == '\n')
        {
            UART_send('\r');
            count++;
        }
        UART_send(*((uint8_t*)buf++));
        count++;
    }

    return count;
}

void UART_send_n(uint8_t* data, size_t length)
{
    __disable_irq();
    if((data == NULL) || (length < 0))
    {
        return;
    }
    else
    {
        for(size_t i = 0;i < length;i++)
        {
            UART_send(* (data + i));
            //UART_send(*data++);
        }
    }
    __enable_irq();
}

void UART_receive(uint8_t* data)
{
    __disable_irq();
    if(data == NULL)

```

```

{
    return;
}
else
{
    while((UART0->S1 & UART_S1_RDRF_MASK) == 0); //Waiting for the
data to recv IE RDRF = 1
    *data = UART0->D; //read data
}
__enable_irq();
}

// Todo - probably best for interrupt receive and polling transmit

void UART_receive_n(uint8_t* data, size_t length)
{
    __disable_irq();
    if((data == NULL) || (length < 0))
    {
        return;
    }
    else
        for(size_t i = 0;i < length;i++)
    {
        UART_receive(data + i);
    }

    __enable_irq();
}
void UART0_IRQHandler()
{
    cb_enum status;
    //if((UART0->S1 & UART_S1_RDRF_MASK) && (UART0->C2 & UART_C2_RIE_MASK))
    // Don't need to check UART_C2_RIE, this is a configuration reg that
we set once
    //if (UART0->S1 & UART_S1_RDRF_MASK)
    if (UART0_BRD_S1_RDRF(UART0))
    {
        RGB_BLUE_TOGGLE();
        // Can probably directly check what triggered interrupt another
way
        //Interrupt caused by receiver
        int8_t data = UART0->D;

        //UART_send(data);

        cb_enum status = cb_buffer_add_item(rx_buffer, data);

        if (status);

        //if (status == CB_SUCCESS)
        //{
        //    //UART0->D = data;
        //}

```

```

    }
    else if ((UART0->S1 & UART_S1_TC_MASK) && (UART0->C2 &
UART_C2_TCIE_MASK))
    {
        //Interrupt caused by transmitter
        int8_t data;
        status = cb_buffer_remove_item(tx_buffer , &data);           //send a
char

        if (status == CB_SUCCESS)
        {
            UART0->D = data;
        }
        //Clear transmit receive interrupt flag
    }
}

#ifndef __PROFILER_H__
#define __PROFILER_H__

#include <stdint.h>
#define CLOCK_REALTIME      (0)
#ifdef KL25Z
//extern const size_t byte_lengths[4];

void systick_init();
void kl25z_profile_option( uint8_t number);
#else

void bbb_profile_option( uint8_t number);

#endif // platform

typedef enum
{
    MEMMOVE_DMA,
    MEMSET_DMA,
    MY_MEMMOVE,
    MY_MEMSET,
    MEMMOVE,
    MEMSET,
} profile_test;

#endif // profiler_h
/***
 * @file memory.h
 * @brief Custom memory functions
 *
 * - my_memmove
 * - my_memcpy
 * - my_memset
 * - my_memzero
 * - my_reverse
 */

```

```

* - reserve_words
* - free_words
* - string_size
*
* @author Shreya Chakraborty
* @author Miles Frain
* @version 1.0
* @date 2018-02-07
*/
#ifndef __MEMORY_H__
#define __MEMORY_H__

/***
 * @brief Move block of memory
 *
 * Copies a length of bytes from source to destination.
 * Can handle overlaps.
 *
 * @param src Memory pointer of where to copy from
 * @param dst Memory pointer of where to copy to
 * @param length Number of bytes to move
 *
 * @return Memory pointer of destination, NULL if src or dst are NULL
 */
uint8_t *my_memmove(uint8_t *src, uint8_t *dst, size_t length);

/***
 * @brief Copy block of memory
 *
 * Copies a length of bytes from source to destination.
 * Undefined behavior if memory regions overlap.
 *
 * @param src Memory pointer of where to copy from
 * @param dst Memory pointer of where to copy to
 * @param length Number of bytes to copy
 *
 * @return Memory pointer of destination, NULL if src or dst are NULL
 */
uint8_t *my_memcpy(uint8_t *src, uint8_t *dst, size_t length);

/***
 * @brief Sets entire memory block to given value
 *
 * @param src Pointer to memory
 * @param length Size of memory block in bytes
 * @param value Value to write to entire memory block
 *
 * @return Pointer to input memory source
 */
uint8_t *my_memset(uint8_t *src, size_t length, uint8_t value);

```

```

/***
 * @brief Sets entire memory block to given zero
 *
 * @param src Pointer to memory
 * @param length Size of memory block in bytes
 *
 * @return Pointer to input memory source
 */
uint8_t *my_memzero(uint8_t *src, size_t length);

/***
 * @brief Reverses order of characters in string
 *
 * @param src Pointer to string
 * @param length Length of string in characters (bytes)
 *
 * @return Pointer to input memory source
 */
uint8_t *my_reverse(uint8_t *src, size_t length);

/***
 * @brief Allocates block of memory
 *
 * @param length Size of memory block to allocate in bytes
 *
 * @return Pointer to allocation, NULL if failure
 */
uint32_t *reserve_words(size_t length);

/***
 * @brief Deallocates a block of memory
 *
 * @param src Pointer to memory to deallocate
 *
 * @return 0 if successful, 1 for failure
 */
uint8_t free_words(uint32_t *src);

/***
 * @brief Finds length of string
 *
 * @param src Pointer to string
 *
 * @return Length of string in characters (bytes), 0 if null pointer
 */
uint32_t string_size(uint8_t *src);

#endif // __MEMORY_H__
/***

```

```

* @file debug.h
* @brief header file for debug.h
* @author Miles Frain
* @author Shreya Chakraborty
* @version 1
* @date 2018-02-10
*/
#ifndef __DEBUG_H__
#define __DEBUG_H__

#include <stdio.h>
#include <stdint.h>

/***
* @brief : takes a pointer to memory and prints the hex output of bytes
*
* @param start: a pointer to the memory location
* @param length: length of bytes to print
*/
void print_array(uint8_t *start, uint32_t length);

#endif /* __DEBUG_H__ */
/***
* @file data.h
* @brief header file for data.h
* @author Miles Frain
* @author Shreya Chakraborty
* @version 1
* @date 2018-02-10
*/
#ifndef __DATA_H__
#define __DATA_H__

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <stddef.h>

#define LITTLE_ENDIANN (0)
#define BIG_ENDIANN (1)
#define SWAP_NO_ERROR (0)
#define SWAP_ERROR (-1)

/***
* @brief :function to report sizes of the ctd data type
*/
void print_cstd_type_sizes();

/***
* @brief :function to report sizes of the stdint data type
*/

```

```

        */
void print_stdint_type_sizes();

/***
 * @brief :function to report sizes of the pointer data type
 */
void print_pointer_sizes();

/***
 * @brief function to determine current endianness config for the program
 *
 * @return returns the determine endianness macro
 */
uint32_t determine_endianness();

/***
 * @brief function to swap current endianness from little to big or from
big to little
 *
 * @param data a pointer to the 1st byte of data ,
 * @param type_length length of the type to swap
 *
 * @return the macro for swap error or swap no error
 */
int32_t swap_data_endianness(uint8_t * data, size_t type_length);

/*function to change from little to big*/
int8_t little_to_big(uint32_t * data);

/*function to change from big to little*/
int8_t big_to_little(uint32_t * data);

#endif /*data.h*/
/***
 * @file circbuf.h
 * @brief circular buffer library
 * @author Shreya Chakraborty
 * @author Miles Ftrain
 * @version 1
 * @date 2018-02-21
 */

#ifndef __CIRCBUF_H__
#define __CIRCBUF_H__

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include "malloc.h"

/***
 * @brief Structure for maintaining handle to circular buffer.
 * Only supports items of 1-byte size.
 * Items added to position after head, and removed from tail.

```

```

*/
typedef struct
{
    int8_t* buffer; // Pointer to buffer in memory
    size_t size; // Total size of buffer
    size_t count; // Number of items in buffer
    int8_t* head; // Pointer to newest item added
    int8_t* tail; // Pointer to oldest item added
}cb_struct;

/**
 * @brief Status return codes for circular buffer functions
 */

typedef enum
{
    CB_SUCCESS, // no error
    CB_NULL_ERROR, // if any pointer parameter is null
    CB_LENGTH_ERROR, // if attempting to initialize with zero length
    CB_FULL_ERROR, // if attempting to add to full buffer
    CB_EMPTY_ERROR, // if attempting to remove from empty buffer
    CB_POSITION_ERROR, // if attempting to peek at invalid position
    CB_ALLOCATION_FAILURE, // if malloc fails
}cb_enum;

/**
 * @brief Initialize circular buffer with 'length' bytes
 * Allocates dynamic memory, and populates struct.
 *
 * @param ptr Pointer to circular buf struct handle.
 * @param length Number of bytes to allocate in buffer
 *
 * @return CB status code
 */
cb_enum cb_init(cb_struct *ptr, size_t length);

/**
 * @brief Free circular buffer dynamic memory and reset struct fields
 *
 * @param ptr Pointer to circular buf struct handle
 *
 * @return CB status code
 */
cb_enum cb_destroy(cb_struct *ptr);

/**
 * @brief Adds byte to circular buffer after head.
 * If buffer is full, will not add item, and instead returns
 * CB_FULL_ERROR.
 *
 * @param ptr Pointer to circular buf struct handle
 * @param data_add Byte to add to buffer
 *
 * @return CB status code
 */

```

```

*/
cb_enum cb_buffer_add_item(cb_struct *ptr, int8_t data_add);

/**
 * @brief Removes byte from tail of circular buffer.
 * If buffer is empty, will not populate data_remove pointer,
 * and will return CB_EMPTY_ERROR.
 *
 * @param ptr Pointer to circular buf struct handle
 * @param data_remove Pointer of where to write data removed.
 *
 * @return CB status code
 */
cb_enum cb_buffer_remove_item(cb_struct *ptr, int8_t *data_remove);

/**
 * @brief Checks if circular buffer is full
 *
 * @param ptr Pointer to circular buf struct handle
 *
 * @return 1 if full, 0 if not full
 * Todo - what should we return if ptr is null?
 */
static inline int cb_is_full(cb_struct *ptr)
{
    if (ptr == NULL || ptr->head == NULL || ptr->tail == NULL || ptr-
>buffer == NULL) //check for null pointer
    {
        return -1; // Evaluates to True / Full
    }
    else if ((ptr->tail == ptr->head + 1) || (ptr->count == ptr->size))
// tail is 1 position ahead of header, buffer is full
    //else if (0)
    {
        return 1; // Full
    }
    return 0; // Not Full
}

/**
 * @brief Checks if circular buffer is empty
 *
 * @param ptr Pointer to circular buf struct handle
 *
 * @return 1 if empty, 0 if not empty
 */
static inline int cb_is_empty(cb_struct *ptr)
{
    if (ptr == NULL || ptr->head == NULL || ptr->tail == NULL || ptr-
>buffer == NULL) //check for null pointer
    {
        return -1; // Evaluates to True / Empty
    }
}

```

```

        else if((ptr->count == 0) && (ptr->tail == ptr->head)) //current item
        count in the buffer is 0 if buffer is empty
    {
        return 1; // Empty
    }
    return 0; // Not empty
}
/***
 * @brief Shows an item at any position in circular buffer
 * without removing item from buffer. If position is not populated,
 * will return CB_POSITION_ERROR.
 *
 * @param ptr Pointer to circular buf struct handle
 * @param position Index from head of where to peek (head is index 0)
 * @param holder Pointer of where to write peeked item.
 *
 * @return CB status code
 */
cb_enum cb_peek(cb_struct *ptr, int8_t position, int8_t *data);

#endif // __CIRCBUF_H__
/***
 * @file conversion.h
 * @brief header files for conversion.h conversion functions
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-02-10
 */

#ifndef CONVERSION_H
#define CONVERSION_H

#include <stdio.h>
#include <stdint.h>

/***
 * @brief :function to convert ascii string back to 32 bit integer and
copy to the pointer address.
 * @param ptr: pointer to location where the output will be copied
 * @param digits: size of input ascii string
 * @param base: base for conversion
 * @return :length of data converted
 */
int32_t my_atoi(uint8_t *ptr, uint8_t digits, uint32_t base);

/***
 * @brief :function to convert 32 bit integer data to ascii string and
copy to the pointer address
 * @param data:32 bit integer to be converted,
 * @param ptr: pointer to the location where output will be copied
 * @param base: base for conversion
 */

```

```

 * @return :length of data converted
 */
uint8_t my_itoa(int32_t data, uint8_t *ptr, uint32_t base);

#endif /*CONVERSION_H*/
/***
 * @file data_processing.h
 * @brief Shows character statistics
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-03-04
 */

#ifndef __DATA_PROCESSING_H__
#define __DATA_PROCESSING_H__

void print_data_process_header(void);

void data_process();

#endif // __DATA_PROCESSING_H__
*****Copyright (C) 2017 by Alex Fosdick - University of Colorado
*****
 * Redistribution, modification or use of this software in source or
binary
 * forms is permitted as long as the files maintain this copyright. Users
are
 * permitted to modify this and use it to learn about the field of
embedded
 * software. Alex Fosdick and the University of Colorado are not liable
for any
 * misuse of this material.
 *

*****
****/
/***
 * @file project1.h
 * @brief This file is to be used to project 1.
 *
 * @author Alex Fosdick
 * @date April 2, 2017
 *
 */
#ifndef __PROJECT1_H__
#define __PROJECT1_H__

#include <stdint.h>

#define DATA_SET_SIZE_W (10)

```

```

#define MEM_SET_SIZE_B (32)
#define MEM_SET_SIZE_W (8)
#define MEM_ZERO_LENGTH (16)

#define TEST_MEMMOVE_LENGTH (16)
#define TEST_ERROR (1)
#define TEST_NO_ERROR (0)
#define TESTCOUNT (8)

/**
 * @brief function to run project1 materials
 *
 * This function calls some various simple tests that you can run to test
 * your code for the project 1. The contents of these functions
 * have been provided.
 *
 * @return void
 */
void project1(void);

/**
 * @brief function to run project1 data operations
 *
 * This function calls the my_itoa and my_atoi functions to validate they
 * work as expected for hexadecimal numbers.
 *
 * @return void
 */
int8_t test_data1();

/**
 * @brief function to run project1 data operations
 *
 * This function calls the my_itoa and my_atoi functions to validate they
 * work as expected for decimal numbers.
 *
 * @return void
 */
int8_t test_data2();

/**
 * @brief function to test the non-overlapped memmove operation
 *
 * This function calls the memmove routine with two sets of data that do
 * not
 * overlap in anyway. This function should print that a move worked
 * correctly
 * for a move from source to destination.
 *
 * @return void
 */
int8_t test_memmove1();

/**

```

```

 * @brief function to test an overlapped Memmove operation Part 1
 *
 * This function calls the memmove routine with two sets of data that not
 * overlap. Overlap exists at the start of the destination and the end
 * of the
 * source pointers. This function should print that a move worked
 * correctly
 * for a move from source to destination regardless of overlap.
 *
 * @return void
 */
int8_t test_memmove2();

/***
 * @brief function to run project1 memmove overlapped test
 *
 * This function calls the memmove routine with two sets of data that not
 * overlap. Overlap exists at the start of the source and the end of the
 * destination pointers. This function should print that a move worked
 * correctly
 * for a move from source to destination regardless of overlap.
 *
 * @return void
 */
int8_t test_memmove3();

/***
 * @brief function to test the memcpy functionality
 *
 * This function calls the my_memcpy functions to validate a copy works
 * correctly.
 *
 * @return void
 */
int8_t test_memcpy();

/***
 * @brief function to test the memset and memzero functionality
 *
 * This function calls the memset and memzero functions. This shoudl zero
 * out
 * the bytes from [] to []. This should set the bytes [] to [] with 0xFF.
 *
 * @return void
 */
int8_t test_memset();

/***
 * @brief function to test the reverse functionality
 *
 * This function calls the my_reverse function to see if a give set of
 * ASCII
 * characters will properly reverse.
 *

```

```

* @return void
*/
int8_t test_reverse();

#endif /* __PROJECT1_H__ */
/***
 * @file platform.h
 * @brief header files for platform.h to define various platforms like
host,platform,bbb
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-02-10
 */

#ifndef PLATFORM_H
#define PLATFORM_H

#include <stdio.h>

#ifdef KL25Z
    #include "MKL25Z4.h"
    // overwrote _write() to give KL25Z printf
    #define PRINTF(...) printf( __VA_ARGS__ )
    #define BEGIN_CRITICAL __disable_irq()
    #define END_CRITICAL __enable_irq()
#else
    #define PRINTF(...) printf( __VA_ARGS__ )
    #define BEGIN_CRITICAL
    #define END_CRITICAL
#endif

#endif
#ifndef __CLOCK_H__
#define __CLOCK_H__
void clock_setup();
#endif // __CLOCK_H__
/*
** #####
** Processors:          MKL25Z128FM4
**                      MKL25Z128FT4
**                      MKL25Z128LH4
**                      MKL25Z128VLK4
**
** Compilers:            Keil ARM C/C++ Compiler
**                      Freescale C/C++ for Embedded ARM
**                      GNU C Compiler
**                      GNU C Compiler - CodeSourcery Sourcery G++
**                      IAR ANSI C/C++ Compiler for ARM
**
** Reference manual:    KL25P80M48SF0RM, Rev.3, Sep 2012
** Version:              rev. 2.5, 2015-02-19
** Build:               b150224
**

```

```
**      Abstract:  
**          Provides a system configuration function and a global variable  
that  
**          contains the system frequency. It configures the device and  
initializes  
**          the oscillator (PLL) that is part of the microcontroller  
device.  
**  
**          Copyright (c) 2015 Freescale Semiconductor, Inc.  
**          All rights reserved.  
**  
**          Redistribution and use in source and binary forms, with or without  
modification,  
**          are permitted provided that the following conditions are met:  
**  
**              o Redistributions of source code must retain the above copyright  
notice, this list  
**                  of conditions and the following disclaimer.  
**  
**              o Redistributions in binary form must reproduce the above  
copyright notice, this  
**                  list of conditions and the following disclaimer in the  
documentation and/or  
**                  other materials provided with the distribution.  
**  
**              o Neither the name of Freescale Semiconductor, Inc. nor the names  
of its  
**                  contributors may be used to endorse or promote products derived  
from this  
**                  software without specific prior written permission.  
**  
**          THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND  
CONTRIBUTORS "AS IS" AND  
**          ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
THE IMPLIED  
**          WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
ARE  
**          DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS  
BE LIABLE FOR  
**          ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
CONSEQUENTIAL DAMAGES  
**          (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
**          LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON  
**          ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR  
TORT  
**          (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  
USE OF THIS  
**          SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
**  
**          http:                      www.freescale.com  
**          mail:                     support@freescale.com  
**
```

```

**      Revisions:
**      - rev. 1.0 (2012-06-13)
**          Initial version.
**      - rev. 1.1 (2012-06-21)
**          Update according to reference manual rev. 1.
**      - rev. 1.2 (2012-08-01)
**          Device type UARTLP changed to UART0.
**      - rev. 1.3 (2012-10-04)
**          Update according to reference manual rev. 3.
**      - rev. 1.4 (2012-11-22)
**          MCG module - bit LOLS in MCG_S register renamed to LOLS0.
**          NV registers - bit EZPORT_DIS in NV_FOPT register removed.
**      - rev. 1.5 (2013-04-05)
**          Changed start of doxygen comment.
**      - rev. 2.0 (2013-10-29)
**          Register accessor macros added to the memory map.
**          Symbols for Processor Expert memory map compatibility added to
the memory map.
**          Startup file for gcc has been updated according to CMSIS 3.2.
**          System initialization updated.
**      - rev. 2.1 (2014-07-16)
**          Module access macro module_BASES replaced by module_BASE_PTRS.
**          System initialization and startup updated.
**      - rev. 2.2 (2014-08-22)
**          System initialization updated - default clock config changed.
**      - rev. 2.3 (2014-08-28)
**          Update of startup files - possibility to override DefaultISR
added.
**      - rev. 2.4 (2014-10-14)
**          Interrupt INT_LPTimer renamed to INT_LPTMR0.
**      - rev. 2.5 (2015-02-19)
**          Renamed interrupt vector LLW to LLWU.
**
** ##########
*/

```

```

/*!
* @file MKL25Z4
* @version 2.5
* @date 2015-02-19
* @brief Device specific configuration file for MKL25Z4 (header file)
*
* Provides a system configuration function and a global variable that
contains
* the system frequency. It configures the device and initializes the
oscillator
* (PLL) that is part of the microcontroller device.
*/

```

```

#ifndef SYSTEM_MKL25Z4_H_
#define SYSTEM_MKL25Z4_H_                                /**< Symbol preventing
repeated inclusion */

```

```
#ifdef __cplusplus
```

```

extern "C" {
#endif

#include <stdint.h>

#ifndef DISABLE_WDOG
#define DISABLE_WDOG           1
#endif


/* MCG mode constants */

#define MCG_MODE_FEI           0U
#define MCG_MODE_FBI           1U
#define MCG_MODE_BLPI          2U
#define MCG_MODE_FEE           3U
#define MCG_MODE_FBE           4U
#define MCG_MODE_BLPE          5U
#define MCG_MODE_PBE           6U
#define MCG_MODE_PEE           7U

/* Predefined clock setups
   0 ... Default part configuration
      Multipurpose Clock Generator (MCG) in FEI mode.
      Reference clock source for MCG module: Slow internal reference
clock
      Core clock = 20.97152MHz
      Bus clock  = 20.97152MHz
   1 ... Maximum achievable clock frequency configuration
      Multipurpose Clock Generator (MCG) in PEE mode.
      Reference clock source for MCG module: System oscillator
reference clock
      Core clock = 48MHz
      Bus clock  = 24MHz
   2 ... Chip internally clocked, ready for Very Low Power Run mode
      Multipurpose Clock Generator (MCG) in BLPI mode.
      Reference clock source for MCG module: Fast internal reference
clock
      Core clock = 4MHz
      Bus clock  = 0.8MHz
   3 ... Chip externally clocked, ready for Very Low Power Run mode
      Multipurpose Clock Generator (MCG) in BLPE mode.
      Reference clock source for MCG module: System oscillator
reference clock
      Core clock = 4MHz
      Bus clock  = 1MHz
   4 ... USB clock setup
      Multipurpose Clock Generator (MCG) in PEE mode.
      Reference clock source for MCG module: System oscillator
reference clock
      Core clock = 48MHz

```

```

        Bus clock = 24MHz
 */

/* Define clock source values */

#define CPU_XTAL_CLK_HZ           8000000u      /* Value of
the external crystal or oscillator clock frequency in Hz */
#define CPU_INT_SLOW_CLK_HZ       32768u       /* Value of
the slow internal oscillator clock frequency in Hz */
#define CPU_INT_FAST_CLK_HZ       4000000u     /* Value of
the fast internal oscillator clock frequency in Hz */

/* RTC oscillator setting */

/* Low power mode enable */
/* SMC_PMPROT: AVLP=1,ALLS=1,AVLLS=1 */
#define SYSTEM_SMC_PMPROT_VALUE    0x2AU          /* SMC_PMPROT
*/

/* Internal reference clock trim */
/* #undef SLOW_TRIM_ADDRESS */                      /* Slow
oscillator not trimmed. Commented out for MISRA compliance. */
/* #undef SLOW_FINE_TRIM_ADDRESS */                  /* Slow
oscillator not trimmed. Commented out for MISRA compliance. */
/* #undef FAST_TRIM_ADDRESS */                      /* Fast
oscillator not trimmed. Commented out for MISRA compliance. */
/* #undef FAST_FINE_TRIM_ADDRESS */                 /* Fast
oscillator not trimmed. Commented out for MISRA compliance. */

#endif CLOCK_SETUP
#if (CLOCK_SETUP == 0)
#define DEFAULT_SYSTEM_CLOCK        20971520u     /* Default
System clock value */
#define MCG_MODE                   MCG_MODE_FEI /* Clock generator
mode */
/* MCG_C1: CLKS=0,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE        0x06U          /* MCG_C1 */
/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSS0=1,LP=0,IRCS=0 */
#define SYSTEM_MCG_C2_VALUE        0x24U          /* MCG_C2 */
/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE        0x00U          /* MCG_C4 */
/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=0,LOCSS0=0 */
#define SYSTEM_MCG_SC_VALUE        0x00U          /* MCG_SC */
/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=0 */
#define SYSTEM_MCG_C5_VALUE        0x00U          /* MCG_C5 */
/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE        0x00U          /* MCG_C6 */
/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE       0x80U          /* OSC0_CR */
/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE    0x00U          /* SMC_PMCTRL
*/
/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=0 */

```

```

#define SYSTEM_SIM_CLKDIV1_VALUE      0x00U           /* SIM_CLKDIV1
*/
/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE      0x000C0000U        /* SIM_SOPT1
*/
/* SIM_SOPT2:
UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE      0x01000000U        /* SIM_SOPT2
*/
#elif (CLOCK_SETUP == 1)
#define DEFAULT_SYSTEM_CLOCK      48000000u          /* Default
System clock value */
#define MCG_MODE                  MCG_MODE_PEE /* Clock generator
mode */
/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE      0x1AU             /* MCG_C1 */
/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFOS0=1,LP=0,IRCS=0 */
#define SYSTEM_MCG_C2_VALUE      0x24U             /* MCG_C2 */
/* MCG_C4: DMX32=0,DRST_DRS=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE      0x00U             /* MCG_C4 */
/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSrv=0,FCRDIV=0,LOCs0=0 */
#define SYSTEM_MCG_SC_VALUE      0x00U             /* MCG_SC */
/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=1 */
#define SYSTEM_MCG_C5_VALUE      0x01U             /* MCG_C5 */
/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE      0x40U             /* MCG_C6 */
/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE     0x80U             /* OSC0_CR */
/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE  0x00U             /* SMC_PMCTRL
*/
/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */
#define SYSTEM_SIM_CLKDIV1_VALUE 0x10010000U        /* SIM_CLKDIV1
*/
/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE  0x000C0000U        /* SIM_SOPT1
*/
/* SIM_SOPT2:
UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFLSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE  0x01010000U        /* SIM_SOPT2
*/
#elif (CLOCK_SETUP == 2)
#define DEFAULT_SYSTEM_CLOCK      4000000u          /* Default
System clock value */
#define MCG_MODE                  MCG_MODE_BLPI /* Clock generator
mode */
/* MCG_C1: CLKS=1,FRDIV=0,IREFS=1,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE      0x46U             /* MCG_C1 */
/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFOS0=1,LP=1,IRCS=1 */
#define SYSTEM_MCG_C2_VALUE      0x27U             /* MCG_C2 */
/* MCG_C4: DMX32=0,DRST_DRS=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE      0x00U             /* MCG_C4 */
/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSrv=0,FCRDIV=0,LOCs0=0 */
#define SYSTEM_MCG_SC_VALUE      0x00U             /* MCG_SC */

```

```

/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=0 */
#define SYSTEM_MCG_C5_VALUE 0x00U /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE 0x00U /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE 0x80U /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE 0x00U /* SMC_PMCTRL */

/*
/* SIM_CLKDIV1: OUTDIV1=0,OUTDIV4=4 */
#define SYSTEM_SIM_CLKDIV1_VALUE 0x00040000U /* SIM_CLKDIV1 */

/*
/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */

/*
/* SIM_SOPT2:
UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE 0x02000000U /* SIM_SOPT2 */

#endif

#if (CLOCK_SETUP == 3)
#define DEFAULT_SYSTEM_CLOCK 4000000u /* Default System clock value */
#define MCG_MODE MCG_MODE_BLPE /* Clock generator mode */
/* MCG_C1: CLKS=2,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE 0x9AU /* MCG_C1 */

/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSS0=1,LP=1,IRCS=1 */
#define SYSTEM_MCG_C2_VALUE 0x27U /* MCG_C2 */

/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE 0x00U /* MCG_C4 */

/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLTPRSRV=0,FCRDIV=1,LOCS0=0 */
#define SYSTEM_MCG_SC_VALUE 0x02U /* MCG_SC */

/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=0 */
#define SYSTEM_MCG_C5_VALUE 0x00U /* MCG_C5 */

/* MCG_C6: LOLIE0=0,PLLS=0,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE 0x00U /* MCG_C6 */

/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE 0x80U /* OSC0_CR */

/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE 0x00U /* SMC_PMCTRL */

/*
/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=3 */
#define SYSTEM_SIM_CLKDIV1_VALUE 0x10030000U /* SIM_CLKDIV1 */

/*
/* SIM_SOPT1: USBREGEN=0,USBSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */

/*
/* SIM_SOPT2:
UART0SRC=0,TPMSRC=2,USBSRC=0,PLLFLSEL=0,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE 0x02000000U /* SIM_SOPT2 */

#endif

#if (CLOCK_SETUP == 4)
#define DEFAULT_SYSTEM_CLOCK 48000000u /* Default System clock value */

```

```

#define MCG_MODE_MCG_MODE_PEE /* Clock generator
mode */
/* MCG_C1: CLKS=0,FRDIV=3,IREFS=0,IRCLKEN=1,IREFSTEN=0 */
#define SYSTEM_MCG_C1_VALUE 0x1AU /* MCG_C1 */
/* MCG_C2: LOCRE0=0,RANGE0=2,HGO0=0,EREFSDIV=1,LP=0,IRCS=0 */
#define SYSTEM_MCG_C2_VALUE 0x24U /* MCG_C2 */
/* MCG_C4: DMX32=0,DRST_DRST=0,FCTRIM=0,SCFTRIM=0 */
#define SYSTEM_MCG_C4_VALUE 0x00U /* MCG_C4 */
/* MCG_SC: ATME=0,ATMS=0,ATMF=0,FLT_PSRV=0,FCRDIV=0,LOCSDIV=0 */
#define SYSTEM_MCG_SC_VALUE 0x00U /* MCG_SC */
/* MCG_C5: PLLCLKEN0=0,PLLSTEN0=0,PRDIV0=1 */
#define SYSTEM_MCG_C5_VALUE 0x01U /* MCG_C5 */
/* MCG_C6: LOLIE0=0,PLLS=1,CME0=0,VDIV0=0 */
#define SYSTEM_MCG_C6_VALUE 0x40U /* MCG_C6 */
/* OSC0_CR: ERCLKEN=1,EREFSTEN=0,SC2P=0,SC4P=0,SC8P=0,SC16P=0 */
#define SYSTEM_OSC0_CR_VALUE 0x80U /* OSC0_CR */
/* SMC_PMCTRL: RUNM=0,STOPA=0,STOPM=0 */
#define SYSTEM_SMC_PMCTRL_VALUE 0x00U /* SMC_PMCTRL */
*/
/* SIM_CLKDIV1: OUTDIV1=1,OUTDIV4=1 */
#define SYSTEM_SIM_CLKDIV1_VALUE 0x10010000U /* SIM_CLKDIV1 */
*/
/* SIM_SOPT1: USBREGEN=0,USBSSSTBY=0,USBVSTBY=0,OSC32KSEL=3 */
#define SYSTEM_SIM_SOPT1_VALUE 0x000C0000U /* SIM_SOPT1 */
*/
/* SIM_SOPT2:
UART0SRC=0,TPMSRC=1,USBSRC=0,PLLFIILSEL=1,CLKOUTSEL=0,RTCCLKOUTSEL=0 */
#define SYSTEM_SIM_SOPT2_VALUE 0x01010000U /* SIM_SOPT2 */
*/

#endif
#else
#define DEFAULT_SYSTEM_CLOCK 20971520U /* Default
System clock value */
#endif

/**
 * @brief System clock frequency (core clock)
 *
 * The system clock frequency supplied to the SysTick timer and the
processor
 * core clock. This variable can be used by the user application to setup
the
 * SysTick timer or configure other parameters. It may also be used by
debugger to
 * query the frequency of the debug timer or configure the trace clock
speed
 * SystemCoreClock is initialized with a correct predefined value.
 */
extern uint32_t SystemCoreClock;

/**
 * @brief Setup the microcontroller system.
*

```

```

 * Typically this function configures the oscillator (PLL) that is part
of the
 * microcontroller device. For systems with variable clock speed it also
updates
 * the variable SystemCoreClock. SystemInit is called from startup_device
file.
 */
void SystemInit (void);

/***
 * @brief Updates the SystemCoreClock variable.
 *
 * It must be called whenever the core clock is changed during program
 * execution. SystemCoreClockUpdate() evaluates the clock register
settings and calculates
 * the current core clock.
 */
void SystemCoreClockUpdate (void);

#ifndef __cplusplus
}
#endif

#endif /* #if !defined(SYSTEM_MKL25Z4_H_) */

/** #####
** Version:          rev. 2.5, 2015-02-19
** Build:           b150224
**
** Abstract:
**     Register bit field access macros.
**
** Copyright (c) 2015 Freescale Semiconductor, Inc.
** All rights reserved.
**
** Redistribution and use in source and binary forms, with or without
modification,
** are permitted provided that the following conditions are met:
**
**     o Redistributions of source code must retain the above copyright
notice, this list
**         of conditions and the following disclaimer.
**
**     o Redistributions in binary form must reproduce the above
copyright notice, this
**         list of conditions and the following disclaimer in the
documentation and/or
**         other materials provided with the distribution.
**
**     o Neither the name of Freescale Semiconductor, Inc. nor the names
of its
**         contributors may be used to endorse or promote products derived
from this
**         software without specific prior written permission.

```

\*\*  
\*\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND  
CONTRIBUTORS "AS IS" AND  
\*\* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
THE IMPLIED  
\*\* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
ARE  
\*\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS  
BE LIABLE FOR  
\*\* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
CONSEQUENTIAL DAMAGES  
\*\* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
\*\* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON  
\*\* ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR  
TORT  
\*\* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  
USE OF THIS  
\*\* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
\*\*  
\*\* http: www.freescale.com  
\*\* mail: support@freescale.com  
\*\*  
\*\* Revisions:  
\*\* - rev. 1.0 (2012-06-13)  
\*\*     Initial version.  
\*\* - rev. 1.1 (2012-06-21)  
\*\*     Update according to reference manual rev. 1.  
\*\* - rev. 1.2 (2012-08-01)  
\*\*     Device type UARTLP changed to UART0.  
\*\* - rev. 1.3 (2012-10-04)  
\*\*     Update according to reference manual rev. 3.  
\*\* - rev. 1.4 (2012-11-22)  
\*\*     MCG module - bit LOLS in MCG\_S register renamed to LOLSO.  
\*\*     NV registers - bit EZPORT\_DIS in NV\_FOPT register removed.  
\*\* - rev. 2.0 (2013-10-29)  
\*\*     Register accessor macros added to the memory map.  
\*\*     Symbols for Processor Expert memory map compatibility added to  
the memory map.  
\*\*     Startup file for gcc has been updated according to CMSIS 3.2.  
\*\*     System initialization updated.  
\*\* - rev. 2.1 (2014-07-16)  
\*\*     Module access macro module\_BASES replaced by module\_BASE\_PTRS.  
\*\*     System initialization and startup updated.  
\*\* - rev. 2.2 (2014-08-22)  
\*\*     System initialization updated - default clock config changed.  
\*\* - rev. 2.3 (2014-08-28)  
\*\*     Update of startup files - possibility to override DefaultISR  
added.  
\*\* - rev. 2.4 (2014-10-14)  
\*\*     Interrupt INT\_LPTimer renamed to INT\_LPTMR0.  
\*\* - rev. 2.5 (2015-02-19)  
\*\*     Renamed interrupt vector LLW to LLWU.

```

** #####
*/
#ifndef _FSL_BITACCESS_H
#define _FSL_BITACCESS_H 1

#include <stdint.h>
#include <stdlib.h>

#define BME_AND_MASK  (1<<26)
#define BME_OR_MASK   (1<<27)
#define BME_XOR_MASK  (3<<26)
#define BME_BFI_MASK(BIT,WIDTH)  (1<<28) | (BIT<<23) | ((WIDTH-1)<<19)
#define BME_UBFX_MASK(BIT,WIDTH)  (1<<28) | (BIT<<23) | ((WIDTH-1)<<19)

/* Decorated Store: Logical AND */
#define BME_AND8(addr, wdata)  (*((volatile uint8_t*)((uintptr_t)addr | BME_AND_MASK)) = wdata)
#define BME_AND16(addr, wdata) (*((volatile uint16_t*)((uintptr_t)addr | BME_AND_MASK)) = wdata)
#define BME_AND32(addr, wdata) (*((volatile uint32_t*)((uintptr_t)addr | BME_AND_MASK)) = wdata)

/* Decorated Store: Logical OR */
#define BME_OR8(addr, wdata)  (*((volatile uint8_t*)((uintptr_t)addr | BME_OR_MASK)) = wdata)
#define BME_OR16(addr, wdata) (*((volatile uint16_t*)((uintptr_t)addr | BME_OR_MASK)) = wdata)
#define BME_OR32(addr, wdata) (*((volatile uint32_t*)((uintptr_t)addr | BME_OR_MASK)) = wdata)

/* Decorated Store: Logical XOR */
#define BME_XOR8(addr, wdata) (*((volatile uint8_t*)((uintptr_t)addr | BME_XOR_MASK)) = wdata)
#define BME_XOR16(addr, wdata) (*((volatile uint16_t*)((uintptr_t)addr | BME_XOR_MASK)) = wdata)
#define BME_XOR32(addr, wdata) (*((volatile uint32_t*)((uintptr_t)addr | BME_XOR_MASK)) = wdata)

/* Decorated Store: Bit Field Insert */
#define BME_BFI8(addr, wdata, bit, width) (*((volatile uint8_t*)((uintptr_t)addr | BME_BFI_MASK(bit,width))) = wdata)
#define BME_BFI16(addr, wdata, bit, width) (*((volatile uint16_t*)((uintptr_t)addr | BME_BFI_MASK(bit,width))) = wdata)
#define BME_BFI32(addr, wdata, bit, width) (*((volatile uint32_t*)((uintptr_t)addr | BME_BFI_MASK(bit,width))) = wdata)

/* Decorated Load: Unsigned Bit Field Extract */
#define BME_UBFX8(addr, bit, width) (*((volatile uint8_t*)((uintptr_t)addr | BME_UBFX_MASK(bit,width)))) 
#define BME_UBFX16(addr, bit, width) (*((volatile uint16_t*)((uintptr_t)addr | BME_UBFX_MASK(bit,width)))) 

```

```

#define BME_UBFX32(addr, bit, width) (*(volatile  
uint32_t*)((uintptr_t)addr | BME_UBFX_MASK(bit,width)))

#endif /* _FSL_BITACCESS_H */

*****/  
**  
* @file uart.h  
* @brief uart library for KL25Z  
* @author Shreya Chakraborty  
* @author Miles Frain  
* @version 1  
* @date 2018-02-21  
*/  
  
#include "MKL25Z4.h"  
#include "circbuf.h"  
  
#ifndef __UART_H__  
#define __UART_H__  
  
/* Todo BAUD rate calculation macro  
 * See section 39.2.1 of KL25 reference manual  
 *  
 * "The 13 bits in SBR[12:0] are referred to collectively as BR, and  
 * they set the modulo divide rate for the baud rate generator. When  
 * BR is 1 - 8191, the baud rate equals baud clock / ((OSR+1) Ð  
 * BR)."  
 */  
  
#define UART_OVERSAMPLING 16  
#define BAUDRATE_ERROR (-1)  
#define NULL_POINTER_ERROR (-1)  
//#define UART_S1_TDRE_MASK (0x80)  
//#define UART_S1_RDRF_MASK (0x20)  
//#define UART_S1_TC_MASK (0x40)  
  
typedef enum BAUD_RATE  
{  
    BAUD_115200 = 115200,  
    BAUD_38400 = 38400,  
    BAUD_57200 = 57200,  
    BAUD_9600 = 9600,  
}  
BAUDRATE;  
  
extern cb_struct *rx_buffer;  
extern cb_struct *tx_buffer;  
  
int8_t UART_configure(BAUDRATE baudselect);  
void UART_send(uint8_t data);  
void UART_send_n(uint8_t* data, size_t length);

```

```

void UART_receive(uint8_t* data);
void UART_receive_n(uint8_t* data, size_t length);
void UART0_IRQHandler();
int _write(int fd, const void *buf, size_t count);

#endif // __UART_H__
#ifndef __SPI_H__
#define __SPI_H__

#include "MKL25Z4.h"
#include <stdint.h>
#include <stdlib.h>

/*Initializes the SPI controller*/
void SPI_init();

/*Reads a single byte from the SPI bus*/
void SPI_read_byte(uint8_t *byte);

/*Sends a single byte on the SPI bus*/
void SPI_write_byte(uint8_t byte);

/*Sends numerous SPI Bytes given a pointer to a byte array and a length
of how many
bytes to send.*/
void SPI_send_packet(uint8_t *p, size_t length);

/*Blocks until SPI transmit buffer has completed transmitting*/
void SPI_flush();
#endif
/*
** ##### Processor: MKL25Z128FM4
** MKL25Z128FT4
** MKL25Z128LH4
** MKL25Z128VLK4
**
** Compiler: Keil ARM C/C++ Compiler
** Freescale C/C++ for Embedded ARM
** GNU C Compiler
** GNU C Compiler - CodeSourcery Sourcery G++
** IAR ANSI C/C++ Compiler for ARM
**
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012
** Version: rev. 2.5, 2015-02-19
** Build: b150721
**
** Abstract:
** CMSIS Peripheral Access Layer for MKL25Z4
**
** Copyright (c) 1997 - 2015 Freescale Semiconductor, Inc.
** All rights reserved.
*/

```

\*\* Redistribution and use in source and binary forms, with or without  
modification,  
\*\* are permitted provided that the following conditions are met:  
\*\*  
\*\* o Redistributions of source code must retain the above copyright  
notice, this list  
\*\* of conditions and the following disclaimer.  
\*\*  
\*\* o Redistributions in binary form must reproduce the above  
copyright notice, this  
\*\* list of conditions and the following disclaimer in the  
documentation and/or  
\*\* other materials provided with the distribution.  
\*\*  
\*\* o Neither the name of Freescale Semiconductor, Inc. nor the names  
of its  
\*\* contributors may be used to endorse or promote products derived  
from this  
\*\* software without specific prior written permission.  
\*\*  
\*\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND  
CONTRIBUTORS "AS IS" AND  
\*\* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
THE IMPLIED  
\*\* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
ARE  
\*\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS  
BE LIABLE FOR  
\*\* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
CONSEQUENTIAL DAMAGES  
\*\* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
\*\* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON  
\*\* ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR  
TORT  
\*\* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  
USE OF THIS  
\*\* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
\*\*  
\*\* http: www.freescale.com  
\*\* mail: support@freescale.com  
\*\*  
\*\* Revisions:  
\*\* - rev. 1.0 (2012-06-13)  
\*\* Initial version.  
\*\* - rev. 1.1 (2012-06-21)  
\*\* Update according to reference manual rev. 1.  
\*\* - rev. 1.2 (2012-08-01)  
\*\* Device type UARTLP changed to UART0.  
\*\* - rev. 1.3 (2012-10-04)  
\*\* Update according to reference manual rev. 3.  
\*\* - rev. 1.4 (2012-11-22)  
\*\* MCG module - bit LOLS in MCG\_S register renamed to LOLS0.

```

**          NV registers - bit EZPORT_DIS in NV_FOPT register removed.
**          - rev. 1.5 (2013-04-05)
**          Changed start of doxygen comment.
**          - rev. 2.0 (2013-10-29)
**          Register accessor macros added to the memory map.
**          Symbols for Processor Expert memory map compatibility added to
the memory map.
**          Startup file for gcc has been updated according to CMSIS 3.2.
**          System initialization updated.
**          - rev. 2.1 (2014-07-16)
**          Module access macro module_BASES replaced by module_BASE_PTRS.
**          System initialization and startup updated.
**          - rev. 2.2 (2014-08-22)
**          System initialization updated - default clock config changed.
**          - rev. 2.3 (2014-08-28)
**          Update of startup files - possibility to override DefaultISR
added.
**          - rev. 2.4 (2014-10-14)
**          Interrupt INT_LPTimer renamed to INT_LPTMR0.
**          - rev. 2.5 (2015-02-19)
**          Renamed interrupt vector LLW to LLWU.
**
** ##########
*/

```

```

/* !
* @file MKL25Z4.h
* @version 2.5
* @date 2015-02-19
* @brief CMSIS Peripheral Access Layer for MKL25Z4
*
* CMSIS Peripheral Access Layer for MKL25Z4
*/

```

```

/* -----
-- MCU activation
-----
*/
/* Prevention from multiple including the same memory map */
#ifndef MKL25Z4_H_ /* Check if memory map has not been already
included */
#define MKL25Z4_H_
#define MCU_MKL25Z4

/* Check if another memory map has not been also included */
#if (defined(MCU_ACTIVE))
    #error MKL25Z4 memory map: There is already included another memory
map. Only one memory map can be included.
#endif /* (defined(MCU_ACTIVE)) */
#define MCU_ACTIVE

```

```

#include <stdint.h>

/** Memory map major version (memory maps with equal major version number
are
 * compatible) */
#define MCU_MEM_MAP_VERSION 0x0200u
/** Memory map minor version */
#define MCU_MEM_MAP_VERSION_MINOR 0x0005u

/* -----
-- Interrupt vector numbers
-----
-- */

/*!
* @addtogroup Interrupt_vector_numbers Interrupt vector numbers
* @{
*/
/* Interrupt Number Definitions */
#define NUMBER_OF_INT_VECTORS 48           /***< Number of
interrupts in the Vector table */

typedef enum IRQn {
    /* Auxiliary constants */
    NotAvail_IRQn          = -128,           /***< Not available
device specific interrupt */

    /* Core interrupts */
    NonMaskableInt_IRQn     = -14,            /***< Non Maskable
Interrupt */
    HardFault_IRQn          = -13,             /***< Cortex-M0 SV Hard
Fault Interrupt */
    SVCall_IRQn             = -5,              /***< Cortex-M0 SV Call
Interrupt */
    PendSV_IRQn             = -2,               /***< Cortex-M0 Pend SV
Interrupt */
    SysTick_IRQn             = -1,               /***< Cortex-M0 System
Tick Interrupt */

    /* Device specific interrupts */
    DMA0_IRQn               = 0,                /***< DMA channel 0
transfer complete */
    DMA1_IRQn               = 1,                /***< DMA channel 1
transfer complete */
    DMA2_IRQn               = 2,                /***< DMA channel 2
transfer complete */
    DMA3_IRQn               = 3,                /***< DMA channel 3
transfer complete */
    Reserved20_IRQn         = 4,                /***< Reserved
interrupt */
}

```

```

    FTFA IRQn          = 5,           /**< Command complete
and read collision */
    LVD_LVW IRQn      = 6,           /**< Low-voltage
detect, low-voltage warning */
    LLWU IRQn         = 7,           /**< Low leakage
wakeup Unit */
    I2C0 IRQn         = 8,           /**< I2C0 interrupt */
    I2C1 IRQn         = 9,           /**< I2C1 interrupt */
    SPI0 IRQn         = 10,          /**< SPI0 single
interrupt vector for all sources */
    SPI1 IRQn         = 11,          /**< SPI1 single
interrupt vector for all sources */
    UART0 IRQn        = 12,          /**< UART0 status and
error */
    UART1 IRQn        = 13,          /**< UART1 status and
error */
    UART2 IRQn        = 14,          /**< UART2 status and
error */
    ADC0 IRQn         = 15,          /**< ADC0 interrupt */
    CMP0 IRQn         = 16,          /**< CMP0 interrupt */
    TPM0 IRQn         = 17,          /**< TPM0 single
interrupt vector for all sources */
    TPM1 IRQn         = 18,          /**< TPM1 single
interrupt vector for all sources */
    TPM2 IRQn         = 19,          /**< TPM2 single
interrupt vector for all sources */
    RTC IRQn          = 20,          /**< RTC alarm */
    RTC_Seconds IRQn   = 21,          /**< RTC seconds */
    PIT IRQn          = 22,          /**< PIT interrupt */
    Reserved39 IRQn    = 23,          /**< Reserved
interrupt */
    USB0 IRQn         = 24,          /**< USB0 interrupt */
    DAC0 IRQn         = 25,          /**< DAC0 interrupt */
    TSI0 IRQn         = 26,          /**< TSI0 interrupt */
    MCG IRQn          = 27,          /**< MCG interrupt */
    LPTMR0 IRQn        = 28,          /**< LPTMR0 interrupt
*/
    Reserved45 IRQn    = 29,          /**< Reserved
interrupt */
    PORTA IRQn        = 30,          /**< PORTA Pin detect
*/
    PORTD IRQn         = 31,          /**< PORTD Pin detect
*/
} IRQn_Type;

/* !
 * @}
 */
/* /* end of group Interrupt_vector_numbers */

/*
-----  

-- Cortex M0 Core Configuration

```

```

----- */
----- */

/*!
 * @addtogroup Cortex_Core_Configuration Cortex M0 Core Configuration
 * @{
 */

#define __CM0PLUS_REV          0x0000    /**< Core revision r0p0
*/
#define __MPU_PRESENT           0          /**< Defines if an MPU
is present or not */
#define __VTOR_PRESENT          1          /**< Defines if an MPU
is present or not */
#define __NVIC_PRIO_BITS        2          /**< Number of priority
bits implemented in the NVIC */
#define __Vendor_SysTickConfig   0          /**< Vendor specific
implementation of SysTickConfig is defined */

#include "core_cm0plus.h"           /* Core Peripheral Access Layer */
#include "system_MKL25Z4.h"         /* Device specific configuration
file */

/*!
 * @}
 */
/* end of group Cortex_Core_Configuration */

/*
----- --
-- Device Peripheral Access Layer
----- */
----- */

/*!
 * @addtogroup Peripheral_access_layer Device Peripheral Access Layer
 * @{
 */

/*
** Start of section using anonymous unions
*/
#if defined(__ARMCC_VERSION)
#pragma push
#pragma anon_unions
#elif defined(__CWCC__)
#pragma push
#pragma cpp_extensions on
#elif defined(__GNUC__)
/* anonymous unions are enabled by default */
#elif defined(__IAR_SYSTEMS_ICC__)
#pragma language=extended

```

```

#else
    #error Not supported compiler type
#endif

/* -----
-- ADC Peripheral Access Layer
-----
*/
/*!
 * @addtogroup ADC_Peripheral_Access_Layer ADC Peripheral Access Layer
 * @{
 */

/** ADC - Register Layout Typedef */
typedef struct {
    __IO uint32_t SC1[2];                                /**< ADC Status and
Control Registers 1, array offset: 0x0, array step: 0x4 */
    __IO uint32_t CFG1;                                  /**< ADC Configuration
Register 1, offset: 0x8 */
    __IO uint32_t CFG2;                                  /**< ADC Configuration
Register 2, offset: 0xC */
    __I  uint32_t R[2];                                  /**< ADC Data Result
Register, array offset: 0x10, array step: 0x4 */
    __IO uint32_t CV1;                                  /**< Compare Value
Registers, offset: 0x18 */
    __IO uint32_t CV2;                                  /**< Compare Value
Registers, offset: 0x1C */
    __IO uint32_t SC2;                                  /**< Status and
Control Register 2, offset: 0x20 */
    __IO uint32_t SC3;                                  /**< Status and
Control Register 3, offset: 0x24 */
    __IO uint32_t OFS;                                 /**< ADC Offset
Correction Register, offset: 0x28 */
    __IO uint32_t PG;                                   /**< ADC Plus-Side
Gain Register, offset: 0x2C */
    __IO uint32_t MG;                                   /**< ADC Minus-Side
Gain Register, offset: 0x30 */
    __IO uint32_t CLPD;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x34 */
    __IO uint32_t CLPS;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x38 */
    __IO uint32_t CLP4;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x3C */
    __IO uint32_t CLP3;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x40 */
    __IO uint32_t CLP2;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x44 */
    __IO uint32_t CLP1;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x48 */
    __IO uint32_t CLP0;                                /**< ADC Plus-Side
General Calibration Value Register, offset: 0x4C */
    uint8_t RESERVED_0[4];
}

```

```

    __IO uint32_t CLMD;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x54 */
    __IO uint32_t CLMS;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x58 */
    __IO uint32_t CLM4;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x5C */
    __IO uint32_t CLM3;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x60 */
    __IO uint32_t CLM2;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x64 */
    __IO uint32_t CLM1;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x68 */
    __IO uint32_t CLM0;                                /**< ADC Minus-Side
General Calibration Value Register, offset: 0x6C */
} ADC_Type, *ADC_MemMapPtr;

/* -----
-- ADC - Register accessor macros
-----
*/
/* !
 * @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor
macros
 * @{
 */

```

```

/* ADC - Register accessors */
#define ADC_SC1_REG(base, index)          ((base)->SC1[index])
#define ADC_SC1_COUNT                     2
#define ADC_CFG1_REG(base)                ((base)->CFG1)
#define ADC_CFG2_REG(base)                ((base)->CFG2)
#define ADC_R_REG(base, index)            ((base)->R[index])
#define ADC_R_COUNT                      2
#define ADC_CV1_REG(base)                 ((base)->CV1)
#define ADC_CV2_REG(base)                 ((base)->CV2)
#define ADC_SC2_REG(base)                 ((base)->SC2)
#define ADC_SC3_REG(base)                 ((base)->SC3)
#define ADC_OFS_REG(base)                 ((base)->OFS)
#define ADC_PG_REG(base)                  ((base)->PG)
#define ADC_MG_REG(base)                  ((base)->MG)
#define ADC_CLPD_REG(base)                ((base)->CLPD)
#define ADC_CLPS_REG(base)                ((base)->CLPS)
#define ADC_CLP4_REG(base)                ((base)->CLP4)
#define ADC_CLP3_REG(base)                ((base)->CLP3)
#define ADC_CLP2_REG(base)                ((base)->CLP2)
#define ADC_CLP1_REG(base)                ((base)->CLP1)
#define ADC_CLP0_REG(base)                ((base)->CLP0)
#define ADC_CLMD_REG(base)                ((base)->CLMD)
#define ADC_CLMS_REG(base)                ((base)->CLMS)
#define ADC_CLM4_REG(base)                ((base)->CLM4)
#define ADC_CLM3_REG(base)                ((base)->CLM3)

```

```

#define ADC_CLM2_REG(base)          ((base)->CLM2)
#define ADC_CLM1_REG(base)          ((base)->CLM1)
#define ADC_CLM0_REG(base)          ((base)->CLM0)

/*!
 * @}
 */
/* end of group ADC_Register_Accessor_Macros */

/*
-----  

-- ADC Register Masks  

-----  

----- */

/*!  

 * @addtogroup ADC_Register_Masks ADC Register Masks
 * @{
 */

/* SC1 Bit Fields */
#define ADC_SC1_ADCH_MASK           0x1Fu
#define ADC_SC1_ADCH_SHIFT          0
#define ADC_SC1_ADCH_WIDTH          5
#define ADC_SC1_ADCH(x)             (((uint32_t)((uint32_t)(x))<<ADC_SC1_ADCH_SHIFT))&ADC_SC1_ADCH_MASK
#define ADC_SC1_DIFF_MASK            0x20u
#define ADC_SC1_DIFF_SHIFT           5
#define ADC_SC1_DIFF_WIDTH           1
#define ADC_SC1_DIFF(x)              (((uint32_t)((uint32_t)(x))<<ADC_SC1_DIFF_SHIFT))&ADC_SC1_DIFF_MASK
#define ADC_SC1_AIEN_MASK            0x40u
#define ADC_SC1_AIEN_SHIFT           6
#define ADC_SC1_AIEN_WIDTH           1
#define ADC_SC1_AIEN(x)              (((uint32_t)((uint32_t)(x))<<ADC_SC1_AIEN_SHIFT))&ADC_SC1_AIEN_MASK
#define ADC_SC1_COCO_MASK            0x80u
#define ADC_SC1_COCO_SHIFT           7
#define ADC_SC1_COCO_WIDTH           1
#define ADC_SC1_COCO(x)              (((uint32_t)((uint32_t)(x))<<ADC_SC1_COCO_SHIFT))&ADC_SC1_COCO_MASK
/* CFG1 Bit Fields */
#define ADC_CFG1_ADICLK_MASK          0x3u
#define ADC_CFG1_ADICLK_SHIFT         0
#define ADC_CFG1_ADICLK_WIDTH         2
#define ADC_CFG1_ADICLK(x)             (((uint32_t)((uint32_t)(x))<<ADC_CFG1_ADICLK_SHIFT))&ADC_CFG1_ADICLK_MASK
#define ADC_CFG1_MODE_MASK             0xCu
#define ADC_CFG1_MODE_SHIFT            2
#define ADC_CFG1_MODE_WIDTH            2
#define ADC_CFG1_MODE(x)               (((uint32_t)((uint32_t)(x))<<ADC_CFG1_MODE_SHIFT))&ADC_CFG1_MODE_MASK
#define ADC_CFG1_ADLSMP_MASK           0x10u

```

```

#define ADC_CFG1_ADLSMP_SHIFT 4
#define ADC_CFG1_ADLSMP_WIDTH 1
#define ADC_CFG1_ADLSMP(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG1_ADLSMP_SHIFT))&ADC_CFG1_ADLSMP_MASK
#define ADC_CFG1_ADIV_MASK 0x60u
#define ADC_CFG1_ADIV_SHIFT 5
#define ADC_CFG1_ADIV_WIDTH 2
#define ADC_CFG1_ADIV(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG1_ADIV_SHIFT))&ADC_CFG1_ADIV_MASK
#define ADC_CFG1_ADLPC_MASK 0x80u
#define ADC_CFG1_ADLPC_SHIFT 7
#define ADC_CFG1_ADLPC_WIDTH 1
#define ADC_CFG1_ADLPC(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG1_ADLPC_SHIFT))&ADC_CFG1_ADLPC_MASK
/* CFG2 Bit Fields */
#define ADC_CFG2_ADLSTS_MASK 0x3u
#define ADC_CFG2_ADLSTS_SHIFT 0
#define ADC_CFG2_ADLSTS_WIDTH 2
#define ADC_CFG2_ADLSTS(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG2_ADLSTS_SHIFT))&ADC_CFG2_ADLSTS_MASK
#define ADC_CFG2_ADHSC_MASK 0x4u
#define ADC_CFG2_ADHSC_SHIFT 2
#define ADC_CFG2_ADHSC_WIDTH 1
#define ADC_CFG2_ADHSC(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG2_ADHSC_SHIFT))&ADC_CFG2_ADHSC_MASK
#define ADC_CFG2_ADACKEN_MASK 0x8u
#define ADC_CFG2_ADACKEN_SHIFT 3
#define ADC_CFG2_ADACKEN_WIDTH 1
#define ADC_CFG2_ADACKEN(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG2_ADACKEN_SHIFT))&ADC_CFG2_ADACKEN_MASK
#define ADC_CFG2_MUXSEL_MASK 0x10u
#define ADC_CFG2_MUXSEL_SHIFT 4
#define ADC_CFG2_MUXSEL_WIDTH 1
#define ADC_CFG2_MUXSEL(x) (((uint32_t)((uint32_t)(x))<<ADC_CFG2_MUXSEL_SHIFT))&ADC_CFG2_MUXSEL_MASK
/* R Bit Fields */
#define ADC_R_D_MASK 0xFFFFu
#define ADC_R_D_SHIFT 0
#define ADC_R_D_WIDTH 16
#define ADC_R_D(x) (((uint32_t)((uint32_t)(x))<<ADC_R_D_SHIFT))&ADC_R_D_MASK
/* CV1 Bit Fields */
#define ADC_CV1_CV_MASK 0xFFFFu
#define ADC_CV1_CV_SHIFT 0
#define ADC_CV1_CV_WIDTH 16
#define ADC_CV1_CV(x) (((uint32_t)((uint32_t)(x))<<ADC_CV1_CV_SHIFT))&ADC_CV1_CV_MASK
/* CV2 Bit Fields */
#define ADC_CV2_CV_MASK 0xFFFFu
#define ADC_CV2_CV_SHIFT 0

```

```

#define ADC_CV2_CV_WIDTH          16
#define ADC_CV2_CV(x) (((uint32_t)((uint32_t)(x))<<ADC_CV2_CV_SHIFT))&ADC_CV2_CV_MASK)
/* SC2 Bit Fields */
#define ADC_SC2_REFSEL_MASK        0x3u
#define ADC_SC2_REFSEL_SHIFT       0
#define ADC_SC2_REFSEL_WIDTH       2
#define ADC_SC2_REFSEL(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_REFSEL_SHIFT))&ADC_SC2_REFSEL_MASK)
#define ADC_SC2_DMAEN_MASK         0x4u
#define ADC_SC2_DMAEN_SHIFT        2
#define ADC_SC2_DMAEN_WIDTH        1
#define ADC_SC2_DMAEN(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_DMAEN_SHIFT))&ADC_SC2_DMAEN_MASK)
#define ADC_SC2_ACREN_MASK         0x8u
#define ADC_SC2_ACREN_SHIFT        3
#define ADC_SC2_ACREN_WIDTH        1
#define ADC_SC2_ACREN(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_ACREN_SHIFT))&ADC_SC2_ACREN_MASK)
#define ADC_SC2_ACFGTE_MASK        0x10u
#define ADC_SC2_ACFGTE_SHIFT       4
#define ADC_SC2_ACFGTE_WIDTH       1
#define ADC_SC2_ACFGTE(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_ACFGTE_SHIFT))&ADC_SC2_ACFGTE_MASK)
#define ADC_SC2_ACFE_MASK          0x20u
#define ADC_SC2_ACFE_SHIFT         5
#define ADC_SC2_ACFE_WIDTH         1
#define ADC_SC2_ACFE(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_ACFE_SHIFT))&ADC_SC2_ACFE_MASK)
#define ADC_SC2_ADTRG_MASK         0x40u
#define ADC_SC2_ADTRG_SHIFT        6
#define ADC_SC2_ADTRG_WIDTH        1
#define ADC_SC2_ADTRG(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_ADTRG_SHIFT))&ADC_SC2_ADTRG_MASK)
#define ADC_SC2_ADACT_MASK         0x80u
#define ADC_SC2_ADACT_SHIFT        7
#define ADC_SC2_ADACT_WIDTH        1
#define ADC_SC2_ADACT(x) (((uint32_t)((uint32_t)(x))<<ADC_SC2_ADACT_SHIFT))&ADC_SC2_ADACT_MASK)
/* SC3 Bit Fields */
#define ADC_SC3_AVGS_MASK          0x3u
#define ADC_SC3_AVGS_SHIFT         0
#define ADC_SC3_AVGS_WIDTH         2
#define ADC_SC3_AVGS(x) (((uint32_t)((uint32_t)(x))<<ADC_SC3_AVGS_SHIFT))&ADC_SC3_AVGS_MASK)
#define ADC_SC3_AVGE_MASK          0x4u
#define ADC_SC3_AVGE_SHIFT         2
#define ADC_SC3_AVGE_WIDTH         1
#define ADC_SC3_AVGE(x) (((uint32_t)((uint32_t)(x))<<ADC_SC3_AVGE_SHIFT))&ADC_SC3_AVGE_MASK)
#define ADC_SC3_ADCO_MASK          0x8u
#define ADC_SC3_ADCO_SHIFT         3
#define ADC_SC3_ADCO_WIDTH         1

```

```

#define ADC_SC3_ADCO(x)
(((uint32_t) (((uint32_t)(x))<<ADC_SC3_ADCO_SHIFT)) &ADC_SC3_ADCO_MASK)
#define ADC_SC3_CALF_MASK 0x40u
#define ADC_SC3_CALF_SHIFT 6
#define ADC_SC3_CALF_WIDTH 1
#define ADC_SC3_CALF(x)
(((uint32_t) (((uint32_t)(x))<<ADC_SC3_CALF_SHIFT)) &ADC_SC3_CALF_MASK)
#define ADC_SC3_CAL_MASK 0x80u
#define ADC_SC3_CAL_SHIFT 7
#define ADC_SC3_CAL_WIDTH 1
#define ADC_SC3_CAL(x)
(((uint32_t) (((uint32_t)(x))<<ADC_SC3_CAL_SHIFT)) &ADC_SC3_CAL_MASK)
/* OFS Bit Fields */
#define ADC_OFS_OFS_MASK 0xFFFFu
#define ADC_OFS_OFS_SHIFT 0
#define ADC_OFS_OFS_WIDTH 16
#define ADC_OFS_OFS(x)
(((uint32_t) (((uint32_t)(x))<<ADC_OFS_OFS_SHIFT)) &ADC_OFS_OFS_MASK)
/* PG Bit Fields */
#define ADC_PG_PG_MASK 0xFFFFu
#define ADC_PG_PG_SHIFT 0
#define ADC_PG_PG_WIDTH 16
#define ADC_PG_PG(x)
(((uint32_t) (((uint32_t)(x))<<ADC_PG_PG_SHIFT)) &ADC_PG_PG_MASK)
/* MG Bit Fields */
#define ADC_MG_MG_MASK 0xFFFFu
#define ADC_MG_MG_SHIFT 0
#define ADC_MG_MG_WIDTH 16
#define ADC_MG_MG(x)
(((uint32_t) (((uint32_t)(x))<<ADC_MG_MG_SHIFT)) &ADC_MG_MG_MASK)
/* CLPD Bit Fields */
#define ADC_CLPD_CLPD_MASK 0x3Fu
#define ADC_CLPD_CLPD_SHIFT 0
#define ADC_CLPD_CLPD_WIDTH 6
#define ADC_CLPD_CLPD(x)
(((uint32_t) (((uint32_t)(x))<<ADC_CLPD_CLPD_SHIFT)) &ADC_CLPD_CLPD_MASK)
/* CLPS Bit Fields */
#define ADC_CLPS_CLPS_MASK 0x3Fu
#define ADC_CLPS_CLPS_SHIFT 0
#define ADC_CLPS_CLPS_WIDTH 6
#define ADC_CLPS_CLPS(x)
(((uint32_t) (((uint32_t)(x))<<ADC_CLPS_CLPS_SHIFT)) &ADC_CLPS_CLPS_MASK)
/* CLP4 Bit Fields */
#define ADC_CLP4_CLP4_MASK 0x3FFu
#define ADC_CLP4_CLP4_SHIFT 0
#define ADC_CLP4_CLP4_WIDTH 10
#define ADC_CLP4_CLP4(x)
(((uint32_t) (((uint32_t)(x))<<ADC_CLP4_CLP4_SHIFT)) &ADC_CLP4_CLP4_MASK)
/* CLP3 Bit Fields */
#define ADC_CLP3_CLP3_MASK 0x1FFu
#define ADC_CLP3_CLP3_SHIFT 0
#define ADC_CLP3_CLP3_WIDTH 9
#define ADC_CLP3_CLP3(x)
(((uint32_t) (((uint32_t)(x))<<ADC_CLP3_CLP3_SHIFT)) &ADC_CLP3_CLP3_MASK)

```

```

/* CLP2 Bit Fields */
#define ADC_CLP2_CLP2_MASK          0xFFu
#define ADC_CLP2_CLP2_SHIFT         0
#define ADC_CLP2_CLP2_WIDTH         8
#define ADC_CLP2_CLP2(x) (((uint32_t)((uint32_t)(x))<<ADC_CLP2_CLP2_SHIFT))&ADC_CLP2_CLP2_MASK)
/* CLP1 Bit Fields */
#define ADC_CLP1_CLP1_MASK          0x7Fu
#define ADC_CLP1_CLP1_SHIFT         0
#define ADC_CLP1_CLP1_WIDTH         7
#define ADC_CLP1_CLP1(x) (((uint32_t)((uint32_t)(x))<<ADC_CLP1_CLP1_SHIFT))&ADC_CLP1_CLP1_MASK)
/* CLP0 Bit Fields */
#define ADC_CLP0_CLP0_MASK          0x3Fu
#define ADC_CLP0_CLP0_SHIFT         0
#define ADC_CLP0_CLP0_WIDTH         6
#define ADC_CLP0_CLP0(x) (((uint32_t)((uint32_t)(x))<<ADC_CLP0_CLP0_SHIFT))&ADC_CLP0_CLP0_MASK)
/* CLMD Bit Fields */
#define ADC_CLMD_CLMD_MASK          0x3Fu
#define ADC_CLMD_CLMD_SHIFT         0
#define ADC_CLMD_CLMD_WIDTH         6
#define ADC_CLMD_CLMD(x) (((uint32_t)((uint32_t)(x))<<ADC_CLMD_CLMD_SHIFT))&ADC_CLMD_CLMD_MASK)
/* CLMS Bit Fields */
#define ADC_CLMS_CLMS_MASK          0x3Fu
#define ADC_CLMS_CLMS_SHIFT         0
#define ADC_CLMS_CLMS_WIDTH         6
#define ADC_CLMS_CLMS(x) (((uint32_t)((uint32_t)(x))<<ADC_CLMS_CLMS_SHIFT))&ADC_CLMS_CLMS_MASK)
/* CLM4 Bit Fields */
#define ADC_CLM4_CLM4_MASK          0x3FFFu
#define ADC_CLM4_CLM4_SHIFT         0
#define ADC_CLM4_CLM4_WIDTH         10
#define ADC_CLM4_CLM4(x) (((uint32_t)((uint32_t)(x))<<ADC_CLM4_CLM4_SHIFT))&ADC_CLM4_CLM4_MASK)
/* CLM3 Bit Fields */
#define ADC_CLM3_CLM3_MASK          0x1FFu
#define ADC_CLM3_CLM3_SHIFT         0
#define ADC_CLM3_CLM3_WIDTH         9
#define ADC_CLM3_CLM3(x) (((uint32_t)((uint32_t)(x))<<ADC_CLM3_CLM3_SHIFT))&ADC_CLM3_CLM3_MASK)
/* CLM2 Bit Fields */
#define ADC_CLM2_CLM2_MASK          0xFFu
#define ADC_CLM2_CLM2_SHIFT         0
#define ADC_CLM2_CLM2_WIDTH         8
#define ADC_CLM2_CLM2(x) (((uint32_t)((uint32_t)(x))<<ADC_CLM2_CLM2_SHIFT))&ADC_CLM2_CLM2_MASK)
/* CLM1 Bit Fields */
#define ADC_CLM1_CLM1_MASK          0x7Fu
#define ADC_CLM1_CLM1_SHIFT         0
#define ADC_CLM1_CLM1_WIDTH         7
#define ADC_CLM1_CLM1(x) (((uint32_t)((uint32_t)(x))<<ADC_CLM1_CLM1_SHIFT))&ADC_CLM1_CLM1_MASK)

```

```

/* CLM0 Bit Fields */
#define ADC_CLM0_CLM0_MASK          0x3Fu
#define ADC_CLM0_CLM0_SHIFT         0
#define ADC_CLM0_CLM0_WIDTH          6
#define ADC_CLM0_CLM0(x)            (((uint32_t)((uint32_t)(x))<<ADC_CLM0_CLM0_SHIFT))&ADC_CLM0_CLM0_MASK)

/*!
 * @}
 */ /* end of group ADC_Register_Masks */

/* ADC - Peripheral instance base addresses */
/** Peripheral ADC0 base address */
#define ADC0_BASE                   (0x4003B000u)
/** Peripheral ADC0 base pointer */
#define ADC0                         ((ADC_Type *)ADC0_BASE)
#define ADC0_BASE_PTR                (ADC0)
/** Array initializer of ADC peripheral base addresses */
#define ADC_BASE_ADDRS               { ADC0_BASE }
/** Array initializer of ADC peripheral base pointers */
#define ADC_BASE_PTRS                { ADC0 }
/** Interrupt vectors for the ADC peripheral type */
#define ADC IRQs                     { ADC0_IRQn }

/* -----
-- ADC - Register accessor macros
-----
*/
/*!
 * @addtogroup ADC_Register_Accessor_Macros ADC - Register accessor
macros
 * @{
 */

/* ADC - Register instance definitions */
/* ADC0 */
#define ADC0_SC1A                  ADC_SC1_REG(ADC0,0)
#define ADC0_SC1B                  ADC_SC1_REG(ADC0,1)
#define ADC0_CFG1                  ADC_CFG1_REG(ADC0)
#define ADC0_CFG2                  ADC_CFG2_REG(ADC0)
#define ADC0_RA                    ADC_R_REG(ADC0,0)
#define ADC0_RB                    ADC_R_REG(ADC0,1)
#define ADC0_CV1                  ADC_CV1_REG(ADC0)
#define ADC0_CV2                  ADC_CV2_REG(ADC0)
#define ADC0_SC2                  ADC_SC2_REG(ADC0)
#define ADC0_SC3                  ADC_SC3_REG(ADC0)
#define ADC0_OFS                  ADC_OFS_REG(ADC0)
#define ADC0_PG                   ADC_PG_REG(ADC0)
#define ADC0_MG                   ADC_MG_REG(ADC0)
#define ADC0_CLPD                 ADC_CLPD_REG(ADC0)

```

```

#define ADC0_CLPS          ADC_CLPS_REG(ADC0)
#define ADC0_CLP4          ADC_CLP4_REG(ADC0)
#define ADC0_CLP3          ADC_CLP3_REG(ADC0)
#define ADC0_CLP2          ADC_CLP2_REG(ADC0)
#define ADC0_CLP1          ADC_CLP1_REG(ADC0)
#define ADC0_CLP0          ADC_CLP0_REG(ADC0)
#define ADC0_CLMD          ADC_CLMD_REG(ADC0)
#define ADC0_CLMS          ADC_CLMS_REG(ADC0)
#define ADC0_CLM4          ADC_CLM4_REG(ADC0)
#define ADC0_CLM3          ADC_CLM3_REG(ADC0)
#define ADC0_CLM2          ADC_CLM2_REG(ADC0)
#define ADC0_CLM1          ADC_CLM1_REG(ADC0)
#define ADC0_CLM0          ADC_CLM0_REG(ADC0)

/* ADC - Register array accessors */
#define ADC0_SC1(index)      ADC_SC1_REG(ADC0, index)
#define ADC0_R(index)        ADC_R_REG(ADC0, index)

/*!
 * @}
 */ /* end of group ADC_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group ADC_Peripheral_Access_Layer */

/*
-----  

-- CMP Peripheral Access Layer  

-----  

*/
/*!  

 * @addtogroup CMP_Peripheral_Access_Layer CMP Peripheral Access Layer  

 * @{
 */

/** CMP - Register Layout Typedef */
typedef struct {
    __IO uint8_t CR0;                      /**< CMP Control  

Register 0, offset: 0x0 */
    __IO uint8_t CR1;                      /**< CMP Control  

Register 1, offset: 0x1 */
    __IO uint8_t FPR;                      /**< CMP Filter Period  

Register, offset: 0x2 */
    __IO uint8_t SCR;                      /**< CMP Status and  

Control Register, offset: 0x3 */
    __IO uint8_t DACCRL;                   /**< DAC Control  

Register, offset: 0x4 */
    __IO uint8_t MUXCR;                   /**< MUX Control  

Register, offset: 0x5 */
} CMP_Type, *CMP_MemMapPtr;

```

```

/* -----
-- CMP - Register accessor macros
-----
*/
/* !
 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor
macros
 * @{
 */

/* CMP - Register accessors */
#define CMP_CR0_REG(base) ((base) -> CR0)
#define CMP_CR1_REG(base) ((base) -> CR1)
#define CMP_FPR_REG(base) ((base) -> FPR)
#define CMP_SCR_REG(base) ((base) -> SCR)
#define CMP_DACCR_REG(base) ((base) -> DACCR)
#define CMP_MUXCR_REG(base) ((base) -> MUXCR)

/* !
 * @}
 */
/* /* end of group CMP_Register_Accessor_Macros */

/* -----
-- CMP Register Masks
-----
*/
/* !
 * @addtogroup CMP_Register_Masks CMP Register Masks
 * @{
 */

/* CR0 Bit Fields */
#define CMP_CR0_HYSTCTR_MASK 0x3u
#define CMP_CR0_HYSTCTR_SHIFT 0
#define CMP_CR0_HYSTCTR_WIDTH 2
#define CMP_CR0_HYSTCTR(x) (((uint8_t)((uint8_t)(x)) << CMP_CR0_HYSTCTR_SHIFT) & CMP_CR0_HYSTCTR_MASK)
#define CMP_CR0_FILTER_CNT_MASK 0x70u
#define CMP_CR0_FILTER_CNT_SHIFT 4
#define CMP_CR0_FILTER_CNT_WIDTH 3
#define CMP_CR0_FILTER_CNT(x) (((uint8_t)((uint8_t)(x)) << CMP_CR0_FILTER_CNT_SHIFT) & CMP_CR0_FILTER_CNT_MASK)
/* CR1 Bit Fields */
#define CMP_CR1_EN_MASK 0x1u
#define CMP_CR1_EN_SHIFT 0
#define CMP_CR1_EN_WIDTH 1

```

```

#define CMP_CR1_EN(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_EN_SHIFT)) & CMP_CR1_EN_MASK)
#define CMP_CR1_OPE_MASK 0x2u
#define CMP_CR1_OPE_SHIFT 1
#define CMP_CR1_OPE_WIDTH 1
#define CMP_CR1_OPE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_OPE_SHIFT)) & CMP_CR1_OPE_MASK)
#define CMP_CR1_COS_MASK 0x4u
#define CMP_CR1_COS_SHIFT 2
#define CMP_CR1_COS_WIDTH 1
#define CMP_CR1_COS(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_COS_SHIFT)) & CMP_CR1_COS_MASK)
#define CMP_CR1_INV_MASK 0x8u
#define CMP_CR1_INV_SHIFT 3
#define CMP_CR1_INV_WIDTH 1
#define CMP_CR1_INV(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_INV_SHIFT)) & CMP_CR1_INV_MASK)
#define CMP_CR1_PMODE_MASK 0x10u
#define CMP_CR1_PMODE_SHIFT 4
#define CMP_CR1_PMODE_WIDTH 1
#define CMP_CR1_PMODE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_PMODE_SHIFT)) & CMP_CR1_PMODE_MASK)
#define CMP_CR1_TRIGM_MASK 0x20u
#define CMP_CR1_TRIGM_SHIFT 5
#define CMP_CR1_TRIGM_WIDTH 1
#define CMP_CR1_TRIGM(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_TRIGM_SHIFT)) & CMP_CR1_TRIGM_MASK)
#define CMP_CR1_WE_MASK 0x40u
#define CMP_CR1_WE_SHIFT 6
#define CMP_CR1_WE_WIDTH 1
#define CMP_CR1_WE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_WE_SHIFT)) & CMP_CR1_WE_MASK)
#define CMP_CR1_SE_MASK 0x80u
#define CMP_CR1_SE_SHIFT 7
#define CMP_CR1_SE_WIDTH 1
#define CMP_CR1_SE(x)
(((uint8_t) (((uint8_t)(x))<<CMP_CR1_SE_SHIFT)) & CMP_CR1_SE_MASK)
/* FPR Bit Fields */
#define CMP_FPR_FILT_PER_MASK 0xFFu
#define CMP_FPR_FILT_PER_SHIFT 0
#define CMP_FPR_FILT_PER_WIDTH 8
#define CMP_FPR_FILT_PER(x)
(((uint8_t) (((uint8_t)(x))<<CMP_FPR_FILT_PER_SHIFT)) & CMP_FPR_FILT_PER_MASK)
/* SCR Bit Fields */
#define CMP_SCR_COUT_MASK 0x1u
#define CMP_SCR_COUT_SHIFT 0
#define CMP_SCR_COUT_WIDTH 1
#define CMP_SCR_COUT(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_COUT_SHIFT)) & CMP_SCR_COUT_MASK)
#define CMP_SCR_CFF_MASK 0x2u
#define CMP_SCR_CFF_SHIFT 1
#define CMP_SCR_CFF_WIDTH 1

```

```

#define CMP_SCR_CFF(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_CFF_SHIFT)) & CMP_SCR_CFF_MASK)
#define CMP_SCR_CFR_MASK 0x4u
#define CMP_SCR_CFR_SHIFT 2
#define CMP_SCR_CFR_WIDTH 1
#define CMP_SCR_CFR(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_CFR_SHIFT)) & CMP_SCR_CFR_MASK)
#define CMP_SCR_IEF_MASK 0x8u
#define CMP_SCR_IEF_SHIFT 3
#define CMP_SCR_IEF_WIDTH 1
#define CMP_SCR_IEF(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_IEF_SHIFT)) & CMP_SCR_IEF_MASK)
#define CMP_SCR_IER_MASK 0x10u
#define CMP_SCR_IER_SHIFT 4
#define CMP_SCR_IER_WIDTH 1
#define CMP_SCR_IER(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_IER_SHIFT)) & CMP_SCR_IER_MASK)
#define CMP_SCR_DMAEN_MASK 0x40u
#define CMP_SCR_DMAEN_SHIFT 6
#define CMP_SCR_DMAEN_WIDTH 1
#define CMP_SCR_DMAEN(x)
(((uint8_t) (((uint8_t)(x))<<CMP_SCR_DMAEN_SHIFT)) & CMP_SCR_DMAEN_MASK)
/* DACCR Bit Fields */
#define CMP_DACCR_VOSEL_MASK 0x3Fu
#define CMP_DACCR_VOSEL_SHIFT 0
#define CMP_DACCR_VOSEL_WIDTH 6
#define CMP_DACCR_VOSEL(x)
(((uint8_t) (((uint8_t)(x))<<CMP_DACCR_VOSEL_SHIFT)) & CMP_DACCR_VOSEL_MASK)
#define CMP_DACCR_VRSEL_MASK 0x40u
#define CMP_DACCR_VRSEL_SHIFT 6
#define CMP_DACCR_VRSEL_WIDTH 1
#define CMP_DACCR_VRSEL(x)
(((uint8_t) (((uint8_t)(x))<<CMP_DACCR_VRSEL_SHIFT)) & CMP_DACCR_VRSEL_MASK)
#define CMP_DACCR_DACEN_MASK 0x80u
#define CMP_DACCR_DACEN_SHIFT 7
#define CMP_DACCR_DACEN_WIDTH 1
#define CMP_DACCR_DACEN(x)
(((uint8_t) (((uint8_t)(x))<<CMP_DACCR_DACEN_SHIFT)) & CMP_DACCR_DACEN_MASK)
/* MUXCR Bit Fields */
#define CMP_MUXCR_MSEL_MASK 0x7u
#define CMP_MUXCR_MSEL_SHIFT 0
#define CMP_MUXCR_MSEL_WIDTH 3
#define CMP_MUXCR_MSEL(x)
(((uint8_t) (((uint8_t)(x))<<CMP_MUXCR_MSEL_SHIFT)) & CMP_MUXCR_MSEL_MASK)
#define CMP_MUXCR_PSEL_MASK 0x38u
#define CMP_MUXCR_PSEL_SHIFT 3
#define CMP_MUXCR_PSEL_WIDTH 3
#define CMP_MUXCR_PSEL(x)
(((uint8_t) (((uint8_t)(x))<<CMP_MUXCR_PSEL_SHIFT)) & CMP_MUXCR_PSEL_MASK)
#define CMP_MUXCR_PSTM_MASK 0x80u
#define CMP_MUXCR_PSTM_SHIFT 7
#define CMP_MUXCR_PSTM_WIDTH 1
#define CMP_MUXCR_PSTM(x)
(((uint8_t) (((uint8_t)(x))<<CMP_MUXCR_PSTM_SHIFT)) & CMP_MUXCR_PSTM_MASK)

```

```

/*!
 * @}
 */
/* end of group CMP_Register_Masks */

/* CMP - Peripheral instance base addresses */
/** Peripheral CMP0 base address */
#define CMP0_BASE (0x40073000u)
/** Peripheral CMP0 base pointer */
#define CMP0 ((CMP_Type *) CMP0_BASE)
#define CMP0_PTR (CMP0)
/** Array initializer of CMP peripheral base addresses */
#define CMP_BASE_ADDRS { CMP0_BASE }
/** Array initializer of CMP peripheral base pointers */
#define CMP_BASE_PTRS { CMP0 }
/** Interrupt vectors for the CMP peripheral type */
#define CMP IRQS { CMP0_IRQn }

/*
-----
-- CMP - Register accessor macros
-----
*/
/*!
 * @addtogroup CMP_Register_Accessor_Macros CMP - Register accessor
macros
 * @{
 */
/* CMP - Register instance definitions */
/* CMP0 */
#define CMP0_CRO CMP_CRO_REG(CMP0)
#define CMP0_CR1 CMP_CR1_REG(CMP0)
#define CMP0_FPR CMP_FPR_REG(CMP0)
#define CMP0_SCR CMP_SCR_REG(CMP0)
#define CMP0_DACCR CMP_DACCR_REG(CMP0)
#define CMP0_MUXCR CMP_MUXCR_REG(CMP0)

/*!
 * @}
 */
/* end of group CMP_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group CMP_Peripheral_Access_Layer */

/*
-----
-- DAC Peripheral Access Layer
-----

```

```

----- */
/*!
 * @addtogroup DAC_Peripheral_Access_Layer DAC Peripheral Access Layer
 * @{
 */

/** DAC - Register Layout Typedef */
typedef struct {
    struct {                                         /* offset: 0x0, array
step: 0x2 */
        __IO uint8_t DATL;                           /**< DAC Data Low
Register, array offset: 0x0, array step: 0x2 */
        __IO uint8_t DATH;                           /**< DAC Data High
Register, array offset: 0x1, array step: 0x2 */
    } DAT[2];
    uint8_t RESERVED_0[28];                         /**< DAC Status
__IO uint8_t SR;                                 /**< DAC Control
Register, offset: 0x20 */
    __IO uint8_t C0;                               /**< DAC Control
Register, offset: 0x21 */
    __IO uint8_t C1;                               /**< DAC Control
Register 1, offset: 0x22 */
    __IO uint8_t C2;                               /**< DAC Control
Register 2, offset: 0x23 */
} DAC_Type, *DAC_MemMapPtr;

/*
-----
-- DAC - Register accessor macros
-----
*/
/*!
 * @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor
macros
 * @{
 */

/* DAC - Register accessors */
#define DAC_DATL_REG(base, index)          ((base) ->DAT[index].DATL)
#define DAC_DATL_COUNT                     2
#define DAC_DATH_REG(base, index)          ((base) ->DAT[index].DATH)
#define DAC_DATH_COUNT                     2
#define DAC_SR_REG(base)                  ((base) ->SR)
#define DAC_C0_REG(base)                  ((base) ->C0)
#define DAC_C1_REG(base)                  ((base) ->C1)
#define DAC_C2_REG(base)                  ((base) ->C2)

*/

```

```

        * @}
    /*/* end of group DAC_Register_Accessor_Macros */

/*
-----*
----- DAC Register Masks
----- */
/* !
 * @addtogroup DAC_Register_Masks DAC Register Masks
 * @{
 */

/* DATL Bit Fields */
#define DAC_DATL_DATA0_MASK          0xFFu
#define DAC_DATL_DATA0_SHIFT         0
#define DAC_DATL_DATA0_WIDTH         8
#define DAC_DATL_DATA0(x) (((uint8_t)((uint8_t)(x))<<DAC_DATL_DATA0_SHIFT))&DAC_DATL_DATA0_MASK
/* DATH Bit Fields */
#define DAC_DATH_DATA1_MASK          0xFu
#define DAC_DATH_DATA1_SHIFT         0
#define DAC_DATH_DATA1_WIDTH         4
#define DAC_DATH_DATA1(x) (((uint8_t)((uint8_t)(x))<<DAC_DATH_DATA1_SHIFT))&DAC_DATH_DATA1_MASK
/* SR Bit Fields */
#define DAC_SR_DACBFRPBF_MASK         0x1u
#define DAC_SR_DACBFRPBF_SHIFT        0
#define DAC_SR_DACBFRPBF_WIDTH        1
#define DAC_SR_DACBFRPBF(x) (((uint8_t)((uint8_t)(x))<<DAC_SR_DACBFRPBF_SHIFT))&DAC_SR_DACBFRPBF_MASK
#define DAC_SR_DACBFRPTF_MASK         0x2u
#define DAC_SR_DACBFRPTF_SHIFT        1
#define DAC_SR_DACBFRPTF_WIDTH        1
#define DAC_SR_DACBFRPTF(x) (((uint8_t)((uint8_t)(x))<<DAC_SR_DACBFRPTF_SHIFT))&DAC_SR_DACBFRPTF_MASK
/* C0 Bit Fields */
#define DAC_C0_DACBBIEN_MASK          0x1u
#define DAC_C0_DACBBIEN_SHIFT         0
#define DAC_C0_DACBBIEN_WIDTH         1
#define DAC_C0_DACBBIEN(x) (((uint8_t)((uint8_t)(x))<<DAC_C0_DACBBIEN_SHIFT))&DAC_C0_DACBBIEN_MASK
#define DAC_C0_DACBTIEN_MASK          0x2u
#define DAC_C0_DACBTIEN_SHIFT         1
#define DAC_C0_DACBTIEN_WIDTH         1
#define DAC_C0_DACBTIEN(x) (((uint8_t)((uint8_t)(x))<<DAC_C0_DACBTIEN_SHIFT))&DAC_C0_DACBTIEN_MASK
#define DAC_C0_LPEN_MASK              0x8u
#define DAC_C0_LPEN_SHIFT             3
#define DAC_C0_LPEN_WIDTH             1

```

```

#define DAC_C0_LPEN(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C0_LPEN_SHIFT)) & DAC_C0_LPEN_MASK)
#define DAC_C0_DACSWTRG_MASK 0x10u
#define DAC_C0_DACSWTRG_SHIFT 4
#define DAC_C0_DACSWTRG_WIDTH 1
#define DAC_C0_DACSWTRG(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C0_DACSWTRG_SHIFT)) & DAC_C0_DACSWTRG_MASK)
#define DAC_C0_DACTRGSEL_MASK 0x20u
#define DAC_C0_DACTRGSEL_SHIFT 5
#define DAC_C0_DACTRGSEL_WIDTH 1
#define DAC_C0_DACTRGSEL(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C0_DACTRGSEL_SHIFT)) & DAC_C0_DACTRGSEL_MASK)
#define DAC_C0_DACRFS_MASK 0x40u
#define DAC_C0_DACRFS_SHIFT 6
#define DAC_C0_DACRFS_WIDTH 1
#define DAC_C0_DACRFS(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C0_DACRFS_SHIFT)) & DAC_C0_DACRFS_MASK)
#define DAC_C0_DACEN_MASK 0x80u
#define DAC_C0_DACEN_SHIFT 7
#define DAC_C0_DACEN_WIDTH 1
#define DAC_C0_DACEN(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C0_DACEN_SHIFT)) & DAC_C0_DACEN_MASK)
/* C1 Bit Fields */
#define DAC_C1_DACBFEN_MASK 0x1u
#define DAC_C1_DACBFEN_SHIFT 0
#define DAC_C1_DACBFEN_WIDTH 1
#define DAC_C1_DACBFEN(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C1_DACBFEN_SHIFT)) & DAC_C1_DACBFEN_MASK)
#define DAC_C1_DACBFMD_MASK 0x4u
#define DAC_C1_DACBFMD_SHIFT 2
#define DAC_C1_DACBFMD_WIDTH 1
#define DAC_C1_DACBFMD(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C1_DACBFMD_SHIFT)) & DAC_C1_DACBFMD_MASK)
#define DAC_C1_DMAEN_MASK 0x80u
#define DAC_C1_DMAEN_SHIFT 7
#define DAC_C1_DMAEN_WIDTH 1
#define DAC_C1_DMAEN(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C1_DMAEN_SHIFT)) & DAC_C1_DMAEN_MASK)
/* C2 Bit Fields */
#define DAC_C2_DACBFUP_MASK 0x1u
#define DAC_C2_DACBFUP_SHIFT 0
#define DAC_C2_DACBFUP_WIDTH 1
#define DAC_C2_DACBFUP(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C2_DACBFUP_SHIFT)) & DAC_C2_DACBFUP_MASK)
#define DAC_C2_DACBFRP_MASK 0x10u
#define DAC_C2_DACBFRP_SHIFT 4
#define DAC_C2_DACBFRP_WIDTH 1
#define DAC_C2_DACBFRP(x)
(((uint8_t) (((uint8_t)(x))<<DAC_C2_DACBFRP_SHIFT)) & DAC_C2_DACBFRP_MASK)

/* !
 * @}
 */ /* end of group DAC_Register_Masks */

```

```

/* DAC - Peripheral instance base addresses */
/** Peripheral DAC0 base address */
#define DAC0_BASE (0x4003F000u)
/** Peripheral DAC0 base pointer */
#define DAC0 ((DAC_Type *)DAC0_BASE)
#define DAC0_BASE_PTR (DAC0)
/** Array initializer of DAC peripheral base addresses */
#define DAC_BASE_ADDRS { DAC0 }
/** Array initializer of DAC peripheral base pointers */
#define DAC_BASE_PTRS { DAC0 }
/** Interrupt vectors for the DAC peripheral type */
#define DAC IRQS { DAC0_IRQn }

/* -----
-- DAC - Register accessor macros
-----
----- */

/*!
 * @addtogroup DAC_Register_Accessor_Macros DAC - Register accessor
macros
 * @{
 */

/* DAC - Register instance definitions */
/* DAC0 */
#define DAC0_DAT0L DAC_DATL_REG(DAC0, 0)
#define DAC0_DAT0H DAC_DATH_REG(DAC0, 0)
#define DAC0_DAT1L DAC_DATL_REG(DAC0, 1)
#define DAC0_DAT1H DAC_DATH_REG(DAC0, 1)
#define DAC0_SR DAC_SR_REG(DAC0)
#define DAC0_C0 DAC_C0_REG(DAC0)
#define DAC0_C1 DAC_C1_REG(DAC0)
#define DAC0_C2 DAC_C2_REG(DAC0)

/* DAC - Register array accessors */
#define DAC0_DATL(index) DAC_DATL_REG(DAC0, index)
#define DAC0_DATH(index) DAC_DATH_REG(DAC0, index)

/*!
 * @}
 */
/* end of group DAC_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group DAC_Peripheral_Access_Layer */

```

```

/*
-----+
-- DMA Peripheral Access Layer
-----+
----- */

/*!
 * @addtogroup DMA_Peripheral_Access_Layer DMA Peripheral Access Layer
 * @{
 */

/** DMA - Register Layout Typedef */
typedef struct {
    uint8_t RESERVED_0[256];
    struct {
        array step: 0x10 /* offset: 0x100,
        _IO uint32_t SAR; /***< Source Address
        Register, array offset: 0x100, array step: 0x10 */
        _IO uint32_t DAR; /***< Destination
        Address Register, array offset: 0x104, array step: 0x10 */
        union { /* offset: 0x108,
        array step: 0x10 */
            _IO uint32_t DSR_BCR; /***< DMA Status
            Register / Byte Count Register, array offset: 0x108, array step: 0x10 */
            struct { /* offset: 0x108,
            array step: 0x10 */
                uint8_t RESERVED_0[3];
                _IO uint8_t DSR; /***< DMA_DSR0
                register...DMA_DSR3 register., array offset: 0x10B, array step: 0x10 */
            } DMA_DSR_ACCESS8BIT;
        };
        _IO uint32_t DCR; /***< DMA Control
        Register, array offset: 0x10C, array step: 0x10 */
    } DMA[4];
} DMA_Type, *DMA_MemMapPtr;

/*
-----+
-- DMA - Register accessor macros
-----+
----- */

/*!
 * @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor
 * macros
 * @{
 */

/* DMA - Register accessors */
#define DMA_SAR_REG(base, index) ((base)->DMA[index].SAR)
#define DMA_SAR_COUNT 4
#define DMA_DAR_REG(base, index) ((base)->DMA[index].DAR)
#define DMA_DAR_COUNT 4

```

```

#define DMA_DSR_BCR_REG(base, index)          ((base) -  

>DMA[index].DSR_BCR)  

#define DMA_DSR_BCR_COUNT                   4  

#define DMA_DSR_REG(base, index)             ((base) -  

>DMA[index].DMA_DSR_ACCESS8BIT.DSR)  

#define DMA_DSR_COUNT                      4  

#define DMA_DCR_REG(base, index)             ((base) ->DMA[index].DCR)  

#define DMA_DCR_COUNT                      4

/*!  

 * @}  

 */ /* end of group DMA_Register_Accessor_Macros */

/*
-----  

-- DMA Register Masks  

-----  

-- */

/*!  

 * @addtogroup DMA_Register_Masks DMA Register Masks
 * @{
 */

/* SAR Bit Fields */
#define DMA_SAR_SAR_MASK                    0xFFFFFFFFu
#define DMA_SAR_SAR_SHIFT                  0
#define DMA_SAR_SAR_WIDTH                 32
#define DMA_SAR_SAR(x)  

(((uint32_t)((uint32_t)(x))<<DMA_SAR_SAR_SHIFT))&DMA_SAR_SAR_MASK)

/* DAR Bit Fields */
#define DMA_DAR_DAR_MASK                  0xFFFFFFFFu
#define DMA_DAR_DAR_SHIFT                0
#define DMA_DAR_DAR_WIDTH               32
#define DMA_DAR_DAR(x)  

(((uint32_t)((uint32_t)(x))<<DMA_DAR_DAR_SHIFT))&DMA_DAR_DAR_MASK

/* DSR_BCR Bit Fields */
#define DMA_DSR_BCR_BCR_MASK              0xFFFFFFFFu
#define DMA_DSR_BCR_BCR_SHIFT             0
#define DMA_DSR_BCR_BCR_WIDTH            24
#define DMA_DSR_BCR_BCR(x)  

(((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_BCR_SHIFT))&DMA_DSR_BCR_BCR_MASK

#define DMA_DSR_BCR_DONE_MASK             0x1000000u
#define DMA_DSR_BCR_DONE_SHIFT            24
#define DMA_DSR_BCR_DONE_WIDTH           1
#define DMA_DSR_BCR_DONE(x)  

(((uint32_t)((uint32_t)(x))<<DMA_DSR_BCR_DONE_SHIFT))&DMA_DSR_BCR_DONE_MASK

#define DMA_DSR_BCR_BSY_MASK              0x2000000u
#define DMA_DSR_BCR_BSY_SHIFT             25
#define DMA_DSR_BCR_BSY_WIDTH            1

```

```

#define DMA_DSR_BCR_BSY(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DSR_BCR_BSY_SHIFT))&DMA_DSR_BCR_BSY_MASK)
#define DMA_DSR_BCR_REQ_MASK 0x4000000u
#define DMA_DSR_BCR_REQ_SHIFT 26
#define DMA_DSR_BCR_REQ_WIDTH 1
#define DMA_DSR_BCR_REQ(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DSR_BCR_REQ_SHIFT))&DMA_DSR_BCR_REQ_MASK)
#define DMA_DSR_BCR_BED_MASK 0x10000000u
#define DMA_DSR_BCR_BED_SHIFT 28
#define DMA_DSR_BCR_BED_WIDTH 1
#define DMA_DSR_BCR_BED(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DSR_BCR_BED_SHIFT))&DMA_DSR_BCR_BED_MASK)
#define DMA_DSR_BCR_BES_MASK 0x20000000u
#define DMA_DSR_BCR_BES_SHIFT 29
#define DMA_DSR_BCR_BES_WIDTH 1
#define DMA_DSR_BCR_BES(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DSR_BCR_BES_SHIFT))&DMA_DSR_BCR_BES_MASK)
#define DMA_DSR_BCR_CE_MASK 0x40000000u
#define DMA_DSR_BCR_CE_SHIFT 30
#define DMA_DSR_BCR_CE_WIDTH 1
#define DMA_DSR_BCR_CE(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DSR_BCR_CE_SHIFT))&DMA_DSR_BCR_CE_MASK)
/* DCR Bit Fields */
#define DMA_DCR_LCH2_MASK 0x3u
#define DMA_DCR_LCH2_SHIFT 0
#define DMA_DCR_LCH2_WIDTH 2
#define DMA_DCR_LCH2(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DCR_LCH2_SHIFT))&DMA_DCR_LCH2_MASK)
#define DMA_DCR_LCH1_MASK 0xCu
#define DMA_DCR_LCH1_SHIFT 2
#define DMA_DCR_LCH1_WIDTH 2
#define DMA_DCR_LCH1(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DCR_LCH1_SHIFT))&DMA_DCR_LCH1_MASK)
#define DMA_DCR_LINKCC_MASK 0x30u
#define DMA_DCR_LINKCC_SHIFT 4
#define DMA_DCR_LINKCC_WIDTH 2
#define DMA_DCR_LINKCC(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DCR_LINKCC_SHIFT))&DMA_DCR_LINKCC_MASK)
#define DMA_DCR_D_REQ_MASK 0x80u
#define DMA_DCR_D_REQ_SHIFT 7
#define DMA_DCR_D_REQ_WIDTH 1
#define DMA_DCR_D_REQ(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DCR_D_REQ_SHIFT))&DMA_DCR_D_REQ_MASK)
#define DMA_DCR_DMOD_MASK 0xF00u
#define DMA_DCR_DMOD_SHIFT 8
#define DMA_DCR_DMOD_WIDTH 4
#define DMA_DCR_DMOD(x)
(((uint32_t) (((uint32_t)(x))<<DMA_DCR_DMOD_SHIFT))&DMA_DCR_DMOD_MASK)
#define DMA_DCR_SMOD_MASK 0xF000u
#define DMA_DCR_SMOD_SHIFT 12

```

```

#define DMA_DCR_SMOD_WIDTH 4
#define DMA_DCR_SMOD(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_SMOD_SHIFT))&DMA_DCR_SMOD_MASK)
#define DMA_DCR_START_MASK 0x10000u
#define DMA_DCR_START_SHIFT 16
#define DMA_DCR_START_WIDTH 1
#define DMA_DCR_START(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_START_SHIFT))&DMA_DCR_START_MASK)
#define DMA_DCR_DSIZE_MASK 0x60000u
#define DMA_DCR_DSIZE_SHIFT 17
#define DMA_DCR_DSIZE_WIDTH 2
#define DMA_DCR_DSIZE(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_DSIZE_SHIFT))&DMA_DCR_DSIZE_MASK)
#define DMA_DCR_DINC_MASK 0x80000u
#define DMA_DCR_DINC_SHIFT 19
#define DMA_DCR_DINC_WIDTH 1
#define DMA_DCR_DINC(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_DINC_SHIFT))&DMA_DCR_DINC_MASK)
#define DMA_DCR_SSIZE_MASK 0x300000u
#define DMA_DCR_SSIZE_SHIFT 20
#define DMA_DCR_SSIZE_WIDTH 2
#define DMA_DCR_SSIZE(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_SSIZE_SHIFT))&DMA_DCR_SSIZE_MASK)
#define DMA_DCR_SINC_MASK 0x400000u
#define DMA_DCR_SINC_SHIFT 22
#define DMA_DCR_SINC_WIDTH 1
#define DMA_DCR_SINC(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_SINC_SHIFT))&DMA_DCR_SINC_MASK)
#define DMA_DCR_EADREQ_MASK 0x800000u
#define DMA_DCR_EADREQ_SHIFT 23
#define DMA_DCR_EADREQ_WIDTH 1
#define DMA_DCR_EADREQ(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_EADREQ_SHIFT))&DMA_DCR_EADREQ_MASK)
#define DMA_DCR_AA_MASK 0x10000000u
#define DMA_DCR_AA_SHIFT 28
#define DMA_DCR_AA_WIDTH 1
#define DMA_DCR_AA(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_AA_SHIFT))&DMA_DCR_AA_MASK)
#define DMA_DCR_CS_MASK 0x20000000u
#define DMA_DCR_CS_SHIFT 29
#define DMA_DCR_CS_WIDTH 1
#define DMA_DCR_CS(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_CS_SHIFT))&DMA_DCR_CS_MASK)
#define DMA_DCR_ERQ_MASK 0x40000000u
#define DMA_DCR_ERQ_SHIFT 30
#define DMA_DCR_ERQ_WIDTH 1
#define DMA_DCR_ERQ(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_ERQ_SHIFT))&DMA_DCR_ERQ_MASK)
#define DMA_DCR_EINT_MASK 0x80000000u
#define DMA_DCR_EINT_SHIFT 31
#define DMA_DCR_EINT_WIDTH 1
#define DMA_DCR_EINT(x) (((uint32_t)((uint32_t)(x))<<DMA_DCR_EINT_SHIFT))&DMA_DCR_EINT_MASK)

```

```

/* !
 * @}
 */
/* / * end of group DMA_Register_Masks */

/* DMA - Peripheral instance base addresses */
/** Peripheral DMA base address */
#define DMA_BASE (0x40008000u)
/** Peripheral DMA base pointer */
#define DMA0 ((DMA_Type *) DMA_BASE)
#define DMA_BASE_PTR (DMA0)
/** Array initializer of DMA peripheral base addresses */
#define DMA_BASE_ADDRS { DMA_BASE }
/** Array initializer of DMA peripheral base pointers */
#define DMA_BASE_PTRS { DMA0 }
/** Interrupt vectors for the DMA peripheral type */
#define DMA_CHN IRQS { DMA0_IRQn, DMA1_IRQn,
DMA2_IRQn, DMA3_IRQn }

/* -----
-- DMA - Register accessor macros
-----
*/
/* !
 * @addtogroup DMA_Register_Accessor_Macros DMA - Register accessor
macros
 * @{
 */
/* DMA - Register instance definitions */
/* DMA */
#define DMA_SAR0 DMA_SAR_REG(DMA0, 0)
#define DMA_DAR0 DMA_DAR_REG(DMA0, 0)
#define DMA_DSR_BCR0 DMA_DSR_BCR_REG(DMA0, 0)
#define DMA_DSR0 DMA_DSR_REG(DMA0, 0)
#define DMA_DCR0 DMA_DCR_REG(DMA0, 0)
#define DMA_SAR1 DMA_SAR_REG(DMA0, 1)
#define DMA_DAR1 DMA_DAR_REG(DMA0, 1)
#define DMA_DSR_BCR1 DMA_DSR_BCR_REG(DMA0, 1)
#define DMA_DSR1 DMA_DSR_REG(DMA0, 1)
#define DMA_DCR1 DMA_DCR_REG(DMA0, 1)
#define DMA_SAR2 DMA_SAR_REG(DMA0, 2)
#define DMA_DAR2 DMA_DAR_REG(DMA0, 2)
#define DMA_DSR_BCR2 DMA_DSR_BCR_REG(DMA0, 2)
#define DMA_DSR2 DMA_DSR_REG(DMA0, 2)
#define DMA_DCR2 DMA_DCR_REG(DMA0, 2)
#define DMA_SAR3 DMA_SAR_REG(DMA0, 3)
#define DMA_DAR3 DMA_DAR_REG(DMA0, 3)
#define DMA_DSR_BCR3 DMA_DSR_BCR_REG(DMA0, 3)
#define DMA_DSR3 DMA_DSR_REG(DMA0, 3)
#define DMA_DCR3 DMA_DCR_REG(DMA0, 3)

```

```

/* DMA - Register array accessors */
#define DMA_SAR(index) DMA_SAR_REG(DMA0,index)
#define DMA_DAR(index) DMA_DAR_REG(DMA0,index)
#define DMA_DSR_BCR(index) DMA_DSR_BCR_REG(DMA0,index)
#define DMA_DSR(index) DMA_DSR_REG(DMA0,index)
#define DMA_DCR(index) DMA_DCR_REG(DMA0,index)

/* !
 * @}
 */ /* end of group DMA_Register_Accessor_Macros */

/* !
 * @}
 */ /* end of group DMA_Peripheral_Access_Layer */

/* -----
-- DMAMUX Peripheral Access Layer
-----
*/
/* !
 * @addtogroup DMAMUX_Peripheral_Access_Layer DMAMUX Peripheral Access
Layer
 * @{
 */

/** DMAMUX - Register Layout Typedef */
typedef struct {
    __IO uint8_t CHCFG[4]; /* < Channel Configuration register, array offset: 0x0, array step: 0x1 */
} DMAMUX_Type, *DMAMUX_MemMapPtr;

/* -----
-- DMAMUX - Register accessor macros
-----
*/
/* !
 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor
macros
 * @{
 */

/* DMAMUX - Register accessors */
#define DMAMUX_CHCFG_REG(base, index) ((base) -> CHCFG[index])
#define DMAMUX_CHCFG_COUNT 4

```

```

/* !
 * @}
 */
/* /* end of group DMAMUX_Register_Accessor_Macros */

/*
-----  

-- DMAMUX Register Masks  

-----  

*/
/* !
 * @addtogroup DMAMUX_Register_Masks DMAMUX Register Masks
 * @{
 */
/* CHCFG Bit Fields */
#define DMAMUX_CHCFG_SOURCE_MASK          0x3Fu
#define DMAMUX_CHCFG_SOURCE_SHIFT         0
#define DMAMUX_CHCFG_SOURCE_WIDTH         6
#define DMAMUX_CHCFG_SOURCE(x)            (((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_SOURCE_SHIFT))&DMAMUX_CHCFG_SOURCE_MASK
#define DMAMUX_CHCFG_TRIG_MASK           0x40u
#define DMAMUX_CHCFG_TRIG_SHIFT          6
#define DMAMUX_CHCFG_TRIG_WIDTH          1
#define DMAMUX_CHCFG_TRIG(x)             (((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_TRIG_SHIFT))&DMAMUX_CHCFG_TRIG_MASK
#define DMAMUX_CHCFG_ENBL_MASK           0x80u
#define DMAMUX_CHCFG_ENBL_SHIFT          7
#define DMAMUX_CHCFG_ENBL_WIDTH          1
#define DMAMUX_CHCFG_ENBL(x)             (((uint8_t)((uint8_t)(x))<<DMAMUX_CHCFG_ENBL_SHIFT))&DMAMUX_CHCFG_ENBL_MASK
/* !
 * @}
 */
/* /* end of group DMAMUX_Register_Masks */

/* DMAMUX - Peripheral instance base addresses */
/** Peripheral DMAMUX0 base address */
#define DMAMUX0_BASE                    (0x40021000u)
/** Peripheral DMAMUX0 base pointer */
#define DMAMUX0                         ((DMAMUX_Type *)DMAMUX0_BASE)
#define DMAMUX0_BASE_PTR                (DMAMUX0)
/** Array initializer of DMAMUX peripheral base addresses */
#define DMAMUX_BASE_ADDRS               { DMAMUX0_BASE }
/** Array initializer of DMAMUX peripheral base pointers */
#define DMAMUX_BASE_PTRS                { DMAMUX0 }

```

```

/* -----
-- DMAMUX - Register accessor macros
-----
*/
/* !
 * @addtogroup DMAMUX_Register_Accessor_Macros DMAMUX - Register accessor
macros
 * @{
 */

/* DMAMUX - Register instance definitions */
/* DMAMUX0 */
#define DMAMUX0_CHCFG0
DMAMUX_CHCFG_REG(DMAMUX0, 0)
#define DMAMUX0_CHCFG1
DMAMUX_CHCFG_REG(DMAMUX0, 1)
#define DMAMUX0_CHCFG2
DMAMUX_CHCFG_REG(DMAMUX0, 2)
#define DMAMUX0_CHCFG3
DMAMUX_CHCFG_REG(DMAMUX0, 3)

/* DMAMUX - Register array accessors */
#define DMAMUX0_CHCFG(index)
DMAMUX_CHCFG_REG(DMAMUX0, index)

/* !
 * @}
 */
/* end of group DMAMUX_Register_Accessor_Macros */

/* !
 * @}
 */
/* end of group DMAMUX_Peripheral_Access_Layer */

/* -----
-- GPIO Peripheral Access Layer
-----
*/
/* !
 * @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access
Layer
 * @{
 */

/** GPIO - Register Layout Typedef */
typedef struct {
    __IO uint32_t PDOR;                                /**< Port Data Output
Register, offset: 0x0 */

```

```

    _O uint32_t PSOR;                                /**< Port Set Output
Register, offset: 0x4 */
    _O uint32_t PCOR;                                /**< Port Clear Output
Register, offset: 0x8 */
    _O uint32_t PTOR;                                /**< Port Toggle
Output Register, offset: 0xC */
    _I uint32_t PDIR;                                /**< Port Data Input
Register, offset: 0x10 */
    _IO uint32_t PDDR;                               /**< Port Data
Direction Register, offset: 0x14 */
} GPIO_Type, *GPIO_MemMapPtr;

/*
-----
-- FGPIO - Register accessor macros
-----
*/
/* !
 * @addtogroup FGPIO_Register_Accessor_Macros FGPIO - Register accessor
macros
 * @{
 */

/* FGPIO - Register accessors */
#define FGPIO_PDOR_REG(base)          ((base)->PDOR)
#define FGPIO_PSOR_REG(base)          ((base)->PSOR)
#define FGPIO_PCOR_REG(base)          ((base)->PCOR)
#define FGPIO_PTOR_REG(base)          ((base)->PTOR)
#define FGPIO_PDIR_REG(base)          ((base)->PDIR)
#define FGPIO_PDDR_REG(base)          ((base)->PDDR)

/* !
 * @}
 */
/* end of group FGPIO_Register_Accessor_Macros */

/*
-----
-- FGPIO Register Masks
-----
*/
/* !
 * @addtogroup FGPIO_Register_Masks FGPIO Register Masks
 * @{
 */

/* PDOR Bit Fields */
#define FGPIO_PDOR PDO_MASK           0xFFFFFFFFu
#define FGPIO_PDOR PDO_SHIFT          0
#define FGPIO_PDOR PDO_WIDTH          32

```

```

#define FGPIO_PDOR_PDO(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PDOR_PDO_SHIFT))&FGPIO_PDOR_PDO_MASK
/* PSOR Bit Fields */
#define FGPIO_PSOR_PTSO_MASK 0xFFFFFFFFu
#define FGPIO_PSOR_PTSO_SHIFT 0
#define FGPIO_PSOR_PTSO_WIDTH 32
#define FGPIO_PSOR_PTSO(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PSOR_PTSO_SHIFT))&FGPIO_PSOR_PTSO_MASK
/* PCOR Bit Fields */
#define FGPIO_PCOR_PTCO_MASK 0xFFFFFFFFu
#define FGPIO_PCOR_PTCO_SHIFT 0
#define FGPIO_PCOR_PTCO_WIDTH 32
#define FGPIO_PCOR_PTCO(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PCOR_PTCO_SHIFT))&FGPIO_PCOR_PTCO_MASK
/* PTOR Bit Fields */
#define FGPIO_PTOR_PTTO_MASK 0xFFFFFFFFu
#define FGPIO_PTOR_PTTO_SHIFT 0
#define FGPIO_PTOR_PTTO_WIDTH 32
#define FGPIO_PTOR_PTTO(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PTOR_PTTO_SHIFT))&FGPIO_PTOR_PTTO_MASK
/* PDIR Bit Fields */
#define FGPIO_PDIR_PDI_MASK 0xFFFFFFFFu
#define FGPIO_PDIR_PDI_SHIFT 0
#define FGPIO_PDIR_PDI_WIDTH 32
#define FGPIO_PDIR_PDI(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PDIR_PDI_SHIFT))&FGPIO_PDIR_PDI_MASK
/* PDDR Bit Fields */
#define FGPIO_PDDR_PDD_MASK 0xFFFFFFFFu
#define FGPIO_PDDR_PDD_SHIFT 0
#define FGPIO_PDDR_PDD_WIDTH 32
#define FGPIO_PDDR_PDD(x)
(((uint32_t)((uint32_t)(x))<<FGPIO_PDDR_PDD_SHIFT))&FGPIO_PDDR_PDD_MASK

/*!
 * @}
 */ /* end of group GPIO_Register_Masks */

```

```

/* GPIO - Peripheral instance base addresses */
/** Peripheral GPIOA base address */
#define GPIOA_BASE (0xF80FF000u)
/** Peripheral GPIOA base pointer */
#define GPIOA ((GPIO_Type *)GPIOA_BASE)
#define GPIOA_BASE_PTR (GPIOA)
/** Peripheral GPIOB base address */
#define GPIOB_BASE (0xF80FF040u)
/** Peripheral GPIOB base pointer */
#define GPIOB ((GPIO_Type *)GPIOB_BASE)
#define GPIOB_BASE_PTR (GPIOB)

```

```

/** Peripheral FGPIOC base address */
#define FGPIOC_BASE (0xF80FF080u)
/** Peripheral FGPIOC base pointer */
#define FGPIOC ((GPIO_Type
*)FGPIOC_BASE)
#define FGPIOC_PTR (FGPIOC)
/** Peripheral FGPIOD base address */
#define FGPIOD_BASE (0xF80FF0C0u)
/** Peripheral FGPIOD base pointer */
#define FGPIOD ((GPIO_Type
*)FGPIOD_BASE)
#define FGPIOD_PTR (FGPIOD)
/** Peripheral FGPIOE base address */
#define FGPIOE_BASE (0xF80FF100u)
/** Peripheral FGPIOE base pointer */
#define FGPIOE ((GPIO_Type
*)FGPIOE_BASE)
#define FGPIOE_PTR (FGPIOE)
/** Array initializer of GPIO peripheral base addresses */
#define GPIO_BASE_ADDRS { GPIOA_BASE,
GPIOB_BASE, FGPIOC_BASE, FGPIOD_BASE, FGPIOE_BASE }
/** Array initializer of GPIO peripheral base pointers */
#define GPIO_BASE_PTRS { GPIOA, GPIOB,
FGPIOC, FGPIOD, FGPIOE }

/* -----
-- Register accessor macros
-----
*/
/*!
 * @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor
macros
 * @{
 */

/* GPIO - Register instance definitions */
/* GPIOA */
#define GPIOA_PDOR GPIO_PDOR_REG(GPIOA)
#define GPIOA_PSOR GPIO_PSOR_REG(GPIOA)
#define GPIOA_PCOR GPIO_PCOR_REG(GPIOA)
#define GPIOA_PTOR GPIO_PTOR_REG(GPIOA)
#define GPIOA_PDIR GPIO_PDIR_REG(GPIOA)
#define GPIOA_PDDR GPIO_PDDR_REG(GPIOA)
/* GPIOB */
#define GPIOB_PDOR GPIO_PDOR_REG(GPIOB)
#define GPIOB_PSOR GPIO_PSOR_REG(GPIOB)
#define GPIOB_PCOR GPIO_PCOR_REG(GPIOB)
#define GPIOB_PTOR GPIO_PTOR_REG(GPIOB)
#define GPIOB_PDIR GPIO_PDIR_REG(GPIOB)
#define GPIOB_PDDR GPIO_PDDR_REG(GPIOB)
/* FGPIOC */

```

```

#define FGPIOC_PDOR           FGPIO_PDOR_REG(FGPIOC)
#define FGPIOC_PSOR           FGPIO_PSOR_REG(FGPIOC)
#define FGPIOC_PCOR           FGPIO_PCOR_REG(FGPIOC)
#define FGPIOC_PTOR           FGPIO_PTOR_REG(FGPIOC)
#define FGPIOC_PDIR           FGPIO_PDIR_REG(FGPIOC)
#define FGPIOC_PDDR           FGPIO_PDDR_REG(FGPIOC)
/* FGPIOD */
#define FGPIOD_PDOR           FGPIO_PDOR_REG(FGPIOD)
#define FGPIOD_PSOR           FGPIO_PSOR_REG(FGPIOD)
#define FGPIOD_PCOR           FGPIO_PCOR_REG(FGPIOD)
#define FGPIOD_PTOR           FGPIO_PTOR_REG(FGPIOD)
#define FGPIOD_PDIR           FGPIO_PDIR_REG(FGPIOD)
#define FGPIOD_PDDR           FGPIO_PDDR_REG(FGPIOD)
/* GPIOE */
#define FGPIOE_PDOR           FGPIO_PDOR_REG(FGPIOE)
#define FGPIOE_PSOR           FGPIO_PSOR_REG(FGPIOE)
#define FGPIOE_PCOR           FGPIO_PCOR_REG(FGPIOE)
#define FGPIOE_PTOR           FGPIO_PTOR_REG(FGPIOE)
#define FGPIOE_PDIR           FGPIO_PDIR_REG(FGPIOE)
#define FGPIOE_PDDR           FGPIO_PDDR_REG(FGPIOE)

/*!
 * @}
 */ /* end of group GPIO_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group GPIO_Peripheral_Access_Layer */

/* -----
-- FTFA Peripheral Access Layer
-----
*/
/*!
 * @addtogroup FTFA_Peripheral_Access_Layer FTFA Peripheral Access Layer
 * @{
 */
/** FTFA - Register Layout Typedef */
typedef struct {
    __IO uint8_t FSTAT;                      /**< Flash Status
Register, offset: 0x0 */
    __IO uint8_t FCNFG;                      /**< Flash
Configuration Register, offset: 0x1 */
    __I  uint8_t FSEC;                       /**< Flash Security
Register, offset: 0x2 */
    __I  uint8_t FOPT;                       /**< Flash Option
Register, offset: 0x3 */
    __IO uint8_t FCCOB3;                     /**< Flash Common
Command Object Registers, offset: 0x4 */

```

```

    __IO uint8_t FCCOB2;                                /**< Flash Common
Command Object Registers, offset: 0x5 */
    __IO uint8_t FCCOB1;                                /**< Flash Common
Command Object Registers, offset: 0x6 */
    __IO uint8_t FCCOB0;                                /**< Flash Common
Command Object Registers, offset: 0x7 */
    __IO uint8_t FCCOB7;                                /**< Flash Common
Command Object Registers, offset: 0x8 */
    __IO uint8_t FCCOB6;                                /**< Flash Common
Command Object Registers, offset: 0x9 */
    __IO uint8_t FCCOB5;                                /**< Flash Common
Command Object Registers, offset: 0xA */
    __IO uint8_t FCCOB4;                                /**< Flash Common
Command Object Registers, offset: 0xB */
    __IO uint8_t FCCOBB;                                /**< Flash Common
Command Object Registers, offset: 0xC */
    __IO uint8_t FCCOBA;                                /**< Flash Common
Command Object Registers, offset: 0xD */
    __IO uint8_t FCCOB9;                                /**< Flash Common
Command Object Registers, offset: 0xE */
    __IO uint8_t FCCOB8;                                /**< Flash Common
Command Object Registers, offset: 0xF */
    __IO uint8_t FPROT3;                                /**< Program Flash
Protection Registers, offset: 0x10 */
    __IO uint8_t FPROT2;                                /**< Program Flash
Protection Registers, offset: 0x11 */
    __IO uint8_t FPROT1;                                /**< Program Flash
Protection Registers, offset: 0x12 */
    __IO uint8_t FPROT0;                                /**< Program Flash
Protection Registers, offset: 0x13 */
} FTFA_Type, *FTFA_MemMapPtr;

/*
-----
-- FTFA - Register accessor macros
-----
*/
/*!
 * @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor
macros
 * @{
 */

/* FTFA - Register accessors */
#define FTFA_FSTAT_REG(base)          ((base) -> FSTAT)
#define FTFA_FCNGF_REG(base)          ((base) -> FCNFG)
#define FTFA_FSEC_REG(base)           ((base) -> FSEC)
#define FTFA_FOPT_REG(base)           ((base) -> FOPT)
#define FTFA_FCCOB3_REG(base)          ((base) -> FCCOB3)
#define FTFA_FCCOB2_REG(base)          ((base) -> FCCOB2)
#define FTFA_FCCOB1_REG(base)          ((base) -> FCCOB1)
#define FTFA_FCCOB0_REG(base)          ((base) -> FCCOB0)

```

```

#define FTFA_FCCOB7_REG(base)          ((base) ->FCCOB7)
#define FTFA_FCCOB6_REG(base)          ((base) ->FCCOB6)
#define FTFA_FCCOB5_REG(base)          ((base) ->FCCOB5)
#define FTFA_FCCOB4_REG(base)          ((base) ->FCCOB4)
#define FTFA_FCCOB_B_REG(base)         ((base) ->FCCOB_B)
#define FTFA_FCCOB_A_REG(base)         ((base) ->FCCOB_A)
#define FTFA_FCCOB9_REG(base)          ((base) ->FCCOB9)
#define FTFA_FCCOB8_REG(base)          ((base) ->FCCOB8)
#define FTFA_FPROT3_REG(base)          ((base) ->FPROT3)
#define FTFA_FPROT2_REG(base)          ((base) ->FPROT2)
#define FTFA_FPROT1_REG(base)          ((base) ->FPROT1)
#define FTFA_FPROT0_REG(base)          ((base) ->FPROT0)

/* !
 * @}
 */ /* end of group FTFA_Register_Accessor_Macros */

/*
-----  

-- FTFA Register Masks  

----- */
/* !
 * @addtogroup FTFA_Register_Masks FTFA Register Masks
 * @{
 */

/* FSTAT Bit Fields */
#define FTFA_FSTAT_MGSTAT0_MASK           0x1u
#define FTFA_FSTAT_MGSTAT0_SHIFT          0
#define FTFA_FSTAT_MGSTAT0_WIDTH          1
#define FTFA_FSTAT_MGSTAT0(x)            (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_MGSTAT0_SHIFT))&FTFA_FSTAT_MGSTAT0_MASK
#define FTFA_FSTAT_FPVIOL_MASK           0x10u
#define FTFA_FSTAT_FPVIOL_SHIFT          4
#define FTFA_FSTAT_FPVIOL_WIDTH          1
#define FTFA_FSTAT_FPVIOL(x)             (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_FPVIOL_SHIFT))&FTFA_FSTAT_FPVIOL_MASK
#define FTFA_FSTAT_ACCERR_MASK           0x20u
#define FTFA_FSTAT_ACCERR_SHIFT          5
#define FTFA_FSTAT_ACCERR_WIDTH          1
#define FTFA_FSTAT_ACCERR(x)             (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_ACCERR_SHIFT))&FTFA_FSTAT_ACCERR_MASK
#define FTFA_FSTAT_RDCOLERR_MASK          0x40u
#define FTFA_FSTAT_RDCOLERR_SHIFT         6
#define FTFA_FSTAT_RDCOLERR_WIDTH         1
#define FTFA_FSTAT_RDCOLERR(x)           (((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_RDCOLERR_SHIFT))&FTFA_FSTAT_RDCOLERR_MASK

```

```

#define FTFA_FSTAT_CCIF_MASK          0x80u
#define FTFA_FSTAT_CCIF_SHIFT         7
#define FTFA_FSTAT_CCIF_WIDTH          1
#define FTFA_FSTAT_CCIF(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSTAT_CCIF_SHIFT))&FTFA_FSTAT_CCIF_MASK
/* FCNFG Bit Fields */
#define FTFA_FCNFG_ERSSUSP_MASK        0x10u
#define FTFA_FCNFG_ERSSUSP_SHIFT       4
#define FTFA_FCNFG_ERSSUSP_WIDTH        1
#define FTFA_FCNFG_ERSSUSP(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_ERSSUSP_SHIFT))&FTFA_FCNFG_ERSSUSP_MASK
#define FTFA_FCNFG_ERSAREQ_MASK        0x20u
#define FTFA_FCNFG_ERSAREQ_SHIFT       5
#define FTFA_FCNFG_ERSAREQ_WIDTH        1
#define FTFA_FCNFG_ERSAREQ(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_ERSAREQ_SHIFT))&FTFA_FCNFG_ERSAREQ_MASK
#define FTFA_FCNFG_RDCOLLIE_MASK        0x40u
#define FTFA_FCNFG_RDCOLLIE_SHIFT       6
#define FTFA_FCNFG_RDCOLLIE_WIDTH        1
#define FTFA_FCNFG_RDCOLLIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_RDCOLLIE_SHIFT))&FTFA_FCNFG_RDCOLLIE_MASK
#define FTFA_FCNFG_CCIE_MASK           0x80u
#define FTFA_FCNFG_CCIE_SHIFT          7
#define FTFA_FCNFG_CCIE_WIDTH           1
#define FTFA_FCNFG_CCIE(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FCNFG_CCIE_SHIFT))&FTFA_FCNFG_CCIE_MASK
/* FSEC Bit Fields */
#define FTFA_FSEC_SEC_MASK             0x3u
#define FTFA_FSEC_SEC_SHIFT              0
#define FTFA_FSEC_SEC_WIDTH               2
#define FTFA_FSEC_SEC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_SEC_SHIFT))&FTFA_FSEC_SEC_MASK
#define FTFA_FSEC_FSLACC_MASK           0xCu
#define FTFA_FSEC_FSLACC_SHIFT          2
#define FTFA_FSEC_FSLACC_WIDTH           2
#define FTFA_FSEC_FSLACC(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_FSLACC_SHIFT))&FTFA_FSEC_FSLACC_MASK
#define FTFA_FSEC_MEEN_MASK             0x30u
#define FTFA_FSEC_MEEN_SHIFT              4
#define FTFA_FSEC_MEEN_WIDTH               2
#define FTFA_FSEC_MEEN(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_MEEN_SHIFT))&FTFA_FSEC_MEEN_MASK
#define FTFA_FSEC_KEYEN_MASK            0xC0u
#define FTFA_FSEC_KEYEN_SHIFT             6
#define FTFA_FSEC_KEYEN_WIDTH              2
#define FTFA_FSEC_KEYEN(x)
(((uint8_t)((uint8_t)(x))<<FTFA_FSEC_KEYEN_SHIFT))&FTFA_FSEC_KEYEN_MASK
/* FOPT Bit Fields */
#define FTFA_FOPT_OPT_MASK              0xFFu
#define FTFA_FOPT_OPT_SHIFT                 0

```

```

#define FTFA_FOPT_OPT_WIDTH 8
#define FTFA_FOPT_OPT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FOPT_OPT_SHIFT))&FTFA_FOPT_OPT_MASK)
/* FCCOB3 Bit Fields */
#define FTFA_FCCOB3_CCOBn_MASK 0xFFu
#define FTFA_FCCOB3_CCOBn_SHIFT 0
#define FTFA_FCCOB3_CCOBn_WIDTH 8
#define FTFA_FCCOB3_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB3_CCOBn_SHIFT))&FTFA_FCCOB3_CCOBn_MASK)
/* FCCOB2 Bit Fields */
#define FTFA_FCCOB2_CCOBn_MASK 0xFFu
#define FTFA_FCCOB2_CCOBn_SHIFT 0
#define FTFA_FCCOB2_CCOBn_WIDTH 8
#define FTFA_FCCOB2_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB2_CCOBn_SHIFT))&FTFA_FCCOB2_CCOBn_MASK)
/* FCCOB1 Bit Fields */
#define FTFA_FCCOB1_CCOBn_MASK 0xFFu
#define FTFA_FCCOB1_CCOBn_SHIFT 0
#define FTFA_FCCOB1_CCOBn_WIDTH 8
#define FTFA_FCCOB1_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB1_CCOBn_SHIFT))&FTFA_FCCOB1_CCOBn_MASK)
/* FCCOB0 Bit Fields */
#define FTFA_FCCOB0_CCOBn_MASK 0xFFu
#define FTFA_FCCOB0_CCOBn_SHIFT 0
#define FTFA_FCCOB0_CCOBn_WIDTH 8
#define FTFA_FCCOB0_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB0_CCOBn_SHIFT))&FTFA_FCCOB0_CCOBn_MASK)
/* FCCOB7 Bit Fields */
#define FTFA_FCCOB7_CCOBn_MASK 0xFFu
#define FTFA_FCCOB7_CCOBn_SHIFT 0
#define FTFA_FCCOB7_CCOBn_WIDTH 8
#define FTFA_FCCOB7_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB7_CCOBn_SHIFT))&FTFA_FCCOB7_CCOBn_MASK)
/* FCCOB6 Bit Fields */
#define FTFA_FCCOB6_CCOBn_MASK 0xFFu
#define FTFA_FCCOB6_CCOBn_SHIFT 0
#define FTFA_FCCOB6_CCOBn_WIDTH 8
#define FTFA_FCCOB6_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB6_CCOBn_SHIFT))&FTFA_FCCOB6_CCOBn_MASK)
/* FCCOB5 Bit Fields */
#define FTFA_FCCOB5_CCOBn_MASK 0xFFu
#define FTFA_FCCOB5_CCOBn_SHIFT 0
#define FTFA_FCCOB5_CCOBn_WIDTH 8
#define FTFA_FCCOB5_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB5_CCOBn_SHIFT))&FTFA_FCCOB5_CCOBn_MASK)
/* FCCOB4 Bit Fields */
#define FTFA_FCCOB4_CCOBn_MASK 0xFFu

```

```

#define FTFA_FCCOB4_CCOBn_SHIFT 0
#define FTFA_FCCOB4_CCOBn_WIDTH 8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK
/* FCCOBB Bit Fields */
#define FTFA_FCCOB4_CCOBn_MASK 0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT 0
#define FTFA_FCCOB4_CCOBn_WIDTH 8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK
/* FCCOBA Bit Fields */
#define FTFA_FCCOB4_CCOBn_MASK 0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT 0
#define FTFA_FCCOB4_CCOBn_WIDTH 8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK
/* FCCOB9 Bit Fields */
#define FTFA_FCCOB4_CCOBn_MASK 0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT 0
#define FTFA_FCCOB4_CCOBn_WIDTH 8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK
/* FCCOB8 Bit Fields */
#define FTFA_FCCOB4_CCOBn_MASK 0xFFu
#define FTFA_FCCOB4_CCOBn_SHIFT 0
#define FTFA_FCCOB4_CCOBn_WIDTH 8
#define FTFA_FCCOB4_CCOBn(x) (((uint8_t)((uint8_t)(x))<<FTFA_FCCOB4_CCOBn_SHIFT))&FTFA_FCCOB4_CCOBn_MASK
/* FPROT3 Bit Fields */
#define FTFA_FPROT3_PROT_MASK 0xFFu
#define FTFA_FPROT3_PROT_SHIFT 0
#define FTFA_FPROT3_PROT_WIDTH 8
#define FTFA_FPROT3_PROT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK
/* FPROT2 Bit Fields */
#define FTFA_FPROT3_PROT_MASK 0xFFu
#define FTFA_FPROT3_PROT_SHIFT 0
#define FTFA_FPROT3_PROT_WIDTH 8
#define FTFA_FPROT3_PROT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK
/* FPROT1 Bit Fields */
#define FTFA_FPROT3_PROT_MASK 0xFFu
#define FTFA_FPROT3_PROT_SHIFT 0
#define FTFA_FPROT3_PROT_WIDTH 8
#define FTFA_FPROT3_PROT(x) (((uint8_t)((uint8_t)(x))<<FTFA_FPROT3_PROT_SHIFT))&FTFA_FPROT3_PROT_MASK

```

```

/* FPROT0 Bit Fields */
#define FTFA_FPROT0_PROT_MASK          0xFFu
#define FTFA_FPROT0_PROT_SHIFT         0
#define FTFA_FPROT0_PROT_WIDTH         8
#define FTFA_FPROT0_PROT(x)            (((uint8_t)((uint8_t)(x))<<FTFA_FPROT0_PROT_SHIFT))&FTFA_FPROT0_PROT_MASK

/*
 * @}
 */
/* end of group FTFA_Register_Masks */

/* FTFA - Peripheral instance base addresses */
/** Peripheral FTFA base address */
#define FTFA_BASE                      (0x40020000u)
/** Peripheral FTFA base pointer */
#define FTFA                           ((FTFA_Type *)FTFA_BASE)
#define FTFA_PTR                       (FTFA)
/** Array initializer of FTFA peripheral base addresses */
#define FTFA_BASE_ADDRS                { FTFA_BASE }
/** Array initializer of FTFA peripheral base pointers */
#define FTFA_BASE_PTRS                 { FTFA }
/** Interrupt vectors for the FTFA peripheral type */
#define FTFA_COMMAND_COMPLETE_IRQS     { FTFA_IRQn }

/* -----
-- FTFA - Register accessor macros
-----
*/
/* !
 * @addtogroup FTFA_Register_Accessor_Macros FTFA - Register accessor
macros
 * @{
 */
/* FTFA - Register instance definitions */
/* FTFA */
#define FTFA_FSTAT                     FTFA_FSTAT_REG(FTFA)
#define FTFA_FCNGF                      FTFA_FCNGF_REG(FTFA)
#define FTFA_FSEC                      FTFA_FSEC_REG(FTFA)
#define FTFA_FOPT                      FTFA_FOPT_REG(FTFA)
#define FTFA_FCCOB3                    FTFA_FCCOB3_REG(FTFA)
#define FTFA_FCCOB2                    FTFA_FCCOB2_REG(FTFA)
#define FTFA_FCCOB1                    FTFA_FCCOB1_REG(FTFA)
#define FTFA_FCCOB0                    FTFA_FCCOB0_REG(FTFA)
#define FTFA_FCCOB7                    FTFA_FCCOB7_REG(FTFA)
#define FTFA_FCCOB6                    FTFA_FCCOB6_REG(FTFA)
#define FTFA_FCCOB5                    FTFA_FCCOB5_REG(FTFA)
#define FTFA_FCCOB4                    FTFA_FCCOB4_REG(FTFA)
#define FTFA_FCCOB8                    FTFA_FCCOB8_REG(FTFA)

```

```

#define FTFA_FCCOBA FTFA_FCCOBA_REG(FTFA)
#define FTFA_FCCOB9 FTFA_FCCOB9_REG(FTFA)
#define FTFA_FCCOB8 FTFA_FCCOB8_REG(FTFA)
#define FTFA_FPROT3 FTFA_FPROT3_REG(FTFA)
#define FTFA_FPROT2 FTFA_FPROT2_REG(FTFA)
#define FTFA_FPROT1 FTFA_FPROT1_REG(FTFA)
#define FTFA_FPROT0 FTFA_FPROT0_REG(FTFA)

/*!
 * @}
 */ /* end of group FTFA_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group FTFA_Peripheral_Access_Layer */

/* -----
-- GPIO Peripheral Access Layer
-----
*/
/*!
 * @addtogroup GPIO_Peripheral_Access_Layer GPIO Peripheral Access Layer
 * @{
 */

/** GPIO - Register Layout Typedef */
typedef struct {
    __IO uint32_t PDOR;                                /**< Port Data Output
Register, offset: 0x0 */
    __O uint32_t PSOR;                                /**< Port Set Output
Register, offset: 0x4 */
    __O uint32_t PCOR;                                /**< Port Clear Output
Register, offset: 0x8 */
    __O uint32_t PTOR;                                /**< Port Toggle
Output Register, offset: 0xC */
    __I uint32_t PDIR;                                /**< Port Data Input
Register, offset: 0x10 */
    __IO uint32_t PDDR;                                /**< Port Data
Direction Register, offset: 0x14 */
} GPIO_Type, *GPIO_MemMapPtr;

/* -----
-- GPIO - Register accessor macros
-----
*/
/*!
 * @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor
macros
*/

```

```

* @{
*/



/* GPIO - Register accessors */
#define GPIO_PDOR_REG(base) ((base)->PDOR)
#define GPIO_PSOR_REG(base) ((base)->PSOR)
#define GPIO_PCOR_REG(base) ((base)->PCOR)
#define GPIO_PTOR_REG(base) ((base)->PTOR)
#define GPIO_PDIR_REG(base) ((base)->PDIR)
#define GPIO_PDDR_REG(base) ((base)->PDDR)

/*!
* @}
*/
/* end of group GPIO_Register_Accessor_Macros */



/* -----
-- GPIO Register Masks
-----
*/
/*!
* @addtogroup GPIO_Register_Masks GPIO Register Masks
* @{
*/
/* PDOR Bit Fields */
#define GPIO_PDOR_PDO_MASK 0xFFFFFFFFu
#define GPIO_PDOR_PDO_SHIFT 0
#define GPIO_PDOR_PDO_WIDTH 32
#define GPIO_PDOR_PDO(x) (((uint32_t)((uint32_t)(x))<<GPIO_PDOR_PDO_SHIFT))&GPIO_PDOR_PDO_MASK)
/* PSOR Bit Fields */
#define GPIO_PSOR PTSO_MASK 0xFFFFFFFFu
#define GPIO_PSOR PTSO_SHIFT 0
#define GPIO_PSOR PTSO_WIDTH 32
#define GPIO_PSOR PTSO(x) (((uint32_t)((uint32_t)(x))<<GPIO_PSOR PTSO_SHIFT))&GPIO_PSOR PTSO_MASK)
/* PCOR Bit Fields */
#define GPIO_PCOR_PTCO_MASK 0xFFFFFFFFu
#define GPIO_PCOR_PTCO_SHIFT 0
#define GPIO_PCOR_PTCO_WIDTH 32
#define GPIO_PCOR_PTCO(x) (((uint32_t)((uint32_t)(x))<<GPIO_PCOR_PTCO_SHIFT))&GPIO_PCOR_PTCO_MASK)
/* PTOR Bit Fields */
#define GPIO_PTOR_PTTO_MASK 0xFFFFFFFFu
#define GPIO_PTOR_PTTO_SHIFT 0
#define GPIO_PTOR_PTTO_WIDTH 32
#define GPIO_PTOR_PTTO(x) (((uint32_t)((uint32_t)(x))<<GPIO_PTOR_PTTO_SHIFT))&GPIO_PTOR_PTTO_MASK)
/* PDIR Bit Fields */
#define GPIO_PDIR_PDI_MASK 0xFFFFFFFFu

```

```

#define GPIO_PDIR_PDI_SHIFT          0
#define GPIO_PDIR_PDI_WIDTH          32
#define GPIO_PDIR_PDI(x)
(((uint32_t)((uint32_t)(x))<<GPIO_PDIR_PDI_SHIFT))&GPIO_PDIR_PDI_MASK)
/* PDDR Bit Fields */
#define GPIO_PDDR_PDD_MASK           0xFFFFFFFFu
#define GPIO_PDDR_PDD_SHIFT          0
#define GPIO_PDDR_PDD_WIDTH          32
#define GPIO_PDDR_PDD(x)
(((uint32_t)((uint32_t)(x))<<GPIO_PDDR_PDD_SHIFT))&GPIO_PDDR_PDD_MASK)

/*
 * @}
 */ /* end of group GPIO_Register_Masks */

/* GPIO - Peripheral instance base addresses */
/** Peripheral GPIOA base address */
#define GPIOA_BASE                   (0x400FF000u)
/** Peripheral GPIOA base pointer */
#define GPIOA                         ((GPIO_Type
*)GPIOA_BASE)
#define GPIOA_BASE_PTR                (GPIOA)
/** Peripheral GPIOB base address */
#define GPIOB_BASE                   (0x400FF040u)
/** Peripheral GPIOB base pointer */
#define GPIOB                         ((GPIO_Type
*)GPIOB_BASE)
#define GPIOB_BASE_PTR                (GPIOB)
/** Peripheral GPIOC base address */
#define GPIOC_BASE                   (0x400FF080u)
/** Peripheral GPIOC base pointer */
#define GPIOC                         ((GPIO_Type
*)GPIOC_BASE)
#define GPIOC_BASE_PTR                (GPIOC)
/** Peripheral GPIOD base address */
#define GPIOD_BASE                   (0x400FF0C0u)
/** Peripheral GPIOD base pointer */
#define GPIOD                         ((GPIO_Type
*)GPIOD_BASE)
#define GPIOD_BASE_PTR                (GPIOD)
/** Peripheral GPIOE base address */
#define GPIOE_BASE                   (0x400FF100u)
/** Peripheral GPIOE base pointer */
#define GPIOE                         ((GPIO_Type
*)GPIOE_BASE)
#define GPIOE_BASE_PTR                (GPIOE)
/** Array initializer of GPIO peripheral base addresses */
#define GPIO_BASE_ADDRS              { GPIOA_BASE,
GPIOB_BASE, GPIOC_BASE, GPIOD_BASE, GPIOE_BASE }
/** Array initializer of GPIO peripheral base pointers */
#define GPIO_BASE_PTRS                { GPIOA, GPIOB, GPIOC,
GPIOD, GPIOE }

```

```

/* -----
-- GPIO - Register accessor macros
-----
*/
/*!
 * @addtogroup GPIO_Register_Accessor_Macros GPIO - Register accessor
macros
 * @{
 */

/* GPIO - Register instance definitions */
/* GPIOA */
#define GPIOA_PDOR           GPIO_PDOR_REG(GPIOA)
#define GPIOA_PSOR           GPIO_PSOR_REG(GPIOA)
#define GPIOA_PCOR           GPIO_PCOR_REG(GPIOA)
#define GPIOA_PTOR           GPIO_PTOR_REG(GPIOA)
#define GPIOA_PDIR           GPIO_PDIR_REG(GPIOA)
#define GPIOA_PDDR           GPIO_PDDR_REG(GPIOA)
/* GPIOB */
#define GPIOB_PDOR           GPIO_PDOR_REG(GPIOB)
#define GPIOB_PSOR           GPIO_PSOR_REG(GPIOB)
#define GPIOB_PCOR           GPIO_PCOR_REG(GPIOB)
#define GPIOB_PTOR           GPIO_PTOR_REG(GPIOB)
#define GPIOB_PDIR           GPIO_PDIR_REG(GPIOB)
#define GPIOB_PDDR           GPIO_PDDR_REG(GPIOB)
/* GPIOC */
#define GPIOC_PDOR           GPIO_PDOR_REG(GPIOC)
#define GPIOC_PSOR           GPIO_PSOR_REG(GPIOC)
#define GPIOC_PCOR           GPIO_PCOR_REG(GPIOC)
#define GPIOC_PTOR           GPIO_PTOR_REG(GPIOC)
#define GPIOC_PDIR           GPIO_PDIR_REG(GPIOC)
#define GPIOC_PDDR           GPIO_PDDR_REG(GPIOC)
/* GPIOD */
#define GPIOD_PDOR           GPIO_PDOR_REG(GPIOD)
#define GPIOD_PSOR           GPIO_PSOR_REG(GPIOD)
#define GPIOD_PCOR           GPIO_PCOR_REG(GPIOD)
#define GPIOD_PTOR           GPIO_PTOR_REG(GPIOD)
#define GPIOD_PDIR           GPIO_PDIR_REG(GPIOD)
#define GPIOD_PDDR           GPIO_PDDR_REG(GPIOD)
/* GPIOE */
#define GPIOE_PDOR           GPIO_PDOR_REG(GPIOE)
#define GPIOE_PSOR           GPIO_PSOR_REG(GPIOE)
#define GPIOE_PCOR           GPIO_PCOR_REG(GPIOE)
#define GPIOE_PTOR           GPIO_PTOR_REG(GPIOE)
#define GPIOE_PDIR           GPIO_PDIR_REG(GPIOE)
#define GPIOE_PDDR           GPIO_PDDR_REG(GPIOE)

/*!
 * @}
 */
/* */ /* end of group GPIO_Register_Accessor_Macros */

```

```

/*!
 * @}
 */
/* end of group GPIO_Peripheral_Access_Layer */

/*
-----
-- I2C Peripheral Access Layer
-----
*/
/*!
 * @addtogroup I2C_Peripheral_Access_Layer I2C Peripheral Access Layer
 * @{
 */

/** I2C - Register Layout Typedef */
typedef struct {
    __IO uint8_t A1;                                /**< I2C Address
Register 1, offset: 0x0 */
    __IO uint8_t F;                                /**< I2C Frequency
Divider register, offset: 0x1 */
    __IO uint8_t C1;                               /**< I2C Control
Register 1, offset: 0x2 */
    __IO uint8_t S;                                /**< I2C Status
register, offset: 0x3 */
    __IO uint8_t D;                                /**< I2C Data I/O
register, offset: 0x4 */
    __IO uint8_t C2;                               /**< I2C Control
Register 2, offset: 0x5 */
    __IO uint8_t FLT;                             /**< I2C Programmable
Input Glitch Filter register, offset: 0x6 */
    __IO uint8_t RA;                               /**< I2C Range Address
register, offset: 0x7 */
    __IO uint8_t SMB;                            /**< I2C SMBus Control
and Status register, offset: 0x8 */
    __IO uint8_t A2;                               /**< I2C Address
Register 2, offset: 0x9 */
    __IO uint8_t SLTH;                            /**< I2C SCL Low
Timeout Register High, offset: 0xA */
    __IO uint8_t SLTL;                            /**< I2C SCL Low
Timeout Register Low, offset: 0xB */
} I2C_Type, *I2C_MemMapPtr;

/*
-----
-- I2C - Register accessor macros
-----
*/
/*!
 * @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor
macros
*/

```

```

* @{
}

/* I2C - Register accessors */
#define I2C_A1_REG(base)          ((base)->A1)
#define I2C_F_REG(base)           ((base)->F)
#define I2C_C1_REG(base)          ((base)->C1)
#define I2C_S_REG(base)           ((base)->S)
#define I2C_D_REG(base)           ((base)->D)
#define I2C_C2_REG(base)          ((base)->C2)
#define I2C_FLT_REG(base)         ((base)->FLT)
#define I2C_RA_REG(base)          ((base)->RA)
#define I2C_SMB_REG(base)         ((base)->SMB)
#define I2C_A2_REG(base)          ((base)->A2)
#define I2C_SLTH_REG(base)        ((base)->SLTH)
#define I2C_SLTL_REG(base)        ((base)->SLTL)

/* !
 * @}
 */ /* end of group I2C_Register_Accessor_Macros */

/* -----
-- I2C Register Masks
-----
*/
/* !
 * @addtogroup I2C_Register_Masks I2C Register Masks
 * @{
 */

/* A1 Bit Fields */
#define I2C_A1_AD_MASK           0xFFu
#define I2C_A1_AD_SHIFT          1
#define I2C_A1_AD_WIDTH          7
#define I2C_A1_AD(x)             (((uint8_t)((uint8_t)(x))<<I2C_A1_AD_SHIFT))&I2C_A1_AD_MASK)
/* F Bit Fields */
#define I2C_F_ICR_MASK           0x3Fu
#define I2C_F_ICR_SHIFT          0
#define I2C_F_ICR_WIDTH          6
#define I2C_F_ICR(x)              (((uint8_t)((uint8_t)(x))<<I2C_F_ICR_SHIFT))&I2C_F_ICR_MASK)
#define I2C_F_MULT_MASK           0xC0u
#define I2C_F_MULT_SHIFT          6
#define I2C_F_MULT_WIDTH          2
#define I2C_F_MULT(x)              (((uint8_t)((uint8_t)(x))<<I2C_F_MULT_SHIFT))&I2C_F_MULT_MASK)
/* C1 Bit Fields */
#define I2C_C1_DMAEN_MASK         0x1u
#define I2C_C1_DMAEN_SHIFT         0

```

```

#define I2C_C1_DMAEN_WIDTH 1
#define I2C_C1_DMAEN(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_DMAEN_SHIFT))&I2C_C1_DMAEN_MASK)
#define I2C_C1_WUEN_MASK 0x2u
#define I2C_C1_WUEN_SHIFT 1
#define I2C_C1_WUEN_WIDTH 1
#define I2C_C1_WUEN(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_WUEN_SHIFT))&I2C_C1_WUEN_MASK)
#define I2C_C1_RSTA_MASK 0x4u
#define I2C_C1_RSTA_SHIFT 2
#define I2C_C1_RSTA_WIDTH 1
#define I2C_C1_RSTA(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_RSTA_SHIFT))&I2C_C1_RSTA_MASK)
#define I2C_C1_TXAK_MASK 0x8u
#define I2C_C1_TXAK_SHIFT 3
#define I2C_C1_TXAK_WIDTH 1
#define I2C_C1_TXAK(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_TXAK_SHIFT))&I2C_C1_TXAK_MASK)
#define I2C_C1_TX_MASK 0x10u
#define I2C_C1_TX_SHIFT 4
#define I2C_C1_TX_WIDTH 1
#define I2C_C1_TX(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_TX_SHIFT))&I2C_C1_TX_MASK)
#define I2C_C1_MST_MASK 0x20u
#define I2C_C1_MST_SHIFT 5
#define I2C_C1_MST_WIDTH 1
#define I2C_C1_MST(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_MST_SHIFT))&I2C_C1_MST_MASK)
#define I2C_C1_IICIE_MASK 0x40u
#define I2C_C1_IICIE_SHIFT 6
#define I2C_C1_IICIE_WIDTH 1
#define I2C_C1_IICIE(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_IICIE_SHIFT))&I2C_C1_IICIE_MASK)
#define I2C_C1_IICEN_MASK 0x80u
#define I2C_C1_IICEN_SHIFT 7
#define I2C_C1_IICEN_WIDTH 1
#define I2C_C1_IICEN(x) (((uint8_t)((uint8_t)(x))<<I2C_C1_IICEN_SHIFT))&I2C_C1_IICEN_MASK)
/* S Bit Fields */
#define I2C_S_RXAK_MASK 0x1u
#define I2C_S_RXAK_SHIFT 0
#define I2C_S_RXAK_WIDTH 1
#define I2C_S_RXAK(x) (((uint8_t)((uint8_t)(x))<<I2C_S_RXAK_SHIFT))&I2C_S_RXAK_MASK)
#define I2C_S_IICIF_MASK 0x2u
#define I2C_S_IICIF_SHIFT 1
#define I2C_S_IICIF_WIDTH 1
#define I2C_S_IICIF(x) (((uint8_t)((uint8_t)(x))<<I2C_S_IICIF_SHIFT))&I2C_S_IICIF_MASK)
#define I2C_S_SRW_MASK 0x4u
#define I2C_S_SRW_SHIFT 2
#define I2C_S_SRW_WIDTH 1
#define I2C_S_SRW(x) (((uint8_t)((uint8_t)(x))<<I2C_S_SRW_SHIFT))&I2C_S_SRW_MASK)

```

```

#define I2C_S_RAM_MASK          0x8u
#define I2C_S_RAM_SHIFT         3
#define I2C_S_RAM_WIDTH         1
#define I2C_S_RAM(x)
  (((uint8_t)((uint8_t)(x)<<I2C_S_RAM_SHIFT))&I2C_S_RAM_MASK)
#define I2C_S_ARBL_MASK         0x10u
#define I2C_S_ARBL_SHIFT        4
#define I2C_S_ARBL_WIDTH        1
#define I2C_S_ARBL(x)
  (((uint8_t)((uint8_t)(x)<<I2C_S_ARBL_SHIFT))&I2C_S_ARBL_MASK)
#define I2C_S_BUSY_MASK          0x20u
#define I2C_S_BUSY_SHIFT         5
#define I2C_S_BUSY_WIDTH         1
#define I2C_S_BUSY(x)
  (((uint8_t)((uint8_t)(x)<<I2C_S_BUSY_SHIFT))&I2C_S_BUSY_MASK)
#define I2C_S_IAAS_MASK          0x40u
#define I2C_S_IAAS_SHIFT         6
#define I2C_S_IAAS_WIDTH         1
#define I2C_S_IAAS(x)
  (((uint8_t)((uint8_t)(x)<<I2C_S_IAAS_SHIFT))&I2C_S_IAAS_MASK)
#define I2C_S_TCF_MASK           0x80u
#define I2C_S_TCF_SHIFT          7
#define I2C_S_TCF_WIDTH          1
#define I2C_S_TCF(x)
  (((uint8_t)((uint8_t)(x)<<I2C_S_TCF_SHIFT))&I2C_S_TCF_MASK)
/* D Bit Fields */
#define I2C_D_DATA_MASK          0xFFu
#define I2C_D_DATA_SHIFT          0
#define I2C_D_DATA_WIDTH          8
#define I2C_D_DATA(x)
  (((uint8_t)((uint8_t)(x)<<I2C_D_DATA_SHIFT))&I2C_D_DATA_MASK)
/* C2 Bit Fields */
#define I2C_C2_AD_MASK           0x7u
#define I2C_C2_AD_SHIFT          0
#define I2C_C2_AD_WIDTH          3
#define I2C_C2_AD(x)
  (((uint8_t)((uint8_t)(x)<<I2C_C2_AD_SHIFT))&I2C_C2_AD_MASK)
#define I2C_C2_RMEN_MASK          0x8u
#define I2C_C2_RMEN_SHIFT         3
#define I2C_C2_RMEN_WIDTH         1
#define I2C_C2_RMEN(x)
  (((uint8_t)((uint8_t)(x)<<I2C_C2_RMEN_SHIFT))&I2C_C2_RMEN_MASK)
#define I2C_C2_SBRC_MASK          0x10u
#define I2C_C2_SBRC_SHIFT         4
#define I2C_C2_SBRC_WIDTH         1
#define I2C_C2_SBRC(x)
  (((uint8_t)((uint8_t)(x)<<I2C_C2_SBRC_SHIFT))&I2C_C2_SBRC_MASK)
#define I2C_C2_HDRS_MASK          0x20u
#define I2C_C2_HDRS_SHIFT         5
#define I2C_C2_HDRS_WIDTH         1
#define I2C_C2_HDRS(x)
  (((uint8_t)((uint8_t)(x)<<I2C_C2_HDRS_SHIFT))&I2C_C2_HDRS_MASK)
#define I2C_C2_ADEXT_MASK          0x40u
#define I2C_C2_ADEXT_SHIFT         6

```

```

#define I2C_C2_ADEXT_WIDTH 1
#define I2C_C2_ADEXT(x)
(((uint8_t) (((uint8_t)(x))<<I2C_C2_ADEXT_SHIFT))&I2C_C2_ADEXT_MASK)
#define I2C_C2_GCAEN_MASK 0x80u
#define I2C_C2_GCAEN_SHIFT 7
#define I2C_C2_GCAEN_WIDTH 1
#define I2C_C2_GCAEN(x)
(((uint8_t) (((uint8_t)(x))<<I2C_C2_GCAEN_SHIFT))&I2C_C2_GCAEN_MASK)
/* FLT Bit Fields */
#define I2C_FLT_FLT_MASK 0x1Fu
#define I2C_FLT_FLT_SHIFT 0
#define I2C_FLT_FLT_WIDTH 5
#define I2C_FLT_FLT(x)
(((uint8_t) (((uint8_t)(x))<<I2C_FLT_FLT_SHIFT))&I2C_FLT_FLT_MASK)
#define I2C_FLT_STOPIE_MASK 0x20u
#define I2C_FLT_STOPIE_SHIFT 5
#define I2C_FLT_STOPIE_WIDTH 1
#define I2C_FLT_STOPIE(x)
(((uint8_t) (((uint8_t)(x))<<I2C_FLT_STOPIE_SHIFT))&I2C_FLT_STOPIE_MASK)
#define I2C_FLT_STOPF_MASK 0x40u
#define I2C_FLT_STOPF_SHIFT 6
#define I2C_FLT_STOPF_WIDTH 1
#define I2C_FLT_STOPF(x)
(((uint8_t) (((uint8_t)(x))<<I2C_FLT_STOPF_SHIFT))&I2C_FLT_STOPF_MASK)
#define I2C_FLT_SHEN_MASK 0x80u
#define I2C_FLT_SHEN_SHIFT 7
#define I2C_FLT_SHEN_WIDTH 1
#define I2C_FLT_SHEN(x)
(((uint8_t) (((uint8_t)(x))<<I2C_FLT_SHEN_SHIFT))&I2C_FLT_SHEN_MASK)
/* RA Bit Fields */
#define I2C_RA_RAD_MASK 0xFEu
#define I2C_RA_RAD_SHIFT 1
#define I2C_RA_RAD_WIDTH 7
#define I2C_RA_RAD(x)
(((uint8_t) (((uint8_t)(x))<<I2C_RA_RAD_SHIFT))&I2C_RA_RAD_MASK)
/* SMB Bit Fields */
#define I2C_SMB_SHTF2IE_MASK 0x1u
#define I2C_SMB_SHTF2IE_SHIFT 0
#define I2C_SMB_SHTF2IE_WIDTH 1
#define I2C_SMB_SHTF2IE(x)
(((uint8_t) (((uint8_t)(x))<<I2C_SMB_SHTF2IE_SHIFT))&I2C_SMB_SHTF2IE_MASK)
#define I2C_SMB_SHTF2_MASK 0x2u
#define I2C_SMB_SHTF2_SHIFT 1
#define I2C_SMB_SHTF2_WIDTH 1
#define I2C_SMB_SHTF2(x)
(((uint8_t) (((uint8_t)(x))<<I2C_SMB_SHTF2_SHIFT))&I2C_SMB_SHTF2_MASK)
#define I2C_SMB_SHTF1_MASK 0x4u
#define I2C_SMB_SHTF1_SHIFT 2
#define I2C_SMB_SHTF1_WIDTH 1
#define I2C_SMB_SHTF1(x)
(((uint8_t) (((uint8_t)(x))<<I2C_SMB_SHTF1_SHIFT))&I2C_SMB_SHTF1_MASK)
#define I2C_SMB_SLTF_MASK 0x8u
#define I2C_SMB_SLTF_SHIFT 3
#define I2C_SMB_SLTF_WIDTH 1

```

```

#define I2C_SMB_SLTF(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_SLTF_SHIFT))&I2C_SMB_SLTF_MASK)
#define I2C_SMB_TCKSEL_MASK 0x10u
#define I2C_SMB_TCKSEL_SHIFT 4
#define I2C_SMB_TCKSEL_WIDTH 1
#define I2C_SMB_TCKSEL(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_TCKSEL_SHIFT))&I2C_SMB_TCKSEL_MASK)
#define I2C_SMB_SIICAEN_MASK 0x20u
#define I2C_SMB_SIICAEN_SHIFT 5
#define I2C_SMB_SIICAEN_WIDTH 1
#define I2C_SMB_SIICAEN(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_SIICAEN_SHIFT))&I2C_SMB_SIICAEN_MASK)
#define I2C_SMB_ALERTEN_MASK 0x40u
#define I2C_SMB_ALERTEN_SHIFT 6
#define I2C_SMB_ALERTEN_WIDTH 1
#define I2C_SMB_ALERTEN(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_ALERTEN_SHIFT))&I2C_SMB_ALERTEN_MASK)
#define I2C_SMB_FACK_MASK 0x80u
#define I2C_SMB_FACK_SHIFT 7
#define I2C_SMB_FACK_WIDTH 1
#define I2C_SMB_FACK(x)
(((uint8_t)((uint8_t)(x))<<I2C_SMB_FACK_SHIFT))&I2C_SMB_FACK_MASK)
/* A2 Bit Fields */
#define I2C_A2_SAD_MASK 0xFEu
#define I2C_A2_SAD_SHIFT 1
#define I2C_A2_SAD_WIDTH 7
#define I2C_A2_SAD(x)
(((uint8_t)((uint8_t)(x))<<I2C_A2_SAD_SHIFT))&I2C_A2_SAD_MASK)
/* SLTH Bit Fields */
#define I2C_SLTH_SS LT_MASK 0xFFu
#define I2C_SLTH_SS LT_SHIFT 0
#define I2C_SLTH_SS LT_WIDTH 8
#define I2C_SLTH_SS LT(x)
(((uint8_t)((uint8_t)(x))<<I2C_SLTH_SS LT_SHIFT))&I2C_SLTH_SS LT_MASK)
/* SLTL Bit Fields */
#define I2C_SLTL_SS LT_MASK 0xFFu
#define I2C_SLTL_SS LT_SHIFT 0
#define I2C_SLTL_SS LT_WIDTH 8
#define I2C_SLTL_SS LT(x)
(((uint8_t)((uint8_t)(x))<<I2C_SLTL_SS LT_SHIFT))&I2C_SLTL_SS LT_MASK)

/*
 * @}
 */ /* end of group I2C_Register_Masks */

/* I2C - Peripheral instance base addresses */
/** Peripheral I2C0 base address */
#define I2C0_BASE (0x40066000u)
/** Peripheral I2C0 base pointer */
#define I2C0 ((I2C_Type *)I2C0_BASE)
#define I2C0_BASE_PTR (I2C0)
/** Peripheral I2C1 base address */
#define I2C1_BASE (0x40067000u)

```

```

/** Peripheral I2C1 base pointer */
#define I2C1 ((I2C_Type *)I2C1_BASE)
#define I2C1_BASE_PTR (I2C1)
/** Array initializer of I2C peripheral base addresses */
#define I2C_BASE_ADDRS { I2C0_BASE, I2C1_BASE }
/** Array initializer of I2C peripheral base pointers */
#define I2C_BASE_PTRS { I2C0, I2C1 }
/** Interrupt vectors for the I2C peripheral type */
#define I2C_IRQS { I2C0_IRQn, I2C1_IRQn }

/* -----
-- I2C - Register accessor macros
-----
*/
/*!
 * @addtogroup I2C_Register_Accessor_Macros I2C - Register accessor
 * macros
 * @{
 */

/* I2C - Register instance definitions */
/* I2C0 */
#define I2C0_A1 I2C_A1_REG(I2C0)
#define I2C0_F I2C_F_REG(I2C0)
#define I2C0_C1 I2C_C1_REG(I2C0)
#define I2C0_S I2C_S_REG(I2C0)
#define I2C0_D I2C_D_REG(I2C0)
#define I2C0_C2 I2C_C2_REG(I2C0)
#define I2C0_FLT I2C_FLT_REG(I2C0)
#define I2C0_RA I2C_RA_REG(I2C0)
#define I2C0_SMB I2C_SMB_REG(I2C0)
#define I2C0_A2 I2C_A2_REG(I2C0)
#define I2C0_SLTH I2C_SLTH_REG(I2C0)
#define I2C0_SLTL I2C_SLTL_REG(I2C0)
/* I2C1 */
#define I2C1_A1 I2C_A1_REG(I2C1)
#define I2C1_F I2C_F_REG(I2C1)
#define I2C1_C1 I2C_C1_REG(I2C1)
#define I2C1_S I2C_S_REG(I2C1)
#define I2C1_D I2C_D_REG(I2C1)
#define I2C1_C2 I2C_C2_REG(I2C1)
#define I2C1_FLT I2C_FLT_REG(I2C1)
#define I2C1_RA I2C_RA_REG(I2C1)
#define I2C1_SMB I2C_SMB_REG(I2C1)
#define I2C1_A2 I2C_A2_REG(I2C1)
#define I2C1_SLTH I2C_SLTH_REG(I2C1)
#define I2C1_SLTL I2C_SLTL_REG(I2C1)

/*!
 * @}
 */
/* end of group I2C_Register_Accessor_Macros */

```

```

/*!
 * @}
 */
/* end of group I2C_Peripheral_Access_Layer */

/*
-----  

-- LLWU Peripheral Access Layer  

-----  

----- */

/*!
 * @addtogroup LLWU_Peripheral_Access_Layer LLWU Peripheral Access Layer
 * @{
 */

/** LLWU - Register Layout Typedef */
typedef struct {
    __IO uint8_t PE1;                                /**< LLWU Pin Enable 1
register, offset: 0x0 */
    __IO uint8_t PE2;                                /**< LLWU Pin Enable 2
register, offset: 0x1 */
    __IO uint8_t PE3;                                /**< LLWU Pin Enable 3
register, offset: 0x2 */
    __IO uint8_t PE4;                                /**< LLWU Pin Enable 4
register, offset: 0x3 */
    __IO uint8_t ME;                                 /**< LLWU Module
Enable register, offset: 0x4 */
    __IO uint8_t F1;                                 /**< LLWU Flag 1
register, offset: 0x5 */
    __IO uint8_t F2;                                 /**< LLWU Flag 2
register, offset: 0x6 */
    __I  uint8_t F3;                                /**< LLWU Flag 3
register, offset: 0x7 */
    __IO uint8_t FILT1;                             /**< LLWU Pin Filter 1
register, offset: 0x8 */
    __IO uint8_t FILT2;                             /**< LLWU Pin Filter 2
register, offset: 0x9 */
} LLWU_Type, *LLWU_MemMapPtr;

/*
-----  

-- LLWU - Register accessor macros  

-----  

----- */

/*!
 * @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor
macros
 * @{
 */

```

```

/* LLWU - Register accessors */
#define LLWU_PE1_REG(base) ((base)->PE1)
#define LLWU_PE2_REG(base) ((base)->PE2)
#define LLWU_PE3_REG(base) ((base)->PE3)
#define LLWU_PE4_REG(base) ((base)->PE4)
#define LLWU_ME_REG(base) ((base)->ME)
#define LLWU_F1_REG(base) ((base)->F1)
#define LLWU_F2_REG(base) ((base)->F2)
#define LLWU_F3_REG(base) ((base)->F3)
#define LLWU_FILT1_REG(base) ((base)->FILT1)
#define LLWU_FILT2_REG(base) ((base)->FILT2)

/* !
 * @}
 */ /* end of group LLWU_Register_Accessor_Macros */

/* -----
-- LLWU Register Masks
-----
*/
/* !
 * @addtogroup LLWU_Register_Masks LLWU Register Masks
 * @{
 */

/* PE1 Bit Fields */
#define LLWU_PE1_WUPE0_MASK 0x3u
#define LLWU_PE1_WUPE0_SHIFT 0
#define LLWU_PE1_WUPE0_WIDTH 2
#define LLWU_PE1_WUPE0(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE1_WUPE0_SHIFT))&LLWU_PE1_WUPE0_MASK
#define LLWU_PE1_WUPE1_MASK 0xCu
#define LLWU_PE1_WUPE1_SHIFT 2
#define LLWU_PE1_WUPE1_WIDTH 2
#define LLWU_PE1_WUPE1(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE1_WUPE1_SHIFT))&LLWU_PE1_WUPE1_MASK
#define LLWU_PE1_WUPE2_MASK 0x30u
#define LLWU_PE1_WUPE2_SHIFT 4
#define LLWU_PE1_WUPE2_WIDTH 2
#define LLWU_PE1_WUPE2(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE1_WUPE2_SHIFT))&LLWU_PE1_WUPE2_MASK
#define LLWU_PE1_WUPE3_MASK 0xC0u
#define LLWU_PE1_WUPE3_SHIFT 6
#define LLWU_PE1_WUPE3_WIDTH 2
#define LLWU_PE1_WUPE3(x) (((uint8_t)((uint8_t)(x))<<LLWU_PE1_WUPE3_SHIFT))&LLWU_PE1_WUPE3_MASK
/* PE2 Bit Fields */
#define LLWU_PE2_WUPE4_MASK 0x3u
#define LLWU_PE2_WUPE4_SHIFT 0
#define LLWU_PE2_WUPE4_WIDTH 2

```

```

#define LLWU_PE2_WUPE4(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE2_WUPE4_SHIFT))&LLWU_PE2_WUPE4_MASK)
#define LLWU_PE2_WUPE5_MASK 0xCu
#define LLWU_PE2_WUPE5_SHIFT 2
#define LLWU_PE2_WUPE5_WIDTH 2
#define LLWU_PE2_WUPE5(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE2_WUPE5_SHIFT))&LLWU_PE2_WUPE5_MASK)
#define LLWU_PE2_WUPE6_MASK 0x30u
#define LLWU_PE2_WUPE6_SHIFT 4
#define LLWU_PE2_WUPE6_WIDTH 2
#define LLWU_PE2_WUPE6(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE2_WUPE6_SHIFT))&LLWU_PE2_WUPE6_MASK)
#define LLWU_PE2_WUPE7_MASK 0xC0u
#define LLWU_PE2_WUPE7_SHIFT 6
#define LLWU_PE2_WUPE7_WIDTH 2
#define LLWU_PE2_WUPE7(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE2_WUPE7_SHIFT))&LLWU_PE2_WUPE7_MASK)
/* PE3 Bit Fields */
#define LLWU_PE3_WUPE8_MASK 0x3u
#define LLWU_PE3_WUPE8_SHIFT 0
#define LLWU_PE3_WUPE8_WIDTH 2
#define LLWU_PE3_WUPE8(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE8_SHIFT))&LLWU_PE3_WUPE8_MASK)
#define LLWU_PE3_WUPE9_MASK 0xCu
#define LLWU_PE3_WUPE9_SHIFT 2
#define LLWU_PE3_WUPE9_WIDTH 2
#define LLWU_PE3_WUPE9(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE9_SHIFT))&LLWU_PE3_WUPE9_MASK)
#define LLWU_PE3_WUPE10_MASK 0x30u
#define LLWU_PE3_WUPE10_SHIFT 4
#define LLWU_PE3_WUPE10_WIDTH 2
#define LLWU_PE3_WUPE10(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE10_SHIFT))&LLWU_PE3_WUPE10_MASK)
#define LLWU_PE3_WUPE11_MASK 0xC0u
#define LLWU_PE3_WUPE11_SHIFT 6
#define LLWU_PE3_WUPE11_WIDTH 2
#define LLWU_PE3_WUPE11(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE3_WUPE11_SHIFT))&LLWU_PE3_WUPE11_MASK)
/* PE4 Bit Fields */
#define LLWU_PE4_WUPE12_MASK 0x3u
#define LLWU_PE4_WUPE12_SHIFT 0
#define LLWU_PE4_WUPE12_WIDTH 2
#define LLWU_PE4_WUPE12(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE12_SHIFT))&LLWU_PE4_WUPE12_MASK)
#define LLWU_PE4_WUPE13_MASK 0xCu
#define LLWU_PE4_WUPE13_SHIFT 2
#define LLWU_PE4_WUPE13_WIDTH 2
#define LLWU_PE4_WUPE13(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE13_SHIFT))&LLWU_PE4_WUPE13_MASK)
#define LLWU_PE4_WUPE14_MASK 0x30u
#define LLWU_PE4_WUPE14_SHIFT 4
#define LLWU_PE4_WUPE14_WIDTH 2
#define LLWU_PE4_WUPE14(x)
(((uint8_t)((uint8_t)(x))<<LLWU_PE4_WUPE14_SHIFT))&LLWU_PE4_WUPE14_MASK)

```

```

#define LLWU_PE4_WUPE15_MASK          0xC0u
#define LLWU_PE4_WUPE15_SHIFT         6
#define LLWU_PE4_WUPE15_WIDTH          2
#define LLWU_PE4_WUPE15(x) (((uint8_t)(x))<<LLWU_PE4_WUPE15_SHIFT) & LLWU_PE4_WUPE15_MASK
/* ME Bit Fields */
#define LLWU_ME_WUME0_MASK            0x1u
#define LLWU_ME_WUME0_SHIFT           0
#define LLWU_ME_WUME0_WIDTH           1
#define LLWU_ME_WUME0(x) (((uint8_t)(x))<<LLWU_ME_WUME0_SHIFT) & LLWU_ME_WUME0_MASK
#define LLWU_ME_WUME1_MASK            0x2u
#define LLWU_ME_WUME1_SHIFT           1
#define LLWU_ME_WUME1_WIDTH           1
#define LLWU_ME_WUME1(x) (((uint8_t)(x))<<LLWU_ME_WUME1_SHIFT) & LLWU_ME_WUME1_MASK
#define LLWU_ME_WUME2_MASK            0x4u
#define LLWU_ME_WUME2_SHIFT           2
#define LLWU_ME_WUME2_WIDTH           1
#define LLWU_ME_WUME2(x) (((uint8_t)(x))<<LLWU_ME_WUME2_SHIFT) & LLWU_ME_WUME2_MASK
#define LLWU_ME_WUME3_MASK            0x8u
#define LLWU_ME_WUME3_SHIFT           3
#define LLWU_ME_WUME3_WIDTH           1
#define LLWU_ME_WUME3(x) (((uint8_t)(x))<<LLWU_ME_WUME3_SHIFT) & LLWU_ME_WUME3_MASK
#define LLWU_ME_WUME4_MASK            0x10u
#define LLWU_ME_WUME4_SHIFT           4
#define LLWU_ME_WUME4_WIDTH           1
#define LLWU_ME_WUME4(x) (((uint8_t)(x))<<LLWU_ME_WUME4_SHIFT) & LLWU_ME_WUME4_MASK
#define LLWU_ME_WUME5_MASK            0x20u
#define LLWU_ME_WUME5_SHIFT           5
#define LLWU_ME_WUME5_WIDTH           1
#define LLWU_ME_WUME5(x) (((uint8_t)(x))<<LLWU_ME_WUME5_SHIFT) & LLWU_ME_WUME5_MASK
#define LLWU_ME_WUME6_MASK            0x40u
#define LLWU_ME_WUME6_SHIFT           6
#define LLWU_ME_WUME6_WIDTH           1
#define LLWU_ME_WUME6(x) (((uint8_t)(x))<<LLWU_ME_WUME6_SHIFT) & LLWU_ME_WUME6_MASK
#define LLWU_ME_WUME7_MASK            0x80u
#define LLWU_ME_WUME7_SHIFT           7
#define LLWU_ME_WUME7_WIDTH           1
#define LLWU_ME_WUME7(x) (((uint8_t)(x))<<LLWU_ME_WUME7_SHIFT) & LLWU_ME_WUME7_MASK
/* F1 Bit Fields */
#define LLWU_F1_WUF0_MASK             0x1u
#define LLWU_F1_WUF0_SHIFT            0
#define LLWU_F1_WUF0_WIDTH            1
#define LLWU_F1_WUF0(x) (((uint8_t)(x))<<LLWU_F1_WUF0_SHIFT) & LLWU_F1_WUF0_MASK
#define LLWU_F1_WUF1_MASK             0x2u
#define LLWU_F1_WUF1_SHIFT            1

```

```

#define LLWU_F1_WUF1_WIDTH 1
#define LLWU_F1_WUF1(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF1_SHIFT))&LLWU_F1_WUF1_MASK)
#define LLWU_F1_WUF2_MASK 0x4u
#define LLWU_F1_WUF2_SHIFT 2
#define LLWU_F1_WUF2_WIDTH 1
#define LLWU_F1_WUF2(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF2_SHIFT))&LLWU_F1_WUF2_MASK)
#define LLWU_F1_WUF3_MASK 0x8u
#define LLWU_F1_WUF3_SHIFT 3
#define LLWU_F1_WUF3_WIDTH 1
#define LLWU_F1_WUF3(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF3_SHIFT))&LLWU_F1_WUF3_MASK)
#define LLWU_F1_WUF4_MASK 0x10u
#define LLWU_F1_WUF4_SHIFT 4
#define LLWU_F1_WUF4_WIDTH 1
#define LLWU_F1_WUF4(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF4_SHIFT))&LLWU_F1_WUF4_MASK)
#define LLWU_F1_WUF5_MASK 0x20u
#define LLWU_F1_WUF5_SHIFT 5
#define LLWU_F1_WUF5_WIDTH 1
#define LLWU_F1_WUF5(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF5_SHIFT))&LLWU_F1_WUF5_MASK)
#define LLWU_F1_WUF6_MASK 0x40u
#define LLWU_F1_WUF6_SHIFT 6
#define LLWU_F1_WUF6_WIDTH 1
#define LLWU_F1_WUF6(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF6_SHIFT))&LLWU_F1_WUF6_MASK)
#define LLWU_F1_WUF7_MASK 0x80u
#define LLWU_F1_WUF7_SHIFT 7
#define LLWU_F1_WUF7_WIDTH 1
#define LLWU_F1_WUF7(x) (((uint8_t)((uint8_t)(x))<<LLWU_F1_WUF7_SHIFT))&LLWU_F1_WUF7_MASK)
/* F2 Bit Fields */
#define LLWU_F2_WUF8_MASK 0x1u
#define LLWU_F2_WUF8_SHIFT 0
#define LLWU_F2_WUF8_WIDTH 1
#define LLWU_F2_WUF8(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF8_SHIFT))&LLWU_F2_WUF8_MASK)
#define LLWU_F2_WUF9_MASK 0x2u
#define LLWU_F2_WUF9_SHIFT 1
#define LLWU_F2_WUF9_WIDTH 1
#define LLWU_F2_WUF9(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF9_SHIFT))&LLWU_F2_WUF9_MASK)
#define LLWU_F2_WUF10_MASK 0x4u
#define LLWU_F2_WUF10_SHIFT 2
#define LLWU_F2_WUF10_WIDTH 1
#define LLWU_F2_WUF10(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF10_SHIFT))&LLWU_F2_WUF10_MASK)
#define LLWU_F2_WUF11_MASK 0x8u
#define LLWU_F2_WUF11_SHIFT 3
#define LLWU_F2_WUF11_WIDTH 1
#define LLWU_F2_WUF11(x) (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF11_SHIFT))&LLWU_F2_WUF11_MASK)

```

```

#define LLWU_F2_WUF12_MASK          0x10u
#define LLWU_F2_WUF12_SHIFT         4
#define LLWU_F2_WUF12_WIDTH          1
#define LLWU_F2_WUF12(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF12_SHIFT))&LLWU_F2_WUF12_MASK)
#define LLWU_F2_WUF13_MASK          0x20u
#define LLWU_F2_WUF13_SHIFT         5
#define LLWU_F2_WUF13_WIDTH          1
#define LLWU_F2_WUF13(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF13_SHIFT))&LLWU_F2_WUF13_MASK)
#define LLWU_F2_WUF14_MASK          0x40u
#define LLWU_F2_WUF14_SHIFT         6
#define LLWU_F2_WUF14_WIDTH          1
#define LLWU_F2_WUF14(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF14_SHIFT))&LLWU_F2_WUF14_MASK)
#define LLWU_F2_WUF15_MASK          0x80u
#define LLWU_F2_WUF15_SHIFT         7
#define LLWU_F2_WUF15_WIDTH          1
#define LLWU_F2_WUF15(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F2_WUF15_SHIFT))&LLWU_F2_WUF15_MASK)
/* F3 Bit Fields */
#define LLWU_F3_MWUF0_MASK          0x1u
#define LLWU_F3_MWUF0_SHIFT         0
#define LLWU_F3_MWUF0_WIDTH          1
#define LLWU_F3_MWUF0(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF0_SHIFT))&LLWU_F3_MWUF0_MASK)
#define LLWU_F3_MWUF1_MASK          0x2u
#define LLWU_F3_MWUF1_SHIFT         1
#define LLWU_F3_MWUF1_WIDTH          1
#define LLWU_F3_MWUF1(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF1_SHIFT))&LLWU_F3_MWUF1_MASK)
#define LLWU_F3_MWUF2_MASK          0x4u
#define LLWU_F3_MWUF2_SHIFT         2
#define LLWU_F3_MWUF2_WIDTH          1
#define LLWU_F3_MWUF2(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF2_SHIFT))&LLWU_F3_MWUF2_MASK)
#define LLWU_F3_MWUF3_MASK          0x8u
#define LLWU_F3_MWUF3_SHIFT         3
#define LLWU_F3_MWUF3_WIDTH          1
#define LLWU_F3_MWUF3(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF3_SHIFT))&LLWU_F3_MWUF3_MASK)
#define LLWU_F3_MWUF4_MASK          0x10u
#define LLWU_F3_MWUF4_SHIFT         4
#define LLWU_F3_MWUF4_WIDTH          1
#define LLWU_F3_MWUF4(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF4_SHIFT))&LLWU_F3_MWUF4_MASK)
#define LLWU_F3_MWUF5_MASK          0x20u
#define LLWU_F3_MWUF5_SHIFT         5
#define LLWU_F3_MWUF5_WIDTH          1
#define LLWU_F3_MWUF5(x)
    (((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF5_SHIFT))&LLWU_F3_MWUF5_MASK)
#define LLWU_F3_MWUF6_MASK          0x40u
#define LLWU_F3_MWUF6_SHIFT         6
#define LLWU_F3_MWUF6_WIDTH          1

```

```

#define LLWU_F3_MWUF6(x)
(((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF6_SHIFT))&LLWU_F3_MWUF6_MASK)
#define LLWU_F3_MWUF7_MASK 0x80u
#define LLWU_F3_MWUF7_SHIFT 7
#define LLWU_F3_MWUF7_WIDTH 1
#define LLWU_F3_MWUF7(x)
(((uint8_t)((uint8_t)(x))<<LLWU_F3_MWUF7_SHIFT))&LLWU_F3_MWUF7_MASK)
/* FILT1 Bit Fields */
#define LLWU_FILT1_FILTSEL_MASK 0xFFu
#define LLWU_FILT1_FILTSEL_SHIFT 0
#define LLWU_FILT1_FILTSEL_WIDTH 4
#define LLWU_FILT1_FILTSEL(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTSEL_SHIFT))&LLWU_FILT1_FILTSEL
_MASK)
#define LLWU_FILT1_FILTE_MASK 0x60u
#define LLWU_FILT1_FILTE_SHIFT 5
#define LLWU_FILT1_FILTE_WIDTH 2
#define LLWU_FILT1_FILTE(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTE_SHIFT))&LLWU_FILT1_FILTE_MAS
K)
#define LLWU_FILT1_FILTF_MASK 0x80u
#define LLWU_FILT1_FILTF_SHIFT 7
#define LLWU_FILT1_FILTF_WIDTH 1
#define LLWU_FILT1_FILTF(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT1_FILTF_SHIFT))&LLWU_FILT1_FILTF_MAS
K)
/* FILT2 Bit Fields */
#define LLWU_FILT2_FILTSEL_MASK 0xFFu
#define LLWU_FILT2_FILTSEL_SHIFT 0
#define LLWU_FILT2_FILTSEL_WIDTH 4
#define LLWU_FILT2_FILTSEL(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT2_FILTSEL_SHIFT))&LLWU_FILT2_FILTSEL
_MASK)
#define LLWU_FILT2_FILTE_MASK 0x60u
#define LLWU_FILT2_FILTE_SHIFT 5
#define LLWU_FILT2_FILTE_WIDTH 2
#define LLWU_FILT2_FILTE(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT2_FILTE_SHIFT))&LLWU_FILT2_FILTE_MAS
K)
#define LLWU_FILT2_FILTF_MASK 0x80u
#define LLWU_FILT2_FILTF_SHIFT 7
#define LLWU_FILT2_FILTF_WIDTH 1
#define LLWU_FILT2_FILTF(x)
(((uint8_t)((uint8_t)(x))<<LLWU_FILT2_FILTF_SHIFT))&LLWU_FILT2_FILTF_MAS
K)

/* !
 * @}
 */
/* end of group LLWU_Register_Masks */

/* LLWU - Peripheral instance base addresses */
/** Peripheral LLWU base address */
#define LLWU_BASE (0x4007C000u)

```

```

/** Peripheral LLWU base pointer */
#define LLWU ((LLWU_Type *)LLWU_BASE)
#define LLWU_BASE_PTR (LLWU)
/** Array initializer of LLWU peripheral base addresses */
#define LLWU_BASE_ADDRS { LLWU_BASE }
/** Array initializer of LLWU peripheral base pointers */
#define LLWU_BASE_PTRS { LLWU }
/** Interrupt vectors for the LLWU peripheral type */
#define LLWU IRQs { LLWU_IRQn }

/* -----
-- LLWU - Register accessor macros
-----
*/
/*!
* @addtogroup LLWU_Register_Accessor_Macros LLWU - Register accessor
macros
* @{
*/
/* LLWU - Register instance definitions */
/* LLWU */
#define LLWU_PE1 LLWU_PE1_REG(LLWU)
#define LLWU_PE2 LLWU_PE2_REG(LLWU)
#define LLWU_PE3 LLWU_PE3_REG(LLWU)
#define LLWU_PE4 LLWU_PE4_REG(LLWU)
#define LLWU_ME LLWU_ME_REG(LLWU)
#define LLWU_F1 LLWU_F1_REG(LLWU)
#define LLWU_F2 LLWU_F2_REG(LLWU)
#define LLWU_F3 LLWU_F3_REG(LLWU)
#define LLWU_FILTER1 LLWU_FILTER1_REG(LLWU)
#define LLWU_FILTER2 LLWU_FILTER2_REG(LLWU)

/*!
* @}
*/
/* end of group LLWU_Register_Accessor_Macros */

/*!
* @}
*/
/* end of group LLWU_Peripheral_Access_Layer */

/* -----
-- LPTMR Peripheral Access Layer
-----
*/
/*!

```

```

 * @addtogroup LPTMR_Peripheral_Access_Layer LPTMR Peripheral Access
Layer
 * @{
 */

/** LPTMR - Register Layout Typedef */
typedef struct {
    __IO uint32_t CSR;                                /**< Low Power Timer
Control Status Register, offset: 0x0 */
    __IO uint32_t PSR;                                /**< Low Power Timer
Prescale Register, offset: 0x4 */
    __IO uint32_t CMR;                                /**< Low Power Timer
Compare Register, offset: 0x8 */
    __IO uint32_t CNR;                                /**< Low Power Timer
Counter Register, offset: 0xC */
} LPTMR_Type, *LPTMR_MemMapPtr;

/* -----
-- LPTMR - Register accessor macros
-----
 */

/*!
 * @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor
macros
 * @{
 */

/* LPTMR - Register accessors */
#define LPTMR_CSR_REG(base)          ((base) -> CSR)
#define LPTMR_PSR_REG(base)          ((base) -> PSR)
#define LPTMR_CMR_REG(base)          ((base) -> CMR)
#define LPTMR_CNR_REG(base)          ((base) -> CNR)

/*!
 * @}
 */
/* end of group LPTMR_Register_Accessor_Macros */

/* -----
-- LPTMR Register Masks
-----
 */

/*!
 * @addtogroup LPTMR_Register_Masks LPTMR Register Masks
 * @{
 */

/* CSR Bit Fields */
#define LPTMR_CSR_TEN_MASK           0x1u

```

```

#define LPTMR_CSR_TEN_SHIFT 0
#define LPTMR_CSR_TEN_WIDTH 1
#define LPTMR_CSR_TEN(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR_TEN_SHIFT))&LPTMR_CSR_TEN_MASK
#define LPTMR_CSR_TMS_MASK 0x2u
#define LPTMR_CSR_TMS_SHIFT 1
#define LPTMR_CSR_TMS_WIDTH 1
#define LPTMR_CSR_TMS(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR_TMS_SHIFT))&LPTMR_CSR_TMS_MASK
#define LPTMR_CSR_TFC_MASK 0x4u
#define LPTMR_CSR_TFC_SHIFT 2
#define LPTMR_CSR_TFC_WIDTH 1
#define LPTMR_CSR_TFC(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR_TFC_SHIFT))&LPTMR_CSR_TFC_MASK
#define LPTMR_CSR TPP MASK 0x8u
#define LPTMR_CSR TPP SHIFT 3
#define LPTMR_CSR TPP WIDTH 1
#define LPTMR_CSR TPP(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR TPP SHIFT))&LPTMR_CSR TPP MASK
#define LPTMR_CSR TPS MASK 0x30u
#define LPTMR_CSR TPS SHIFT 4
#define LPTMR_CSR TPS WIDTH 2
#define LPTMR_CSR TPS(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR TPS SHIFT))&LPTMR_CSR TPS MASK
#define LPTMR_CSR TIE MASK 0x40u
#define LPTMR_CSR TIE SHIFT 6
#define LPTMR_CSR TIE WIDTH 1
#define LPTMR_CSR TIE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR TIE SHIFT))&LPTMR_CSR TIE MASK
#define LPTMR_CSR TCF MASK 0x80u
#define LPTMR_CSR TCF SHIFT 7
#define LPTMR_CSR TCF WIDTH 1
#define LPTMR_CSR TCF(x) (((uint32_t)((uint32_t)(x))<<LPTMR_CSR TCF SHIFT))&LPTMR_CSR TCF MASK
/* PSR Bit Fields */
#define LPTMR_PSR PCS MASK 0x3u
#define LPTMR_PSR PCS SHIFT 0
#define LPTMR_PSR PCS WIDTH 2
#define LPTMR_PSR PCS(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR PCS SHIFT))&LPTMR_PSR PCS MASK
#define LPTMR_PSR PBYP MASK 0x4u
#define LPTMR_PSR PBYP SHIFT 2
#define LPTMR_PSR PBYP WIDTH 1
#define LPTMR_PSR PBYP(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR PBYP SHIFT))&LPTMR_PSR PBYP MASK
#define LPTMR_PSR PRESCALE MASK 0x78u
#define LPTMR_PSR PRESCALE SHIFT 3
#define LPTMR_PSR PRESCALE WIDTH 4
#define LPTMR_PSR PRESCALE(x) (((uint32_t)((uint32_t)(x))<<LPTMR_PSR PRESCALE SHIFT))&LPTMR_PSR PRESCALE MASK
/* CMR Bit Fields */
#define LPTMR_CMR_COMPARE MASK 0xFFFFu
#define LPTMR_CMR_COMPARE SHIFT 0

```

```

#define LPTMR_CMRA_COMPARE_WIDTH          16
#define LPTMR_CMRA_COMPARE(x)             (((uint32_t)((uint32_t)(x))<<LPTMR_CMRA_COMPARE_SHIFT))&LPTMR_CMRA_COMPARE_MASK
/* CNR Bit Fields */
#define LPTMR_CNR_COUNTER_MASK           0xFFFFu
#define LPTMR_CNR_COUNTER_SHIFT          0
#define LPTMR_CNR_COUNTER_WIDTH          16
#define LPTMR_CNR_COUNTER(x)             (((uint32_t)((uint32_t)(x))<<LPTMR_CNR_COUNTER_SHIFT))&LPTMR_CNR_COUNTER_MASK

/* !
 * @}
 */ /* end of group LPTMR_Register_Masks */

/* LPTMR - Peripheral instance base addresses */
/** Peripheral LPTMR0 base address */
#define LPTMR0_BASE                      (0x40040000u)
/** Peripheral LPTMR0 base pointer */
#define LPTMR0                           ((LPTMR_Type *)LPTMR0_BASE)
#define LPTMR0_BASE_PTR                  (LPTMR0)
/** Array initializer of LPTMR peripheral base addresses */
#define LPTMR_BASE_ADDRS                { LPTMR0_BASE }
/** Array initializer of LPTMR peripheral base pointers */
#define LPTMR_BASE_PTRS                 { LPTMR0 }
/** Interrupt vectors for the LPTMR peripheral type */
#define LPTMR IRQS                      { LPTMR0_IRQn }

/* -----
-- LPTMR - Register accessor macros
-----
*/
/* !
 * @addtogroup LPTMR_Register_Accessor_Macros LPTMR - Register accessor macros
 * @{
 */

/* LPTMR - Register instance definitions */
/* LPTMR0 */
#define LPTMR0_CSR                      LPTMR_CSR_REG(LPTMR0)
#define LPTMR0_PSR                      LPTMR_PSR_REG(LPTMR0)
#define LPTMR0_CMRA                     LPTMR_CMRA_REG(LPTMR0)
#define LPTMR0_CNR                      LPTMR_CNR_REG(LPTMR0)

/* !
 * @}
 */ /* end of group LPTMR_Register_Accessor_Macros */

```

```

/*!
 * @}
 */
/* end of group LPTMR_Peripheral_Access_Layer */

/*
-----  

-- MCG Peripheral Access Layer  

-----  

----- */

/*!
 * @addtogroup MCG_Peripheral_Access_Layer MCG Peripheral Access Layer
 * @{
 */

/** MCG - Register Layout Typedef */
typedef struct {
    __IO uint8_t C1;                                /**< MCG Control 1
Register, offset: 0x0 */
    __IO uint8_t C2;                                /**< MCG Control 2
Register, offset: 0x1 */
    __IO uint8_t C3;                                /**< MCG Control 3
Register, offset: 0x2 */
    __IO uint8_t C4;                                /**< MCG Control 4
Register, offset: 0x3 */
    __IO uint8_t C5;                                /**< MCG Control 5
Register, offset: 0x4 */
    __IO uint8_t C6;                                /**< MCG Control 6
Register, offset: 0x5 */
    __IO uint8_t S;                                 /**< MCG Status
Register, offset: 0x6 */
    uint8_t RESERVED_0[1];
    __IO uint8_t SC;                                /**< MCG Status and
Control Register, offset: 0x8 */
    uint8_t RESERVED_1[1];
    __IO uint8_t ATCVH;                            /**< MCG Auto Trim
Compare Value High Register, offset: 0xA */
    __IO uint8_t ATCVL;                            /**< MCG Auto Trim
Compare Value Low Register, offset: 0xB */
    __I  uint8_t C7;                                /**< MCG Control 7
Register, offset: 0xC */
    __IO uint8_t C8;                                /**< MCG Control 8
Register, offset: 0xD */
    __I  uint8_t C9;                                /**< MCG Control 9
Register, offset: 0xE */
    __I  uint8_t C10;                             /**< MCG Control 10
Register, offset: 0xF */
} MCG_Type, *MCG_MemMapPtr;

/*
-----  

-----
```

```

-- MCG - Register accessor macros
-----
----- */

/*!
 * @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor
macros
 * @{
 */

/* MCG - Register accessors */
#define MCG_C1_REG(base) ((base)->C1)
#define MCG_C2_REG(base) ((base)->C2)
#define MCG_C3_REG(base) ((base)->C3)
#define MCG_C4_REG(base) ((base)->C4)
#define MCG_C5_REG(base) ((base)->C5)
#define MCG_C6_REG(base) ((base)->C6)
#define MCG_S_REG(base) ((base)->S)
#define MCG_SC_REG(base) ((base)->SC)
#define MCG_ATCVH_REG(base) ((base)->ATCVH)
#define MCG_ATCVL_REG(base) ((base)->ATCVL)
#define MCG_C7_REG(base) ((base)->C7)
#define MCG_C8_REG(base) ((base)->C8)
#define MCG_C9_REG(base) ((base)->C9)
#define MCG_C10_REG(base) ((base)->C10)

/*!
 * @}
 */
/* end of group MCG_Register_Accessor_Macros */

/*
-----
-- MCG Register Masks
-----
----- */

/*!
 * @addtogroup MCG_Register_Masks MCG Register Masks
 * @{
 */

/* C1 Bit Fields */
#define MCG_C1_IREFSTEN_MASK 0x1u
#define MCG_C1_IREFSTEN_SHIFT 0
#define MCG_C1_IREFSTEN_WIDTH 1
#define MCG_C1_IREFSTEN(x) (((uint8_t)((uint8_t)(x)) << MCG_C1_IREFSTEN_SHIFT) & MCG_C1_IREFSTEN_MASK)
#define MCG_C1_IRCLKEN_MASK 0x2u
#define MCG_C1_IRCLKEN_SHIFT 1
#define MCG_C1_IRCLKEN_WIDTH 1
#define MCG_C1_IRCLKEN(x) (((uint8_t)((uint8_t)(x)) << MCG_C1_IRCLKEN_SHIFT) & MCG_C1_IRCLKEN_MASK)

```

```

#define MCG_C1_IREFS_MASK          0x4u
#define MCG_C1_IREFS_SHIFT         2
#define MCG_C1_IREFS_WIDTH         1
#define MCG_C1_IREFS(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C1_IREFS_SHIFT) & MCG_C1_IREFS_MASK)
#define MCG_C1_FRDIV_MASK          0x38u
#define MCG_C1_FRDIV_SHIFT         3
#define MCG_C1_FRDIV_WIDTH         3
#define MCG_C1_FRDIV(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C1_FRDIV_SHIFT) & MCG_C1_FRDIV_MASK)
#define MCG_C1_CLKS_MASK            0xC0u
#define MCG_C1_CLKS_SHIFT           6
#define MCG_C1_CLKS_WIDTH           2
#define MCG_C1_CLKS(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C1_CLKS_SHIFT) & MCG_C1_CLKS_MASK)
/* C2 Bit Fields */
#define MCG_C2_IRCS_MASK            0x1u
#define MCG_C2_IRCS_SHIFT           0
#define MCG_C2_IRCS_WIDTH           1
#define MCG_C2_IRCS(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_IRCS_SHIFT) & MCG_C2_IRCS_MASK)
#define MCG_C2_LP_MASK              0x2u
#define MCG_C2_LP_SHIFT             1
#define MCG_C2_LP_WIDTH             1
#define MCG_C2_LP(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_LP_SHIFT) & MCG_C2_LP_MASK)
#define MCG_C2_EREFS0_MASK           0x4u
#define MCG_C2_EREFS0_SHIFT          2
#define MCG_C2_EREFS0_WIDTH          1
#define MCG_C2_EREFS0(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_EREFS0_SHIFT) & MCG_C2_EREFS0_MASK)
#define MCG_C2_HGO0_MASK             0x8u
#define MCG_C2_HGO0_SHIFT            3
#define MCG_C2_HGO0_WIDTH            1
#define MCG_C2_HGO0(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_HGO0_SHIFT) & MCG_C2_HGO0_MASK)
#define MCG_C2_RANGE0_MASK           0x30u
#define MCG_C2_RANGE0_SHIFT          4
#define MCG_C2_RANGE0_WIDTH          2
#define MCG_C2_RANGE0(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_RANGE0_SHIFT) & MCG_C2_RANGE0_MASK)
#define MCG_C2_LOCRE0_MASK           0x80u
#define MCG_C2_LOCRE0_SHIFT          7
#define MCG_C2_LOCRE0_WIDTH          1
#define MCG_C2_LOCRE0(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C2_LOCRE0_SHIFT) & MCG_C2_LOCRE0_MASK)
/* C3 Bit Fields */
#define MCG_C3_SCTRIM_MASK           0xFFu
#define MCG_C3_SCTRIM_SHIFT          0
#define MCG_C3_SCTRIM_WIDTH          8
#define MCG_C3_SCTRIM(x)
  (((uint8_t) ((uint8_t)(x)) << MCG_C3_SCTRIM_SHIFT) & MCG_C3_SCTRIM_MASK)
/* C4 Bit Fields */
#define MCG_C4_SCFTRIM_MASK           0x1u

```

```

#define MCG_C4_SCFTRIM_SHIFT 0
#define MCG_C4_SCFTRIM_WIDTH 1
#define MCG_C4_SCFTRIM(x) (((uint8_t)((uint8_t)(x))<<MCG_C4_SCFTRIM_SHIFT))&MCG_C4_SCFTRIM_MASK)
#define MCG_C4_FCTRIM_MASK 0x1Eu
#define MCG_C4_FCTRIM_SHIFT 1
#define MCG_C4_FCTRIM_WIDTH 4
#define MCG_C4_FCTRIM(x) (((uint8_t)((uint8_t)(x))<<MCG_C4_FCTRIM_SHIFT))&MCG_C4_FCTRIM_MASK)
#define MCG_C4_DRST_DRS_MASK 0x60u
#define MCG_C4_DRST_DRS_SHIFT 5
#define MCG_C4_DRST_DRS_WIDTH 2
#define MCG_C4_DRST_DRS(x) (((uint8_t)((uint8_t)(x))<<MCG_C4_DRST_DRS_SHIFT))&MCG_C4_DRST_DRS_MASK)
#define MCG_C4_DMX32_MASK 0x80u
#define MCG_C4_DMX32_SHIFT 7
#define MCG_C4_DMX32_WIDTH 1
#define MCG_C4_DMX32(x) (((uint8_t)((uint8_t)(x))<<MCG_C4_DMX32_SHIFT))&MCG_C4_DMX32_MASK)
/* C5 Bit Fields */
#define MCG_C5_PRDIV0_MASK 0x1Fu
#define MCG_C5_PRDIV0_SHIFT 0
#define MCG_C5_PRDIV0_WIDTH 5
#define MCG_C5_PRDIV0(x) (((uint8_t)((uint8_t)(x))<<MCG_C5_PRDIV0_SHIFT))&MCG_C5_PRDIV0_MASK)
#define MCG_C5_PLLSTENO_MASK 0x20u
#define MCG_C5_PLLSTENO_SHIFT 5
#define MCG_C5_PLLSTENO_WIDTH 1
#define MCG_C5_PLLSTENO(x) (((uint8_t)((uint8_t)(x))<<MCG_C5_PLLSTENO_SHIFT))&MCG_C5_PLLSTENO_MASK)
#define MCG_C5_PLLCLKENO_MASK 0x40u
#define MCG_C5_PLLCLKENO_SHIFT 6
#define MCG_C5_PLLCLKENO_WIDTH 1
#define MCG_C5_PLLCLKENO(x) (((uint8_t)((uint8_t)(x))<<MCG_C5_PLLCLKENO_SHIFT))&MCG_C5_PLLCLKENO_MASK)
/* C6 Bit Fields */
#define MCG_C6_VDIV0_MASK 0x1Fu
#define MCG_C6_VDIV0_SHIFT 0
#define MCG_C6_VDIV0_WIDTH 5
#define MCG_C6_VDIV0(x) (((uint8_t)((uint8_t)(x))<<MCG_C6_VDIV0_SHIFT))&MCG_C6_VDIV0_MASK)
#define MCG_C6_CME0_MASK 0x20u
#define MCG_C6_CME0_SHIFT 5
#define MCG_C6_CME0_WIDTH 1
#define MCG_C6_CME0(x) (((uint8_t)((uint8_t)(x))<<MCG_C6_CME0_SHIFT))&MCG_C6_CME0_MASK)
#define MCG_C6_PLLS_MASK 0x40u
#define MCG_C6_PLLS_SHIFT 6
#define MCG_C6_PLLS_WIDTH 1
#define MCG_C6_PLLS(x) (((uint8_t)((uint8_t)(x))<<MCG_C6_PLLS_SHIFT))&MCG_C6_PLLS_MASK)
#define MCG_C6_LOLIE0_MASK 0x80u
#define MCG_C6_LOLIE0_SHIFT 7

```

```

#define MCG_C6_LOLIE0_WIDTH 1
#define MCG_C6_LOLIE0(x) (((uint8_t)((uint8_t)(x))<<MCG_C6_LOLIE0_SHIFT))&MCG_C6_LOLIE0_MASK)
/* S Bit Fields */
#define MCG_S_IRCST_MASK 0x1u
#define MCG_S_IRCST_SHIFT 0
#define MCG_S_IRCST_WIDTH 1
#define MCG_S_IRCST(x) (((uint8_t)((uint8_t)(x))<<MCG_S_IRCST_SHIFT))&MCG_S_IRCST_MASK)
#define MCG_S_OSCINIT0_MASK 0x2u
#define MCG_S_OSCINIT0_SHIFT 1
#define MCG_S_OSCINIT0_WIDTH 1
#define MCG_S_OSCINIT0(x) (((uint8_t)((uint8_t)(x))<<MCG_S_OSCINIT0_SHIFT))&MCG_S_OSCINIT0_MASK)
#define MCG_S_CLKST_MASK 0xCu
#define MCG_S_CLKST_SHIFT 2
#define MCG_S_CLKST_WIDTH 2
#define MCG_S_CLKST(x) (((uint8_t)((uint8_t)(x))<<MCG_S_CLKST_SHIFT))&MCG_S_CLKST_MASK)
#define MCG_S_IREFST_MASK 0x10u
#define MCG_S_IREFST_SHIFT 4
#define MCG_S_IREFST_WIDTH 1
#define MCG_S_IREFST(x) (((uint8_t)((uint8_t)(x))<<MCG_S_IREFST_SHIFT))&MCG_S_IREFST_MASK)
#define MCG_S_PLLST_MASK 0x20u
#define MCG_S_PLLST_SHIFT 5
#define MCG_S_PLLST_WIDTH 1
#define MCG_S_PLLST(x) (((uint8_t)((uint8_t)(x))<<MCG_S_PLLST_SHIFT))&MCG_S_PLLST_MASK)
#define MCG_S_LOCK0_MASK 0x40u
#define MCG_S_LOCK0_SHIFT 6
#define MCG_S_LOCK0_WIDTH 1
#define MCG_S_LOCK0(x) (((uint8_t)((uint8_t)(x))<<MCG_S_LOCK0_SHIFT))&MCG_S_LOCK0_MASK)
#define MCG_S_LOLS0_MASK 0x80u
#define MCG_S_LOLS0_SHIFT 7
#define MCG_S_LOLS0_WIDTH 1
#define MCG_S_LOLS0(x) (((uint8_t)((uint8_t)(x))<<MCG_S_LOLS0_SHIFT))&MCG_S_LOLS0_MASK)
/* SC Bit Fields */
#define MCG_SC_LOCS0_MASK 0x1u
#define MCG_SC_LOCS0_SHIFT 0
#define MCG_SC_LOCS0_WIDTH 1
#define MCG_SC_LOCS0(x) (((uint8_t)((uint8_t)(x))<<MCG_SC_LOCS0_SHIFT))&MCG_SC_LOCS0_MASK)
#define MCG_SC_FCRDIV_MASK 0xEu
#define MCG_SC_FCRDIV_SHIFT 1
#define MCG_SC_FCRDIV_WIDTH 3
#define MCG_SC_FCRDIV(x) (((uint8_t)((uint8_t)(x))<<MCG_SC_FCRDIV_SHIFT))&MCG_SC_FCRDIV_MASK)
#define MCG_SC_FLTPRSRV_MASK 0x10u
#define MCG_SC_FLTPRSRV_SHIFT 4
#define MCG_SC_FLTPRSRV_WIDTH 1

```

```

#define MCG_SC_FLTPRSRV(x)
(((uint8_t) (((uint8_t)(x))<<MCG_SC_FLTPRSRV_SHIFT)) & MCG_SC_FLTPRSRV_MASK)
#define MCG_SC_ATMF_MASK 0x20u
#define MCG_SC_ATMF_SHIFT 5
#define MCG_SC_ATMF_WIDTH 1
#define MCG_SC_ATMF(x)
(((uint8_t) (((uint8_t)(x))<<MCG_SC_ATMF_SHIFT)) & MCG_SC_ATMF_MASK)
#define MCG_SC_ATMS_MASK 0x40u
#define MCG_SC_ATMS_SHIFT 6
#define MCG_SC_ATMS_WIDTH 1
#define MCG_SC_ATMS(x)
(((uint8_t) (((uint8_t)(x))<<MCG_SC_ATMS_SHIFT)) & MCG_SC_ATMS_MASK)
#define MCG_SC_ATME_MASK 0x80u
#define MCG_SC_ATME_SHIFT 7
#define MCG_SC_ATME_WIDTH 1
#define MCG_SC_ATME(x)
(((uint8_t) (((uint8_t)(x))<<MCG_SC_ATME_SHIFT)) & MCG_SC_ATME_MASK)
/* ATCVH Bit Fields */
#define MCG_ATCVH_ATCVH_MASK 0xFFu
#define MCG_ATCVH_ATCVH_SHIFT 0
#define MCG_ATCVH_ATCVH_WIDTH 8
#define MCG_ATCVH_ATCVH(x)
(((uint8_t) (((uint8_t)(x))<<MCG_ATCVH_ATCVH_SHIFT)) & MCG_ATCVH_ATCVH_MASK)
/* ATCVL Bit Fields */
#define MCG_ATCVL_ATCVL_MASK 0xFFu
#define MCG_ATCVL_ATCVL_SHIFT 0
#define MCG_ATCVL_ATCVL_WIDTH 8
#define MCG_ATCVL_ATCVL(x)
(((uint8_t) (((uint8_t)(x))<<MCG_ATCVL_ATCVL_SHIFT)) & MCG_ATCVL_ATCVL_MASK)
/* C8 Bit Fields */
#define MCG_C8_LOLRE_MASK 0x40u
#define MCG_C8_LOLRE_SHIFT 6
#define MCG_C8_LOLRE_WIDTH 1
#define MCG_C8_LOLRE(x)
(((uint8_t) (((uint8_t)(x))<<MCG_C8_LOLRE_SHIFT)) & MCG_C8_LOLRE_MASK)

/*!
 * @}
 */ /* end of group MCG_Register_Masks */

```

```

/* MCG - Peripheral instance base addresses */
/** Peripheral MCG base address */
#define MCG_BASE (0x40064000u)
/** Peripheral MCG base pointer */
#define MCG ((MCG_Type *)MCG_BASE)
#define MCG_BASE_PTR (MCG)
/** Array initializer of MCG peripheral base addresses */
#define MCG_BASE_ADDRS { MCG_BASE }
/** Array initializer of MCG peripheral base pointers */
#define MCG_BASE_PTRS { MCG }
/** Interrupt vectors for the MCG peripheral type */
#define MCG IRQS { MCG_IRQn }

```

```

/* -----
-- MCG - Register accessor macros
-----
*/
/*!
 * @addtogroup MCG_Register_Accessor_Macros MCG - Register accessor
macros
 * @{
 */

/* MCG - Register instance definitions */
/* MCG */
#define MCG_C1 MCG_C1_REG (MCG)
#define MCG_C2 MCG_C2_REG (MCG)
#define MCG_C3 MCG_C3_REG (MCG)
#define MCG_C4 MCG_C4_REG (MCG)
#define MCG_C5 MCG_C5_REG (MCG)
#define MCG_C6 MCG_C6_REG (MCG)
#define MCG_S MCG_S_REG (MCG)
#define MCG_SC MCG_SC_REG (MCG)
#define MCG_ATCVH MCG_ATCVH_REG (MCG)
#define MCG_ATCVL MCG_ATCVL_REG (MCG)
#define MCG_C7 MCG_C7_REG (MCG)
#define MCG_C8 MCG_C8_REG (MCG)
#define MCG_C9 MCG_C9_REG (MCG)
#define MCG_C10 MCG_C10_REG (MCG)

/*!
 * @}
 */
/* /* end of group MCG_Register_Accessor_Macros */

/* MCG C2 [EREFS] backward compatibility */
#define MCG_C2_EREFS_MASK (MCG_C2_EREFS0_MASK)
#define MCG_C2_EREFS_SHIFT (MCG_C2_EREFS0_SHIFT)
#define MCG_C2_EREFS_WIDTH (MCG_C2_EREFS0_WIDTH)
#define MCG_C2_EREFS(x) (MCG_C2_EREFS0(x))

/* MCG C2 [HGO] backward compatibility */
#define MCG_C2_HGO_MASK (MCG_C2_HGO0_MASK)
#define MCG_C2_HGO_SHIFT (MCG_C2_HGO0_SHIFT)
#define MCG_C2_HGO_WIDTH (MCG_C2_HGO0_WIDTH)
#define MCG_C2_HGO(x) (MCG_C2_HGO0(x))

/* MCG C2 [RANGE] backward compatibility */
#define MCG_C2_RANGE_MASK (MCG_C2_RANGE0_MASK)
#define MCG_C2_RANGE_SHIFT (MCG_C2_RANGE0_SHIFT)
#define MCG_C2_RANGE_WIDTH (MCG_C2_RANGE0_WIDTH)
#define MCG_C2_RANGE(x) (MCG_C2_RANGE0(x))

/*

```

```

        * @}
    */ /* end of group MCG_Peripheral_Access_Layer */

/*
-----
-- MCM Peripheral Access Layer
-----
*/
/*!
 * @addtogroup MCM_Peripheral_Access_Layer MCM Peripheral Access Layer
 * @{
 */

/** MCM - Register Layout Typedef */
typedef struct {
    uint8_t RESERVED_0[8];
    __I uint16_t PLASC;                                /**< Crossbar Switch
    (AXBS) Slave Configuration, offset: 0x8 */
    __I uint16_t PLAMC;                                /**< Crossbar Switch
    (AXBS) Master Configuration, offset: 0xA */
    __IO uint32_t PLACR;                                /**< Platform Control
    Register, offset: 0xC */
    uint8_t RESERVED_1[48];
    __IO uint32_t CPO;                                  /**< Compute Operation
    Control Register, offset: 0x40 */
} MCM_Type, *MCM_MemMapPtr;

/*
-----
-- MCM - Register accessor macros
-----
*/
/*!
 * @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor
 * macros
 * @{
 */

/* MCM - Register accessors */
#define MCM_PLASC_REG(base)          ((base)->PLASC)
#define MCM_PLAMC_REG(base)          ((base)->PLAMC)
#define MCM_PLACR_REG(base)          ((base)->PLACR)
#define MCM_CPO_REG(base)            ((base)->CPO)

/*!
 * @}
 */
/* end of group MCM_Register_Accessor_Macros */

```

```

/*
-----  

-- MCM Register Masks  

-----  

----- */
  

/*!  

 * @addtogroup MCM_Register_Masks MCM Register Masks
 * @{
 * /  

  

/* PLASC Bit Fields */  

#define MCM_PLASC_ASC_MASK 0xFFu  

#define MCM_PLASC_ASC_SHIFT 0  

#define MCM_PLASC_ASC_WIDTH 8  

#define MCM_PLASC_ASC(x) (((uint16_t)((uint16_t)(x))<<MCM_PLASC_ASC_SHIFT))&MCM_PLASC_ASC_MASK)  

/* PLAMC Bit Fields */  

#define MCM_PLAMC_AMC_MASK 0xFFu  

#define MCM_PLAMC_AMC_SHIFT 0  

#define MCM_PLAMC_AMC_WIDTH 8  

#define MCM_PLAMC_AMC(x) (((uint16_t)((uint16_t)(x))<<MCM_PLAMC_AMC_SHIFT))&MCM_PLAMC_AMC_MASK)  

/* PLACR Bit Fields */  

#define MCM_PLACR_ARB_MASK 0x200u  

#define MCM_PLACR_ARB_SHIFT 9  

#define MCM_PLACR_ARB_WIDTH 1  

#define MCM_PLACR_ARB(x) (((uint32_t)((uint32_t)(x))<<MCM_PLACR_ARB_SHIFT))&MCM_PLACR_ARB_MASK)  

#define MCM_PLACR_CFCC_MASK 0x400u  

#define MCM_PLACR_CFCC_SHIFT 10  

#define MCM_PLACR_CFCC_WIDTH 1  

#define MCM_PLACR_CFCC(x) (((uint32_t)((uint32_t)(x))<<MCM_PLACR_CFCC_SHIFT))&MCM_PLACR_CFCC_MASK)  

#define MCM_PLACR_DFCDA_MASK 0x800u  

#define MCM_PLACR_DFCDA_SHIFT 11  

#define MCM_PLACR_DFCDA_WIDTH 1  

#define MCM_PLACR_DFCDA(x) (((uint32_t)((uint32_t)(x))<<MCM_PLACR_DFCDA_SHIFT))&MCM_PLACR_DFCDA_MASK)  

#define MCM_PLACR_DFCIC_MASK 0x1000u  

#define MCM_PLACR_DFCIC_SHIFT 12  

#define MCM_PLACR_DFCIC_WIDTH 1  

#define MCM_PLACR_DFCIC(x) (((uint32_t)((uint32_t)(x))<<MCM_PLACR_DFCIC_SHIFT))&MCM_PLACR_DFCIC_MASK)  

#define MCM_PLACR_DFCC_MASK 0x2000u  

#define MCM_PLACR_DFCC_SHIFT 13  

#define MCM_PLACR_DFCC_WIDTH 1  

#define MCM_PLACR_DFCC(x) (((uint32_t)((uint32_t)(x))<<MCM_PLACR_DFCC_SHIFT))&MCM_PLACR_DFCC_MASK)  

#define MCM_PLACR_EFDS_MASK 0x4000u  

#define MCM_PLACR_EFDS_SHIFT 14  

#define MCM_PLACR_EFDS_WIDTH 1

```

```

#define MCM_PLACR_EFDS(x)
(((uint32_t)((uint32_t)(x))<<MCM_PLACR_EFDS_SHIFT))&MCM_PLACR_EFDS_MASK)
#define MCM_PLACR_DFCS_MASK 0x8000u
#define MCM_PLACR_DFCS_SHIFT 15
#define MCM_PLACR_DFCS_WIDTH 1
#define MCM_PLACR_DFCS(x)
(((uint32_t)((uint32_t)(x))<<MCM_PLACR_DFCS_SHIFT))&MCM_PLACR_DFCS_MASK)
#define MCM_PLACR_ESFC_MASK 0x10000u
#define MCM_PLACR_ESFC_SHIFT 16
#define MCM_PLACR_ESFC_WIDTH 1
#define MCM_PLACR_ESFC(x)
(((uint32_t)((uint32_t)(x))<<MCM_PLACR_ESFC_SHIFT))&MCM_PLACR_ESFC_MASK)
/* CPO Bit Fields */
#define MCM_CPO_CPOREQ_MASK 0x1u
#define MCM_CPO_CPOREQ_SHIFT 0
#define MCM_CPO_CPOREQ_WIDTH 1
#define MCM_CPO_CPOREQ(x)
(((uint32_t)((uint32_t)(x))<<MCM_CPO_CPOREQ_SHIFT))&MCM_CPO_CPOREQ_MASK)
#define MCM_CPO_CPOACK_MASK 0x2u
#define MCM_CPO_CPOACK_SHIFT 1
#define MCM_CPO_CPOACK_WIDTH 1
#define MCM_CPO_CPOACK(x)
(((uint32_t)((uint32_t)(x))<<MCM_CPO_CPOACK_SHIFT))&MCM_CPO_CPOACK_MASK)
#define MCM_CPO_CPOWOI_MASK 0x4u
#define MCM_CPO_CPOWOI_SHIFT 2
#define MCM_CPO_CPOWOI_WIDTH 1
#define MCM_CPO_CPOWOI(x)
(((uint32_t)((uint32_t)(x))<<MCM_CPO_CPOWOI_SHIFT))&MCM_CPO_CPOWOI_MASK)

/* !
 * @}
 */ /* end of group MCM_Register_Masks */


```

```

/* MCM - Peripheral instance base addresses */
/** Peripheral MCM base address */
#define MCM_BASE (0xF0003000u)
/** Peripheral MCM base pointer */
#define MCM ((MCM_Type *)MCM_BASE)
#define MCM_BASE_PTR (MCM)
/** Array initializer of MCM peripheral base addresses */
#define MCM_BASE_ADDRS { MCM_BASE }
/** Array initializer of MCM peripheral base pointers */
#define MCM_BASE_PTRS { MCM }

/* -----
-- MCM - Register accessor macros
-----
-- */

/* !
 * @addtogroup MCM_Register_Accessor_Macros MCM - Register accessor
macros

```

```

* @{
*/
/* MCM - Register instance definitions */
/* MCM */
#define MCM_PLASC MCM_PLASC_REG(MCM)
#define MCM_PLAMC MCM_PLAMC_REG(MCM)
#define MCM_PLACR MCM_PLACR_REG(MCM)
#define MCM_CPO MCM_CPO_REG(MCM)

/*!!
* @}
*/
/* end of group MCM_Register_Accessor_Macros */

/*!!
* @}
*/
/* end of group MCM_Peripheral_Access_Layer */

/*
-----
-- MTB Peripheral Access Layer
-----
*/
/*!!
* @addtogroup MTB_Peripheral_Access_Layer MTB Peripheral Access Layer
* @{
*/
/** MTB - Register Layout Typedef */
typedef struct {
    __IO uint32_t POSITION;           /**< MTB Position
Register, offset: 0x0 */
    __IO uint32_t MASTER;            /**< MTB Master
Register, offset: 0x4 */
    __IO uint32_t FLOW;              /**< MTB Flow
Register, offset: 0x8 */
    __I  uint32_t BASE;             /**< MTB Base
Register, offset: 0xC */
    uint8_t RESERVED_0[3824];
    __I  uint32_t MODECTRL;          /**< Integration Mode
Control Register, offset: 0xF00 */
    uint8_t RESERVED_1[156];
    __I  uint32_t TAGSET;            /**< Claim TAG Set
Register, offset: 0xFA0 */
    __I  uint32_t TAGCLEAR;          /**< Claim TAG Clear
Register, offset: 0xFA4 */
    uint8_t RESERVED_2[8];
    __I  uint32_t LOCKACCESS;        /**< Lock Access
Register, offset: 0xFB0 */
}

```

```

    __I uint32_t LOCKSTAT;                                /**< Lock Status
Register, offset: 0xFB4 */
    __I uint32_t AUTHSTAT;                               /**< Authentication
Status Register, offset: 0xFB8 */
    __I uint32_t DEVICESEARCH;                           /**< Device
Architecture Register, offset: 0xFBC */
        uint8_t RESERVED_3[8];
    __I uint32_t DEVICECFG;                             /**< Device
Configuration Register, offset: 0xFC8 */
    __I uint32_t DEVICETYPID;                           /**< Device Type
Identifier Register, offset: 0xFCC */
    __I uint32_t PERIPHID[8];                           /**< Peripheral ID
Register, array offset: 0xFD0, array step: 0x4 */
    __I uint32_t COMPID[4];                            /**< Component ID
Register, array offset: 0xFF0, array step: 0x4 */
} MTB_Type, *MTB_MemMapPtr;

/* -----
-- MTB - Register accessor macros
-----
*/
/*!
* @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor
macros
* @{
*/
/* MTB - Register accessors */
#define MTB_POSITION_REG(base)                         ((base) -> POSITION)
#define MTB_MASTER_REG(base)                          ((base) -> MASTER)
#define MTB_FLOW_REG(base)                           ((base) -> FLOW)
#define MTB_BASE_REG(base)                           ((base) -> BASE)
#define MTB_MODECTRL_REG(base)                        ((base) -> MODECTRL)
#define MTB_TAGSET_REG(base)                          ((base) -> TAGSET)
#define MTB_TAGCLEAR_REG(base)                        ((base) -> TAGCLEAR)
#define MTB_LOCKACCESS_REG(base)                      ((base) -> LOCKACCESS)
#define MTB_LOCKSTAT_REG(base)                        ((base) -> LOCKSTAT)
#define MTB_AUTHSTAT_REG(base)                        ((base) -> AUTHSTAT)
#define MTB_DEVICESEARCH_REG(base)                   ((base) -> DEVICESEARCH)
#define MTB_DEVICECFG_REG(base)                       ((base) -> DEVICECFG)
#define MTB_DEVICETYPID_REG(base)                     ((base) -> DEVICETYPID)
#define MTB_PERIPHID_REG(base, index)                ((base) -> PERIPHID[index])
#define MTB_PERIPHID_COUNT                         8
#define MTB_COMPID_REG(base, index)                  ((base) -> COMPID[index])
#define MTB_COMPID_COUNT                           4

/*
* @{
*/
/* end of group MTB_Register_Accessor_Macros */

```

```

/* -----
-- MTB Register Masks
-----
*/
/* !
* @addtogroup MTB_Register_Masks MTB Register Masks
* @{
*/
/* POSITION Bit Fields */
#define MTB_POSITION_WRAP_MASK           0x4u
#define MTB_POSITION_WRAP_SHIFT          2
#define MTB_POSITION_WRAP_WIDTH          1
#define MTB_POSITION_WRAP(x)             (((uint32_t)((uint32_t)(x))<<MTB_POSITION_WRAP_SHIFT))&MTB_POSITION_WRAP_MASK
#define MTB_POSITION_POINTER_MASK         0xFFFFFFFF8u
#define MTB_POSITION_POINTER_SHIFT       3
#define MTB_POSITION_POINTER_WIDTH       29
#define MTB_POSITION_POINTER(x)          (((uint32_t)((uint32_t)(x))<<MTB_POSITION_POINTER_SHIFT))&MTB_POSITION_POINTER_MASK
/* MASTER Bit Fields */
#define MTB_MASTER_MASK_MASK            0x1Fu
#define MTB_MASTER_MASK_SHIFT           0
#define MTB_MASTER_MASK_WIDTH           5
#define MTB_MASTER_MASK(x)              (((uint32_t)((uint32_t)(x))<<MTB_MASTER_MASK_SHIFT))&MTB_MASTER_MASK_MASK
#define MTB_MASTER_TSTARTEN_MASK         0x20u
#define MTB_MASTER_TSTARTEN_SHIFT        5
#define MTB_MASTER_TSTARTEN_WIDTH       1
#define MTB_MASTER_TSTARTEN(x)           (((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTARTEN_SHIFT))&MTB_MASTER_TSTARTEN_MASK
#define MTB_MASTER_TSTOPEN_MASK          0x40u
#define MTB_MASTER_TSTOPEN_SHIFT         6
#define MTB_MASTER_TSTOPEN_WIDTH        1
#define MTB_MASTER_TSTOPEN(x)           (((uint32_t)((uint32_t)(x))<<MTB_MASTER_TSTOPEN_SHIFT))&MTB_MASTER_TSTOPEN_MASK
#define MTB_MASTER_SFRWPRIV_MASK         0x80u
#define MTB_MASTER_SFRWPRIV_SHIFT        7
#define MTB_MASTER_SFRWPRIV_WIDTH       1
#define MTB_MASTER_SFRWPRIV(x)          (((uint32_t)((uint32_t)(x))<<MTB_MASTER_SFRWPRIV_SHIFT))&MTB_MASTER_SFRWPRIV_MASK
#define MTB_MASTER_RAMPRIV_MASK          0x100u
#define MTB_MASTER_RAMPRIV_SHIFT         8
#define MTB_MASTER_RAMPRIV_WIDTH        1

```

```

#define MTB_MASTER_RAMPRIV(x)
(((uint32_t) (((uint32_t) (x)) << MTB_MASTER_RAMPRIV_SHIFT)) & MTB_MASTER_RAMPR
IV_MASK)

#define MTB_MASTER_HALTREQ_MASK 0x200u
#define MTB_MASTER_HALTREQ_SHIFT 9
#define MTB_MASTER_HALTREQ_WIDTH 1
#define MTB_MASTER_HALTREQ(x)
(((uint32_t) (((uint32_t) (x)) << MTB_MASTER_HALTREQ_SHIFT)) & MTB_MASTER_HALTREQ_
MASK)

#define MTB_MASTER_EN_MASK 0x80000000u
#define MTB_MASTER_EN_SHIFT 31
#define MTB_MASTER_EN_WIDTH 1
#define MTB_MASTER_EN(x)
(((uint32_t) (((uint32_t) (x)) << MTB_MASTER_EN_SHIFT)) & MTB_MASTER_EN_MASK)

/* FLOW Bit Fields */
#define MTB_FLOW_AUTOSTOP_MASK 0x1u
#define MTB_FLOW_AUTOSTOP_SHIFT 0
#define MTB_FLOW_AUTOSTOP_WIDTH 1
#define MTB_FLOW_AUTOSTOP(x)
(((uint32_t) (((uint32_t) (x)) << MTB_FLOW_AUTOSTOP_SHIFT)) & MTB_FLOW_AUTOSTOP_
MASK)

#define MTB_FLOW_AUTOHALT_MASK 0x2u
#define MTB_FLOW_AUTOHALT_SHIFT 1
#define MTB_FLOW_AUTOHALT_WIDTH 1
#define MTB_FLOW_AUTOHALT(x)
(((uint32_t) (((uint32_t) (x)) << MTB_FLOW_AUTOHALT_SHIFT)) & MTB_FLOW_AUTOHALT_
MASK)

#define MTB_FLOW_WATERMARK_MASK 0xFFFFFFFF8u
#define MTB_FLOW_WATERMARK_SHIFT 3
#define MTB_FLOW_WATERMARK_WIDTH 29
#define MTB_FLOW_WATERMARK(x)
(((uint32_t) (((uint32_t) (x)) << MTB_FLOW_WATERMARK_SHIFT)) & MTB_FLOW_WATERMA
RK_MASK)

/* BASE Bit Fields */
#define MTB_BASE_BASEADDR_MASK 0xFFFFFFFFu
#define MTB_BASE_BASEADDR_SHIFT 0
#define MTB_BASE_BASEADDR_WIDTH 32
#define MTB_BASE_BASEADDR(x)
(((uint32_t) (((uint32_t) (x)) << MTB_BASE_BASEADDR_SHIFT)) & MTB_BASE_BASEADDR_
MASK)

/* MODECTRL Bit Fields */
#define MTB_MODECTRL_MODECTRL_MASK 0xFFFFFFFFu
#define MTB_MODECTRL_MODECTRL_SHIFT 0
#define MTB_MODECTRL_MODECTRL_WIDTH 32
#define MTB_MODECTRL_MODECTRL(x)
(((uint32_t) (((uint32_t) (x)) << MTB_MODECTRL_MODECTRL_SHIFT)) & MTB_MODECTRL_-
MODECTRL_MASK)

/* TAGSET Bit Fields */
#define MTB_TAGSET_TAGSET_MASK 0xFFFFFFFFu
#define MTB_TAGSET_TAGSET_SHIFT 0
#define MTB_TAGSET_TAGSET_WIDTH 32
#define MTB_TAGSET_TAGSET(x)
(((uint32_t) (((uint32_t) (x)) << MTB_TAGSET_TAGSET_SHIFT)) & MTB_TAGSET_TAGSET_
MASK)

```

```

/* TAGCLEAR Bit Fields */
#define MTB_TAGCLEAR_TAGCLEAR_MASK          0xFFFFFFFFu
#define MTB_TAGCLEAR_TAGCLEAR_SHIFT         0
#define MTB_TAGCLEAR_TAGCLEAR_WIDTH         32
#define MTB_TAGCLEAR_TAGCLEAR(x)
(((uint32_t) (((uint32_t) (x)) << MTB_TAGCLEAR_TAGCLEAR_SHIFT)) & MTB_TAGCLEAR_TAGCLEAR_MASK)

/* LOCKACCESS Bit Fields */
#define MTB_LOCKACCESS_LOCKACCESS_MASK       0xFFFFFFFFu
#define MTB_LOCKACCESS_LOCKACCESS_SHIFT      0
#define MTB_LOCKACCESS_LOCKACCESS_WIDTH      32
#define MTB_LOCKACCESS_LOCKACCESS(x)
(((uint32_t) (((uint32_t) (x)) << MTB_LOCKACCESS_LOCKACCESS_SHIFT)) & MTB_LOCKACCESS_LOCKACCESS_MASK)

/* LOCKSTAT Bit Fields */
#define MTB_LOCKSTAT_LOCKSTAT_MASK          0xFFFFFFFFu
#define MTB_LOCKSTAT_LOCKSTAT_SHIFT         0
#define MTB_LOCKSTAT_LOCKSTAT_WIDTH         32
#define MTB_LOCKSTAT_LOCKSTAT(x)
(((uint32_t) (((uint32_t) (x)) << MTB_LOCKSTAT_LOCKSTAT_SHIFT)) & MTB_LOCKSTAT_LOCKSTAT_MASK)

/* AUTHSTAT Bit Fields */
#define MTB_AUTHSTAT_BIT0_MASK              0x1u
#define MTB_AUTHSTAT_BIT0_SHIFT             0
#define MTB_AUTHSTAT_BIT0_WIDTH             1
#define MTB_AUTHSTAT_BIT0(x)
(((uint32_t) (((uint32_t) (x)) << MTB_AUTHSTAT_BIT0_SHIFT)) & MTB_AUTHSTAT_BIT0_MASK)

#define MTB_AUTHSTAT_BIT1_MASK              0x2u
#define MTB_AUTHSTAT_BIT1_SHIFT             1
#define MTB_AUTHSTAT_BIT1_WIDTH             1
#define MTB_AUTHSTAT_BIT1(x)
(((uint32_t) (((uint32_t) (x)) << MTB_AUTHSTAT_BIT1_SHIFT)) & MTB_AUTHSTAT_BIT1_MASK)

#define MTB_AUTHSTAT_BIT2_MASK              0x4u
#define MTB_AUTHSTAT_BIT2_SHIFT             2
#define MTB_AUTHSTAT_BIT2_WIDTH             1
#define MTB_AUTHSTAT_BIT2(x)
(((uint32_t) (((uint32_t) (x)) << MTB_AUTHSTAT_BIT2_SHIFT)) & MTB_AUTHSTAT_BIT2_MASK)

#define MTB_AUTHSTAT_BIT3_MASK              0x8u
#define MTB_AUTHSTAT_BIT3_SHIFT             3
#define MTB_AUTHSTAT_BIT3_WIDTH             1
#define MTB_AUTHSTAT_BIT3(x)
(((uint32_t) (((uint32_t) (x)) << MTB_AUTHSTAT_BIT3_SHIFT)) & MTB_AUTHSTAT_BIT3_MASK)

/* DEVICESEARCH Bit Fields */
#define MTB_DEVICESEARCH_DEVICESEARCH_MASK   0xFFFFFFFFu
#define MTB_DEVICESEARCH_DEVICESEARCH_SHIFT  0
#define MTB_DEVICESEARCH_DEVICESEARCH_WIDTH  32
#define MTB_DEVICESEARCH_DEVICESEARCH(x)
(((uint32_t) (((uint32_t) (x)) << MTB_DEVICESEARCH_DEVICESEARCH_SHIFT)) & MTB_DEVICESEARCH_DEVICESEARCH_MASK)

/* DEVICECFG Bit Fields */

```

```

#define MTB_DEVICECFG_DEVICECFG_MASK           0xFFFFFFFFu
#define MTB_DEVICECFG_DEVICECFG_SHIFT          0
#define MTB_DEVICECFG_DEVICECFG_WIDTH          32
#define MTB_DEVICECFG_DEVICECFG(x)
(((uint32_t) (((uint32_t)(x)) << MTB_DEVICECFG_DEVICECFG_SHIFT)) & MTB_DEVICECFG_DEVICECFG_MASK)
/* DEVICETYPID Bit Fields */
#define MTB_DEVICETYPID_DEVICETYPID_MASK        0xFFFFFFFFu
#define MTB_DEVICETYPID_DEVICETYPID_SHIFT       0
#define MTB_DEVICETYPID_DEVICETYPID_WIDTH        32
#define MTB_DEVICETYPID_DEVICETYPID(x)
(((uint32_t) (((uint32_t)(x)) << MTB_DEVICETYPID_DEVICETYPID_SHIFT)) & MTB_DEVICETYPID_DEVICETYPID_MASK)
/* PERIPHID Bit Fields */
#define MTB_PERIPHID_PERIPHID_MASK              0xFFFFFFFFu
#define MTB_PERIPHID_PERIPHID_SHIFT             0
#define MTB_PERIPHID_PERIPHID_WIDTH             32
#define MTB_PERIPHID_PERIPHID(x)
(((uint32_t) (((uint32_t)(x)) << MTB_PERIPHID_PERIPHID_SHIFT)) & MTB_PERIPHID_PERIPHID_MASK)
/* COMPID Bit Fields */
#define MTB_COMPID_COMPID_MASK                  0xFFFFFFFFu
#define MTB_COMPID_COMPID_SHIFT                 0
#define MTB_COMPID_COMPID_WIDTH                 32
#define MTB_COMPID_COMPID(x)
(((uint32_t) (((uint32_t)(x)) << MTB_COMPID_COMPID_SHIFT)) & MTB_COMPID_COMPID_MASK)

/* !
 * @}
 */ /* end of group MTB_Register_Masks */

```

```

/* MTB - Peripheral instance base addresses */
/** Peripheral MTB base address */
#define MTB_BASE                                (0xF0000000u)
/** Peripheral MTB base pointer */
#define MTB                                     ((MTB_Type *) MTB_BASE)
#define MTB_BASE_PTR                           (MTB)
/** Array initializer of MTB peripheral base addresses */
#define MTB_BASE_ADDRS                         { MTB_BASE }
/** Array initializer of MTB peripheral base pointers */
#define MTB_BASE_PTRS                          { MTB }

```

```

/* -----
-- MTB - Register accessor macros
-----
*/

```

```

/* !
 * @addtogroup MTB_Register_Accessor_Macros MTB - Register accessor
macros
 * @{

```

```

*/
```

```

/* MTB - Register instance definitions */
/* MTB */

#define MTB_POSITION           MTB_POSITION_REG(MTB)
#define MTB_MASTER              MTB_MASTER_REG(MTB)
#define MTB_FLOW                MTB_FLOW_REG(MTB)
#define MTB_BASER               MTB_BASE_REG(MTB)
#define MTB_MODECTRL             MTB_MODECTRL_REG(MTB)
#define MTB_TAGSET               MTB_TAGSET_REG(MTB)
#define MTB_TAGCLEAR              MTB_TAGCLEAR_REG(MTB)
#define MTB_LOCKACCESS             MTB_LOCKACCESS_REG(MTB)
#define MTB_LOCKSTAT              MTB_LOCKSTAT_REG(MTB)
#define MTB_AUTHSTAT              MTB_AUTHSTAT_REG(MTB)
#define MTB_DEVICESEARCH            MTB_DEVICESEARCH_REG(MTB)
#define MTB_DEVICECFG              MTB_DEVICECFG_REG(MTB)
#define MTB_DEVICETYPID             MTB_DEVICETYPID_REG(MTB)
#define MTB_PERIPHID4              MTB_PERIPHID_REG(MTB, 0)
#define MTB_PERIPHID5              MTB_PERIPHID_REG(MTB, 1)
#define MTB_PERIPHID6              MTB_PERIPHID_REG(MTB, 2)
#define MTB_PERIPHID7              MTB_PERIPHID_REG(MTB, 3)
#define MTB_PERIPHID0              MTB_PERIPHID_REG(MTB, 4)
#define MTB_PERIPHID1              MTB_PERIPHID_REG(MTB, 5)
#define MTB_PERIPHID2              MTB_PERIPHID_REG(MTB, 6)
#define MTB_PERIPHID3              MTB_PERIPHID_REG(MTB, 7)
#define MTB_COMPID0                MTB_COMPID_REG(MTB, 0)
#define MTB_COMPID1                MTB_COMPID_REG(MTB, 1)
#define MTB_COMPID2                MTB_COMPID_REG(MTB, 2)
#define MTB_COMPID3                MTB_COMPID_REG(MTB, 3)

/* MTB - Register array accessors */
#define MTB_PERIPHID(index)        MTB_PERIPHID_REG(MTB, index)
#define MTB_COMPID(index)          MTB_COMPID_REG(MTB, index)

/* !
 * @}
 */ /* end of group MTB_Register_Accessor_Macros */

/* !
 * @}
 */ /* end of group MTB_Peripheral_Access_Layer */

/*
-----  

-- MTBDWT Peripheral Access Layer  

-----  

----- */
/* !

```

```

 * @addtogroup MTBDWT_Peripheral_Access_Layer MTBDWT Peripheral Access
Layer
 * @{
 */

/** MTBDWT - Register Layout Typedef */
typedef struct {
    __I uint32_t CTRL;                                /**< MTB DWT Control
Register, offset: 0x0 */
    uint8_t RESERVED_0[28];
    struct {                                         /* offset: 0x20, array
step: 0x10 */
        __IO uint32_t COMP;                            /**< MTB_DWT
Comparator Register, array offset: 0x20, array step: 0x10 */
        __IO uint32_t MASK;                           /**< MTB_DWT
Comparator Mask Register, array offset: 0x24, array step: 0x10 */
        __IO uint32_t FCT;                            /**< MTB_DWT
Comparator Function Register 0..MTB_DWT Comparator Function Register 1,
array offset: 0x28, array step: 0x10 */
        uint8_t RESERVED_0[4];
    } COMPARATOR[2];
    uint8_t RESERVED_1[448];
    __IO uint32_t TBCTRL;                            /**< MTB_DWT Trace
Buffer Control Register, offset: 0x200 */
    uint8_t RESERVED_2[3524];
    __I uint32_t DEVICECFG;                         /**< Device
Configuration Register, offset: 0xFC8 */
    __I uint32_t DEVICETYPID;                      /**< Device Type
Identifier Register, offset: 0xFCC */
    __I uint32_t PERIPHID[8];                      /**< Peripheral ID
Register, array offset: 0xFD0, array step: 0x4 */
    __I uint32_t COMPID[4];                         /**< Component ID
Register, array offset: 0xFF0, array step: 0x4 */
} MTBDWT_Type, *MTBDWT_MemMapPtr;

/*
-----
-- MTBDWT - Register accessor macros
-----
*/
/*!
 * @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor
macros
 * @{
 */

/* MTBDWT - Register accessors */
#define MTBDWT_CTRL_REG(base)          ((base)->CTRL)
#define MTBDWT_COMP_REG(base, index)   ((base)-
>COMPARATOR[index].COMP)
#define MTBDWT_COMP_COUNT             2

```

```

#define MTBDWT_MASK_REG(base, index)          ((base) -  

>COMPARATOR[index].MASK)  

#define MTBDWT_MASK_COUNT                   2  

#define MTBDWT_FCT_REG(base, index)          ((base) -  

>COMPARATOR[index].FCT)  

#define MTBDWT_FCT_COUNT                   2  

#define MTBDWT_TBCTRL_REG(base)             ((base) ->TBCTRL)  

#define MTBDWT_DEVICECFG_REG(base)          ((base) ->DEVICECFG)  

#define MTBDWT_DEVICETYPID_REG(base)         ((base) ->DEVICETYPID)  

#define MTBDWT_PERIPHID_REG(base, index)     ((base) -  

>PERIPHID[index])  

#define MTBDWT_PERIPHID_COUNT              8  

#define MTBDWT_COMPID_REG(base, index)        ((base) ->COMPID[index])  

#define MTBDWT_COMPID_COUNT                4

/*!  
 * @}  
 */ /* end of group MTBDWT_Register_Accessor_Macros */

/* -----
-- MTBDWT Register Masks
-----
*/
/*!  
 * @addtogroup MTBDWT_Register_Masks MTBDWT Register Masks
 * @{
 */

/* CTRL Bit Fields */
#define MTBDWT_CTRL_DWTCFGCTRL_MASK          0xFFFFFFFFu
#define MTBDWT_CTRL_DWTCFGCTRL_SHIFT         0
#define MTBDWT_CTRL_DWTCFGCTRL_WIDTH         28
#define MTBDWT_CTRL_DWTCFGCTRL(x)            (((uint32_t)((uint32_t)(x)) << MTBDWT_CTRL_DWTCFGCTRL_SHIFT) & MTBDWT_CTRL_DWTCFGCTRL_MASK)
#define MTBDWT_CTRL_NUMCMP_MASK              0xF0000000u
#define MTBDWT_CTRL_NUMCMP_SHIFT             28
#define MTBDWT_CTRL_NUMCMP_WIDTH             4
#define MTBDWT_CTRL_NUMCMP(x)               (((uint32_t)((uint32_t)(x)) << MTBDWT_CTRL_NUMCMP_SHIFT) & MTBDWT_CTRL_NUMCMP_MASK)
/* COMP Bit Fields */
#define MTBDWT_COMP_COMP_MASK                0xFFFFFFFFu
#define MTBDWT_COMP_COMP_SHIFT               0
#define MTBDWT_COMP_COMP_WIDTH              32
#define MTBDWT_COMP_COMP(x)                 (((uint32_t)((uint32_t)(x)) << MTBDWT_COMP_COMP_SHIFT) & MTBDWT_COMP_COMP_MASK)
/* MASK Bit Fields */
#define MTBDWT_MASK_MASK_MASK                0x1Fu
#define MTBDWT_MASK_MASK_SHIFT               0

```

```

#define MTBDWT_MASK_WIDTH 5
#define MTBDWT_MASK(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_MASK_SHIFT))&MTBDWT_MASK_MASK
/* FCT Bit Fields */
#define MTBDWT_FCT_FUNCTION_MASK 0xFu
#define MTBDWT_FCT_FUNCTION_SHIFT 0
#define MTBDWT_FCT_FUNCTION_WIDTH 4
#define MTBDWT_FCT_FUNCTION(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_FUNCTION_SHIFT))&MTBDWT_FCT_FUNCTION_MASK
#define MTBDWT_FCT_DATAVMATCH_MASK 0x100u
#define MTBDWT_FCT_DATAVMATCH_SHIFT 8
#define MTBDWT_FCT_DATAVMATCH_WIDTH 1
#define MTBDWT_FCT_DATAVMATCH(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVMATCH_SHIFT))&MTBDWT_FCT_DATAVMATCH_MASK
#define MTBDWT_FCT_DATAVSIZE_MASK 0xC00u
#define MTBDWT_FCT_DATAVSIZE_SHIFT 10
#define MTBDWT_FCT_DATAVSIZE_WIDTH 2
#define MTBDWT_FCT_DATAVSIZE(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVSIZE_SHIFT))&MTBDWT_FCT_DATAVSIZE_MASK
#define MTBDWT_FCT_DATAVADDR0_MASK 0xF000u
#define MTBDWT_FCT_DATAVADDR0_SHIFT 12
#define MTBDWT_FCT_DATAVADDR0_WIDTH 4
#define MTBDWT_FCT_DATAVADDR0(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_DATAVADDR0_SHIFT))&MTBDWT_FCT_DATAVADDR0_MASK
#define MTBDWT_FCT_MATCHED_MASK 0x1000000u
#define MTBDWT_FCT_MATCHED_SHIFT 24
#define MTBDWT_FCT_MATCHED_WIDTH 1
#define MTBDWT_FCT_MATCHED(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_FCT_MATCHED_SHIFT))&MTBDWT_FCT_MATCHED_MASK
/* TBCTRL Bit Fields */
#define MTBDWT_TBCTRL_ACMP0_MASK 0x1u
#define MTBDWT_TBCTRL_ACMP0_SHIFT 0
#define MTBDWT_TBCTRL_ACMP0_WIDTH 1
#define MTBDWT_TBCTRL_ACMP0(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACMP0_SHIFT))&MTBDWT_TBCTRL_ACMP0_MASK
#define MTBDWT_TBCTRL_ACMP1_MASK 0x2u
#define MTBDWT_TBCTRL_ACMP1_SHIFT 1
#define MTBDWT_TBCTRL_ACMP1_WIDTH 1
#define MTBDWT_TBCTRL_ACMP1(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_ACMP1_SHIFT))&MTBDWT_TBCTRL_ACMP1_MASK
#define MTBDWT_TBCTRL_NUMCOMP_MASK 0xF000000u
#define MTBDWT_TBCTRL_NUMCOMP_SHIFT 28
#define MTBDWT_TBCTRL_NUMCOMP_WIDTH 4
#define MTBDWT_TBCTRL_NUMCOMP(x) (((uint32_t)((uint32_t)(x))<<MTBDWT_TBCTRL_NUMCOMP_SHIFT))&MTBDWT_TBCTRL_NUMCOMP_MASK

```

```

/* DEVICECFG Bit Fields */
#define MTBDWT_DEVICECFG_DEVICECFG_MASK          0xFFFFFFFFu
#define MTBDWT_DEVICECFG_DEVICECFG_SHIFT         0
#define MTBDWT_DEVICECFG_DEVICECFG_WIDTH          32
#define MTBDWT_DEVICECFG_DEVICECFG(x)             (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICECFG_DEVICECFG_SHIFT))&MTBDWT_DEVICECFG_DEVICECFG_MASK
/* DEVICETYPID Bit Fields */
#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK        0xFFFFFFFFu
#define MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT       0
#define MTBDWT_DEVICETYPID_DEVICETYPID_WIDTH        32
#define MTBDWT_DEVICETYPID_DEVICETYPID(x)           (((uint32_t)((uint32_t)(x))<<MTBDWT_DEVICETYPID_DEVICETYPID_SHIFT))&MTBDWT_DEVICETYPID_DEVICETYPID_MASK
/* PERIPHID Bit Fields */
#define MTBDWT_PERIPHID_PERIPHID_MASK            0xFFFFFFFFu
#define MTBDWT_PERIPHID_PERIPHID_SHIFT           0
#define MTBDWT_PERIPHID_PERIPHID_WIDTH          32
#define MTBDWT_PERIPHID_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<MTBDWT_PERIPHID_PERIPHID_SHIFT))&MTBDWT_PERIPHID_PERIPHID_MASK
/* COMPID Bit Fields */
#define MTBDWT_COMPID_COMPID_MASK                 0xFFFFFFFFu
#define MTBDWT_COMPID_COMPID_SHIFT                0
#define MTBDWT_COMPID_COMPID_WIDTH               32
#define MTBDWT_COMPID_COMPID(x)                  (((uint32_t)((uint32_t)(x))<<MTBDWT_COMPID_COMPID_SHIFT))&MTBDWT_COMPID_COMPID_MASK

/* !
 * @}
 */
/* /* end of group MTBDWT_Register_Masks */

/* MTBDWT - Peripheral instance base addresses */
/** Peripheral MTBDWT base address */
#define MTBDWT_BASE                                (0xF0001000u)
/** Peripheral MTBDWT base pointer */
#define MTBDWT                                     ((MTBDWT_Type
*)MTBDWT_BASE)
#define MTBDWT_BASE_PTR                           (MTBDWT)
/** Array initializer of MTBDWT peripheral base addresses */
#define MTBDWT_BASE_ADDRS                         { MTBDWT_BASE }
/** Array initializer of MTBDWT peripheral base pointers */
#define MTBDWT_BASE_PTRS                          { MTBDWT }

/* -----
-- MTBDWT - Register accessor macros
-----
*/
/* !

```

```

 * @addtogroup MTBDWT_Register_Accessor_Macros MTBDWT - Register accessor
macros
 * @{
 *

/* MTBDWT - Register instance definitions */
/* MTBDWT */
#define MTBDWT_CTRL MTBDWT_CTRL_REG(MTBDWT)
#define MTBDWT_COMP0
MTBDWT_COMP_REG(MTBDWT, 0)
#define MTBDWT_MASK0
MTBDWT_MASK_REG(MTBDWT, 0)
#define MTBDWT_FCT0 MTBDWT_FCT_REG(MTBDWT, 0)
#define MTBDWT_COMP1
MTBDWT_COMP_REG(MTBDWT, 1)
#define MTBDWT_MASK1
MTBDWT_MASK_REG(MTBDWT, 1)
#define MTBDWT_FCT1 MTBDWT_FCT_REG(MTBDWT, 1)
#define MTBDWT_TBCTRL
MTBDWT_TBCTRL_REG(MTBDWT)
#define MTBDWT_DEVICECFG
MTBDWT_DEVICECFG_REG(MTBDWT)
#define MTBDWT_DEVICETYPID
MTBDWT_DEVICETYPID_REG(MTBDWT)
#define MTBDWT_PERIPHID4
MTBDWT_PERIPHID_REG(MTBDWT, 0)
#define MTBDWT_PERIPHID5
MTBDWT_PERIPHID_REG(MTBDWT, 1)
#define MTBDWT_PERIPHID6
MTBDWT_PERIPHID_REG(MTBDWT, 2)
#define MTBDWT_PERIPHID7
MTBDWT_PERIPHID_REG(MTBDWT, 3)
#define MTBDWT_PERIPHIDO
MTBDWT_PERIPHID_REG(MTBDWT, 4)
#define MTBDWT_PERIPHID1
MTBDWT_PERIPHID_REG(MTBDWT, 5)
#define MTBDWT_PERIPHID2
MTBDWT_PERIPHID_REG(MTBDWT, 6)
#define MTBDWT_PERIPHID3
MTBDWT_PERIPHID_REG(MTBDWT, 7)
#define MTBDWT_COMPID0
MTBDWT_COMPID_REG(MTBDWT, 0)
#define MTBDWT_COMPID1
MTBDWT_COMPID_REG(MTBDWT, 1)
#define MTBDWT_COMPID2
MTBDWT_COMPID_REG(MTBDWT, 2)
#define MTBDWT_COMPID3
MTBDWT_COMPID_REG(MTBDWT, 3)

/* MTBDWT - Register array accessors */
#define MTBDWT_COMP(index)
MTBDWT_COMP_REG(MTBDWT, index)

```

```

#define MTBDWT__MASK(index)
MTBDWT__MASK_REG(MTBDWT, index)
#define MTBDWT__FCT(index)
MTBDWT__FCT_REG(MTBDWT, index)
#define MTBDWT__PERIPHID(index)
MTBDWT__PERIPHID_REG(MTBDWT, index)
#define MTBDWT__COMPID(index)
MTBDWT__COMPID_REG(MTBDWT, index)

/*!
 * @}
 */ /* end of group MTBDWT_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group MTBDWT_Peripheral_Access_Layer */

/*
-----NV Peripheral Access Layer
-----
*/
/*!
 * @addtogroup NV_Peripheral_Access_Layer NV Peripheral Access Layer
 * @{
 */

/** NV - Register Layout Typedef */
typedef struct {
    __I uint8_t BACKKEY3;                                /**< Backdoor
    Comparison Key 3., offset: 0x0 */
    __I uint8_t BACKKEY2;                                /**< Backdoor
    Comparison Key 2., offset: 0x1 */
    __I uint8_t BACKKEY1;                                /**< Backdoor
    Comparison Key 1., offset: 0x2 */
    __I uint8_t BACKKEY0;                                /**< Backdoor
    Comparison Key 0., offset: 0x3 */
    __I uint8_t BACKKEY7;                                /**< Backdoor
    Comparison Key 7., offset: 0x4 */
    __I uint8_t BACKKEY6;                                /**< Backdoor
    Comparison Key 6., offset: 0x5 */
    __I uint8_t BACKKEY5;                                /**< Backdoor
    Comparison Key 5., offset: 0x6 */
    __I uint8_t BACKKEY4;                                /**< Backdoor
    Comparison Key 4., offset: 0x7 */
    __I uint8_t FPROT3;                                 /**< Non-volatile P-
    Flash Protection 1 - Low Register, offset: 0x8 */
    __I uint8_t FPROT2;                                 /**< Non-volatile P-
    Flash Protection 1 - High Register, offset: 0x9 */
    __I uint8_t FPROT1;                                 /**< Non-volatile P-
    Flash Protection 0 - Low Register, offset: 0xA */
}

```

```

    __I uint8_t FPROT0;                                /**< Non-volatile P-
Flash Protection 0 - High Register, offset: 0xB */
    __I uint8_t FSEC;                                 /**< Non-volatile
Flash Security Register, offset: 0xC */
    __I uint8_t FOPT;                                /**< Non-volatile
Flash Option Register, offset: 0xD */
} NV_Type, *NV_MemMapPtr;

/*
-----
-- NV - Register accessor macros
-----
*/
/*!
* @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros
* @{
*/
/* NV - Register accessors */
#define NV_BACKKEY3_REG(base)          ((base) -> BACKKEY3)
#define NV_BACKKEY2_REG(base)          ((base) -> BACKKEY2)
#define NV_BACKKEY1_REG(base)          ((base) -> BACKKEY1)
#define NV_BACKKEY0_REG(base)          ((base) -> BACKKEY0)
#define NV_BACKKEY7_REG(base)          ((base) -> BACKKEY7)
#define NV_BACKKEY6_REG(base)          ((base) -> BACKKEY6)
#define NV_BACKKEY5_REG(base)          ((base) -> BACKKEY5)
#define NV_BACKKEY4_REG(base)          ((base) -> BACKKEY4)
#define NV_FPROT3_REG(base)           ((base) -> FPROT3)
#define NV_FPROT2_REG(base)           ((base) -> FPROT2)
#define NV_FPROT1_REG(base)           ((base) -> FPROT1)
#define NV_FPROT0_REG(base)           ((base) -> FPROT0)
#define NV_FSEC_REG(base)             ((base) -> FSEC)
#define NV_FOPT_REG(base)             ((base) -> FOPT)

/*!
* @}
*/
/* end of group NV_Register_Accessor_Macros */

/*
-----
-- NV Register Masks
-----
*/
/*!
* @addtogroup NV_Register_Masks NV Register Masks
* @{
*/
/* BACKKEY3 Bit Fields */
#define NV_BACKKEY3_KEY_MASK          0xFFu

```

```

#define NV_BACKKEY3_KEY_SHIFT 0
#define NV_BACKKEY3_KEY_WIDTH 8
#define NV_BACKKEY3_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY3_KEY_SHIFT)) & NV_BACKKEY3_KEY_MASK)
/* BACKKEY2 Bit Fields */
#define NV_BACKKEY2_KEY_MASK 0xFFu
#define NV_BACKKEY2_KEY_SHIFT 0
#define NV_BACKKEY2_KEY_WIDTH 8
#define NV_BACKKEY2_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY2_KEY_SHIFT)) & NV_BACKKEY2_KEY_MASK)
/* BACKKEY1 Bit Fields */
#define NV_BACKKEY1_KEY_MASK 0xFFu
#define NV_BACKKEY1_KEY_SHIFT 0
#define NV_BACKKEY1_KEY_WIDTH 8
#define NV_BACKKEY1_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY1_KEY_SHIFT)) & NV_BACKKEY1_KEY_MASK)
/* BACKKEY0 Bit Fields */
#define NV_BACKKEY0_KEY_MASK 0xFFu
#define NV_BACKKEY0_KEY_SHIFT 0
#define NV_BACKKEY0_KEY_WIDTH 8
#define NV_BACKKEY0_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY0_KEY_SHIFT)) & NV_BACKKEY0_KEY_MASK)
/* BACKKEY7 Bit Fields */
#define NV_BACKKEY7_KEY_MASK 0xFFu
#define NV_BACKKEY7_KEY_SHIFT 0
#define NV_BACKKEY7_KEY_WIDTH 8
#define NV_BACKKEY7_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY7_KEY_SHIFT)) & NV_BACKKEY7_KEY_MASK)
/* BACKKEY6 Bit Fields */
#define NV_BACKKEY6_KEY_MASK 0xFFu
#define NV_BACKKEY6_KEY_SHIFT 0
#define NV_BACKKEY6_KEY_WIDTH 8
#define NV_BACKKEY6_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY6_KEY_SHIFT)) & NV_BACKKEY6_KEY_MASK)
/* BACKKEY5 Bit Fields */
#define NV_BACKKEY5_KEY_MASK 0xFFu
#define NV_BACKKEY5_KEY_SHIFT 0
#define NV_BACKKEY5_KEY_WIDTH 8
#define NV_BACKKEY5_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY5_KEY_SHIFT)) & NV_BACKKEY5_KEY_MASK)
/* BACKKEY4 Bit Fields */
#define NV_BACKKEY4_KEY_MASK 0xFFu
#define NV_BACKKEY4_KEY_SHIFT 0
#define NV_BACKKEY4_KEY_WIDTH 8
#define NV_BACKKEY4_KEY(x) (((uint8_t) (((uint8_t) (x)) << NV_BACKKEY4_KEY_SHIFT)) & NV_BACKKEY4_KEY_MASK)
/* FPROT3 Bit Fields */
#define NV_FPROT3_PROT_MASK 0xFFu
#define NV_FPROT3_PROT_SHIFT 0
#define NV_FPROT3_PROT_WIDTH 8
#define NV_FPROT3_PROT(x) (((uint8_t) (((uint8_t) (x)) << NV_FPROT3_PROT_SHIFT)) & NV_FPROT3_PROT_MASK)
/* FPROT2 Bit Fields */
#define NV_FPROT2_PROT_MASK 0xFFu

```

```

#define NV_FPROT2_PROT_SHIFT 0
#define NV_FPROT2_PROT_WIDTH 8
#define NV_FPROT2_PROT(x) (((uint8_t)((uint8_t)(x)<<NV_FPROT2_PROT_SHIFT))&NV_FPROT2_PROT_MASK)
/* FPROT1 Bit Fields */
#define NV_FPROT1_PROT_MASK 0xFFu
#define NV_FPROT1_PROT_SHIFT 0
#define NV_FPROT1_PROT_WIDTH 8
#define NV_FPROT1_PROT(x) (((uint8_t)((uint8_t)(x)<<NV_FPROT1_PROT_SHIFT))&NV_FPROT1_PROT_MASK)
/* FPROT0 Bit Fields */
#define NV_FPROT0_PROT_MASK 0xFFu
#define NV_FPROT0_PROT_SHIFT 0
#define NV_FPROT0_PROT_WIDTH 8
#define NV_FPROT0_PROT(x) (((uint8_t)((uint8_t)(x)<<NV_FPROT0_PROT_SHIFT))&NV_FPROT0_PROT_MASK)
/* FSEC Bit Fields */
#define NV_FSEC_SEC_MASK 0x3u
#define NV_FSEC_SEC_SHIFT 0
#define NV_FSEC_SEC_WIDTH 2
#define NV_FSEC_SEC(x) (((uint8_t)((uint8_t)(x)<<NV_FSEC_SEC_SHIFT))&NV_FSEC_SEC_MASK)
#define NV_FSEC_FSLACC_MASK 0xCu
#define NV_FSEC_FSLACC_SHIFT 2
#define NV_FSEC_FSLACC_WIDTH 2
#define NV_FSEC_FSLACC(x) (((uint8_t)((uint8_t)(x)<<NV_FSEC_FSLACC_SHIFT))&NV_FSEC_FSLACC_MASK)
#define NV_FSEC_MEEN_MASK 0x30u
#define NV_FSEC_MEEN_SHIFT 4
#define NV_FSEC_MEEN_WIDTH 2
#define NV_FSEC_MEEN(x) (((uint8_t)((uint8_t)(x)<<NV_FSEC_MEEN_SHIFT))&NV_FSEC_MEEN_MASK)
#define NV_FSEC_KEYEN_MASK 0xC0u
#define NV_FSEC_KEYEN_SHIFT 6
#define NV_FSEC_KEYEN_WIDTH 2
#define NV_FSEC_KEYEN(x) (((uint8_t)((uint8_t)(x)<<NV_FSEC_KEYEN_SHIFT))&NV_FSEC_KEYEN_MASK)
/* FOPT Bit Fields */
#define NV_FOPT_LPBOOT0_MASK 0x1u
#define NV_FOPT_LPBOOT0_SHIFT 0
#define NV_FOPT_LPBOOT0_WIDTH 1
#define NV_FOPT_LPBOOT0(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_LPBOOT0_SHIFT))&NV_FOPT_LPBOOT0_MASK)
#define NV_FOPT_NMI_DIS_MASK 0x4u
#define NV_FOPT_NMI_DIS_SHIFT 2
#define NV_FOPT_NMI_DIS_WIDTH 1
#define NV_FOPT_NMI_DIS(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_NMI_DIS_SHIFT))&NV_FOPT_NMI_DIS_MASK)
#define NV_FOPT_RESET_PIN_CFG_MASK 0x8u
#define NV_FOPT_RESET_PIN_CFG_SHIFT 3
#define NV_FOPT_RESET_PIN_CFG_WIDTH 1
#define NV_FOPT_RESET_PIN_CFG(x) (((uint8_t)((uint8_t)(x)<<NV_FOPT_RESET_PIN_CFG_SHIFT))&NV_FOPT_RESET_PIN_CFG_MASK)

```

```

#define NV_FOPT_LPBOOT1_MASK          0x10u
#define NV_FOPT_LPBOOT1_SHIFT         4
#define NV_FOPT_LPBOOT1_WIDTH         1
#define NV_FOPT_LPBOOT1(x)
  (((uint8_t) ((uint8_t)(x)) << NV_FOPT_LPBOOT1_SHIFT) & NV_FOPT_LPBOOT1_MASK)
#define NV_FOPT_FAST_INIT_MASK        0x20u
#define NV_FOPT_FAST_INIT_SHIFT       5
#define NV_FOPT_FAST_INIT_WIDTH       1
#define NV_FOPT_FAST_INIT(x)
  (((uint8_t) ((uint8_t)(x)) << NV_FOPT_FAST_INIT_SHIFT) & NV_FOPT_FAST_INIT_MASK)

/* !
 * @}
 */
/* end of group NV_Register_Masks */

/* NV - Peripheral instance base addresses */
/** Peripheral FTFA_FlashConfig base address */
#define FTFA_FlashConfig_BASE          (0x400u)
/** Peripheral FTFA_FlashConfig base pointer */
#define FTFA_FlashConfig               ((NV_Type
*)FTFA_FlashConfig_BASE)
#define FTFA_FlashConfig_BASE_PTR      (FTFA_FlashConfig)
/** Array initializer of NV peripheral base addresses */
#define NV_BASE_ADDRS                 { FTFA_FlashConfig_BASE
}
/** Array initializer of NV peripheral base pointers */
#define NV_BASE_PTRS                  { FTFA_FlashConfig }

/* -----
-- NV - Register accessor macros
-----
*/
/* !
 * @addtogroup NV_Register_Accessor_Macros NV - Register accessor macros
 * @{
 */
/* NV - Register instance definitions */
/* FTFA_FlashConfig */
#define NV_BACKKEY3
NV_BACKKEY3_REG(FTFA_FlashConfig)
#define NV_BACKKEY2
NV_BACKKEY2_REG(FTFA_FlashConfig)
#define NV_BACKKEY1
NV_BACKKEY1_REG(FTFA_FlashConfig)
#define NV_BACKKEY0
NV_BACKKEY0_REG(FTFA_FlashConfig)
#define NV_BACKKEY7
NV_BACKKEY7_REG(FTFA_FlashConfig)

```

```

#define NV_BACKKEY6
NV_BACKKEY6_REG(FTFA_FlashConfig)
#define NV_BACKKEY5
NV_BACKKEY5_REG(FTFA_FlashConfig)
#define NV_BACKKEY4
NV_BACKKEY4_REG(FTFA_FlashConfig)
#define NV_FPROT3
NV_FPROT3_REG(FTFA_FlashConfig)
#define NV_FPROT2
NV_FPROT2_REG(FTFA_FlashConfig)
#define NV_FPROT1
NV_FPROT1_REG(FTFA_FlashConfig)
#define NV_FPROTO
NV_FPROTO_REG(FTFA_FlashConfig)
#define NV_FSEC
NV_FSEC_REG(FTFA_FlashConfig)
#define NV_FOPT
NV_FOPT_REG(FTFA_FlashConfig)

/*!
 * @}
 */ /* end of group NV_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group NV_Peripheral_Access_Layer */

/* -----
-- OSC Peripheral Access Layer
-----
*/
/*!
 * @addtogroup OSC_Peripheral_Access_Layer OSC Peripheral Access Layer
 * @{
 */
/** OSC - Register Layout Typedef */
typedef struct {
    __IO uint8_t CR;                                /**< OSC Control
Register, offset: 0x0 */
} OSC_Type, *OSC_MemMapPtr;

/* -----
-- OSC - Register accessor macros
-----
*/
/*!

```

```

 * @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor
macros
 * @{
 */

/* OSC - Register accessors */
#define OSC_CR_REG(base) ( (base) ->CR)

/*!
 * @}
 */ /* end of group OSC_Register_Accessor_Macros */

/* -----
-- OSC Register Masks
-----
*/
/*!
 * @addtogroup OSC_Register_Masks OSC Register Masks
 * @{
 */

/* CR Bit Fields */
#define OSC_CR_SC16P_MASK 0x1u
#define OSC_CR_SC16P_SHIFT 0
#define OSC_CR_SC16P_WIDTH 1
#define OSC_CR_SC16P(x) (((uint8_t)((uint8_t)(x)) << OSC_CR_SC16P_SHIFT)) & OSC_CR_SC16P_MASK
#define OSC_CR_SC8P_MASK 0x2u
#define OSC_CR_SC8P_SHIFT 1
#define OSC_CR_SC8P_WIDTH 1
#define OSC_CR_SC8P(x) (((uint8_t)((uint8_t)(x)) << OSC_CR_SC8P_SHIFT)) & OSC_CR_SC8P_MASK
#define OSC_CR_SC4P_MASK 0x4u
#define OSC_CR_SC4P_SHIFT 2
#define OSC_CR_SC4P_WIDTH 1
#define OSC_CR_SC4P(x) (((uint8_t)((uint8_t)(x)) << OSC_CR_SC4P_SHIFT)) & OSC_CR_SC4P_MASK
#define OSC_CR_SC2P_MASK 0x8u
#define OSC_CR_SC2P_SHIFT 3
#define OSC_CR_SC2P_WIDTH 1
#define OSC_CR_SC2P(x) (((uint8_t)((uint8_t)(x)) << OSC_CR_SC2P_SHIFT)) & OSC_CR_SC2P_MASK
#define OSC_CR_EREFSTEN_MASK 0x20u
#define OSC_CR_EREFSTEN_SHIFT 5
#define OSC_CR_EREFSTEN_WIDTH 1
#define OSC_CR_EREFSTEN(x) (((uint8_t)((uint8_t)(x)) << OSC_CR_EREFSTEN_SHIFT)) & OSC_CR_EREFSTEN_MASK
#define OSC_CR_ERCLKEN_MASK 0x80u
#define OSC_CR_ERCLKEN_SHIFT 7
#define OSC_CR_ERCLKEN_WIDTH 1

```

```

#define OSC_CR_ERCLKEN(x)
(((uint8_t)((uint8_t)(x))<<OSC_CR_ERCLKEN_SHIFT))&OSC_CR_ERCLKEN_MASK)

/*!
 * @}
 */
/* end of group OSC_Register_Masks */

/* OSC - Peripheral instance base addresses */
/** Peripheral OSC0 base address */
#define OSC0_BASE (0x40065000u)
/** Peripheral OSC0 base pointer */
#define OSC0 ((OSC_Type *)OSC0_BASE)
#define OSC0_BASE_PTR (OSC0)
/** Array initializer of OSC peripheral base addresses */
#define OSC_BASE_ADDRS { OSC0_BASE }
/** Array initializer of OSC peripheral base pointers */
#define OSC_BASE_PTRS { OSC0 }

/* -----
-- OSC - Register accessor macros
-----
*/
/*!
 * @addtogroup OSC_Register_Accessor_Macros OSC - Register accessor
macros
 * @{
 */
/* OSC - Register instance definitions */
/* OSC0 */
#define OSC0_CR OSC_CR_REG(OSC0)

/*!
 * @}
 */
/* end of group OSC_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group OSC_Peripheral_Access_Layer */

/* -----
-- PIT Peripheral Access Layer
-----
*/
/*!
 * @addtogroup PIT_Peripheral_Access_Layer PIT Peripheral Access Layer
 */

```



```

#define PIT_TFLG_COUNT 2

/* !
 * @}
 */ /* end of group PIT_Register_Accessor_Macros */

/*
-----  

-- PIT Register Masks  

-----  

----- */

/* !
 * @addtogroup PIT_Register_Masks PIT Register Masks
 * @{
 */

/* MCR Bit Fields */
#define PIT_MCR_FRZ_MASK 0x1u
#define PIT_MCR_FRZ_SHIFT 0
#define PIT_MCR_FRZ_WIDTH 1
#define PIT_MCR_FRZ(x) (((uint32_t)((uint32_t)(x))<<PIT_MCR_FRZ_SHIFT))&PIT_MCR_FRZ_MASK
#define PIT_MCR_MDIS_MASK 0x2u
#define PIT_MCR_MDIS_SHIFT 1
#define PIT_MCR_MDIS_WIDTH 1
#define PIT_MCR_MDIS(x) (((uint32_t)((uint32_t)(x))<<PIT_MCR_MDIS_SHIFT))&PIT_MCR_MDIS_MASK

/* LTMR64H Bit Fields */
#define PIT_LTMR64H_LTH_MASK 0xFFFFFFFFu
#define PIT_LTMR64H_LTH_SHIFT 0
#define PIT_LTMR64H_LTH_WIDTH 32
#define PIT_LTMR64H_LTH(x) (((uint32_t)((uint32_t)(x))<<PIT_LTMR64H_LTH_SHIFT))&PIT_LTMR64H_LTH_MASK

/* LTMR64L Bit Fields */
#define PIT_LTMR64L_LTL_MASK 0xFFFFFFFFu
#define PIT_LTMR64L_LTL_SHIFT 0
#define PIT_LTMR64L_LTL_WIDTH 32
#define PIT_LTMR64L_LTL(x) (((uint32_t)((uint32_t)(x))<<PIT_LTMR64L_LTL_SHIFT))&PIT_LTMR64L_LTL_MASK

/* LDVAL Bit Fields */
#define PIT_LDVAL_TSV_MASK 0xFFFFFFFFu
#define PIT_LDVAL_TSV_SHIFT 0
#define PIT_LDVAL_TSV_WIDTH 32
#define PIT_LDVAL_TSV(x) (((uint32_t)((uint32_t)(x))<<PIT_LDVAL_TSV_SHIFT))&PIT_LDVAL_TSV_MASK

/* CVAL Bit Fields */
#define PIT_CVAL_TVL_MASK 0xFFFFFFFFu
#define PIT_CVAL_TVL_SHIFT 0
#define PIT_CVAL_TVL_WIDTH 32

```

```

#define PIT_CVAL_TVL(x)
(((uint32_t)((uint32_t)(x))<<PIT_CVAL_TVL_SHIFT))&PIT_CVAL_TVL_MASK)
/* TCTRL Bit Fields */
#define PIT_TCTRL_TEN_MASK 0x1u
#define PIT_TCTRL_TEN_SHIFT 0
#define PIT_TCTRL_TEN_WIDTH 1
#define PIT_TCTRL_TEN(x)
(((uint32_t)((uint32_t)(x))<<PIT_TCTRL_TEN_SHIFT))&PIT_TCTRL_TEN_MASK)
#define PIT_TCTRL_TIE_MASK 0x2u
#define PIT_TCTRL_TIE_SHIFT 1
#define PIT_TCTRL_TIE_WIDTH 1
#define PIT_TCTRL_TIE(x)
(((uint32_t)((uint32_t)(x))<<PIT_TCTRL_TIE_SHIFT))&PIT_TCTRL_TIE_MASK)
#define PIT_TCTRL_CHN_MASK 0x4u
#define PIT_TCTRL_CHN_SHIFT 2
#define PIT_TCTRL_CHN_WIDTH 1
#define PIT_TCTRL_CHN(x)
(((uint32_t)((uint32_t)(x))<<PIT_TCTRL_CHN_SHIFT))&PIT_TCTRL_CHN_MASK)
/* TFLG Bit Fields */
#define PIT_TFLG_TIF_MASK 0x1u
#define PIT_TFLG_TIF_SHIFT 0
#define PIT_TFLG_TIF_WIDTH 1
#define PIT_TFLG_TIF(x)
(((uint32_t)((uint32_t)(x))<<PIT_TFLG_TIF_SHIFT))&PIT_TFLG_TIF_MASK)

/*!
 * @}
 */ /* end of group PIT_Register_Masks */

/* PIT - Peripheral instance base addresses */
/** Peripheral PIT base address */
#define PIT_BASE (0x40037000u)
/** Peripheral PIT base pointer */
#define PIT ((PIT_Type *)PIT_BASE)
#define PIT_BASE_PTR (PIT)
/** Array initializer of PIT peripheral base addresses */
#define PIT_BASE_ADDRS { PIT_BASE }
/** Array initializer of PIT peripheral base pointers */
#define PIT_BASE_PTRS { PIT }
/** Interrupt vectors for the PIT peripheral type */
#define PIT IRQS { PIT_IRQn, PIT_IRQn }

/* -----
-- PIT - Register accessor macros
-----
*/
/*!
 * @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor
macros
 * @{
 */

```

```

/* PIT - Register instance definitions */
/* PIT */

#define PIT_MCR                                PIT_MCR_REG(PIT)
#define PIT_LTMR64H                             PIT_LTMR64H_REG(PIT)
#define PIT_LTMR64L                             PIT_LTMR64L_REG(PIT)
#define PIT_LDVAL0                               PIT_LDVAL_REG(PIT,0)
#define PIT_CVAL0                                PIT_CVAL_REG(PIT,0)
#define PIT_TCTRL0                               PIT_TCTRL_REG(PIT,0)
#define PIT_TFLG0                                 PIT_TFLG_REG(PIT,0)
#define PIT_LDVAL1                               PIT_LDVAL_REG(PIT,1)
#define PIT_CVAL1                                PIT_CVAL_REG(PIT,1)
#define PIT_TCTRL1                               PIT_TCTRL_REG(PIT,1)
#define PIT_TFLG1                                 PIT_TFLG_REG(PIT,1)

/* PIT - Register array accessors */
#define PIT_LDVAL(index)                         PIT_LDVAL_REG(PIT,index)
#define PIT_CVAL(index)                           PIT_CVAL_REG(PIT,index)
#define PIT_TCTRL(index)                          PIT_TCTRL_REG(PIT,index)
#define PIT_TFLG(index)                           PIT_TFLG_REG(PIT,index)

/*!
 * @}
 */ /* end of group PIT_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group PIT_Peripheral_Access_Layer */

/* -----
-- PMC Peripheral Access Layer
----- */
/*!

 * @addtogroup PMC_Peripheral_Access_Layer PMC Peripheral Access Layer
 * @{
 */

/** PMC - Register Layout Typedef */
typedef struct {
    __IO uint8_t LVDSC1;                      /**< Low Voltage
Detect Status And Control 1 register, offset: 0x0 */
    __IO uint8_t LVDSC2;                      /**< Low Voltage
Detect Status And Control 2 register, offset: 0x1 */
    __IO uint8_t REGSC;                       /**< Regulator Status
And Control register, offset: 0x2 */
} PMC_Type, *PMC_MemMapPtr;

```

```

/*
-----  

-- PMC - Register accessor macros  

----- */  

  

/*!  

 * @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor  

macros  

 * @{  

 */  

  

/* PMC - Register accessors */  

#define PMC_LVDSC1_REG(base) ((base)->LVDSC1)  

#define PMC_LVDSC2_REG(base) ((base)->LVDSC2)  

#define PMC_REGSC_REG(base) ((base)->REGSC)  

  

/*!  

 * @}  

 */ /* end of group PMC_Register_Accessor_Macros */  

  

/*
-----  

-- PMC Register Masks  

----- */  

  

/*!  

 * @addtogroup PMC_Register_Masks PMC Register Masks  

 * @{  

 */  

  

/* LVDSC1 Bit Fields */  

#define PMC_LVDSC1_LVDV_MASK 0x3u  

#define PMC_LVDSC1_LVDV_SHIFT 0  

#define PMC_LVDSC1_LVDV_WIDTH 2  

#define PMC_LVDSC1_LVDV(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDV_SHIFT))&PMC_LVDSC1_LVDV_MASK)  

#define PMC_LVDSC1_LVDRE_MASK 0x10u  

#define PMC_LVDSC1_LVDRE_SHIFT 4  

#define PMC_LVDSC1_LVDRE_WIDTH 1  

#define PMC_LVDSC1_LVDRE(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDRE_SHIFT))&PMC_LVDSC1_LVDRE_MASK)  

#define PMC_LVDSC1_LVDIE_MASK 0x20u  

#define PMC_LVDSC1_LVDIE_SHIFT 5  

#define PMC_LVDSC1_LVDIE_WIDTH 1  

#define PMC_LVDSC1_LVDIE(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDIE_SHIFT))&PMC_LVDSC1_LVDIE_MASK)  

#define PMC_LVDSC1_LVDACK_MASK 0x40u  

#define PMC_LVDSC1_LVDACK_SHIFT 6

```

```

#define PMC_LVDSC1_LVDACK_WIDTH 1
#define PMC_LVDSC1_LVDACK(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDACK_SHIFT))&PMC_LVDSC1_LVDACK_M
#define PMC_LVDSC1_LVDF_MASK 0x80u
#define PMC_LVDSC1_LVDF_SHIFT 7
#define PMC_LVDSC1_LVDF_WIDTH 1
#define PMC_LVDSC1_LVDF(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC1_LVDF_SHIFT))&PMC_LVDSC1_LVDF_MASK
/* LVDSC2 Bit Fields */
#define PMC_LVDSC2_LVWV_MASK 0x3u
#define PMC_LVDSC2_LVWV_SHIFT 0
#define PMC_LVDSC2_LVWV_WIDTH 2
#define PMC_LVDSC2_LVWV(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWV_SHIFT))&PMC_LVDSC2_LVWV_MASK
#define PMC_LVDSC2_LVWIE_MASK 0x20u
#define PMC_LVDSC2_LVWIE_SHIFT 5
#define PMC_LVDSC2_LVWIE_WIDTH 1
#define PMC_LVDSC2_LVWIE(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWIE_SHIFT))&PMC_LVDSC2_LVWIE_M
#define PMC_LVDSC2_LVWACK_MASK 0x40u
#define PMC_LVDSC2_LVWACK_SHIFT 6
#define PMC_LVDSC2_LVWACK_WIDTH 1
#define PMC_LVDSC2_LVWACK(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWACK_SHIFT))&PMC_LVDSC2_LVWACK_M
#define PMC_LVDSC2_LVWF_MASK 0x80u
#define PMC_LVDSC2_LVWF_SHIFT 7
#define PMC_LVDSC2_LVWF_WIDTH 1
#define PMC_LVDSC2_LVWF(x) (((uint8_t)((uint8_t)(x))<<PMC_LVDSC2_LVWF_SHIFT))&PMC_LVDSC2_LVWF_MASK
/* REGSC Bit Fields */
#define PMC_REGSC_BGBE_MASK 0x1u
#define PMC_REGSC_BGBE_SHIFT 0
#define PMC_REGSC_BGBE_WIDTH 1
#define PMC_REGSC_BGBE(x) (((uint8_t)((uint8_t)(x))<<PMC_REGSC_BGBE_SHIFT))&PMC_REGSC_BGBE_MASK
#define PMC_REGSC_REGONS_MASK 0x4u
#define PMC_REGSC_REGONS_SHIFT 2
#define PMC_REGSC_REGONS_WIDTH 1
#define PMC_REGSC_REGONS(x) (((uint8_t)((uint8_t)(x))<<PMC_REGSC_REGONS_SHIFT))&PMC_REGSC_REGONS_M
#define PMC_REGSC_ACKISO_MASK 0x8u
#define PMC_REGSC_ACKISO_SHIFT 3
#define PMC_REGSC_ACKISO_WIDTH 1
#define PMC_REGSC_ACKISO(x) (((uint8_t)((uint8_t)(x))<<PMC_REGSC_ACKISO_SHIFT))&PMC_REGSC_ACKISO_M
#define PMC_REGSC_BGEN_MASK 0x10u
#define PMC_REGSC_BGEN_SHIFT 4
#define PMC_REGSC_BGEN_WIDTH 1

```

```

#define PMC_REGSC_BGEN(x)
(((uint8_t)((uint8_t)(x))<<PMC_REGSC_BGEN_SHIFT))&PMC_REGSC_BGEN_MASK)

/*!
 * @}
 */
/* end of group PMC_Register_Masks */

/* PMC - Peripheral instance base addresses */
/** Peripheral PMC base address */
#define PMC_BASE (0x4007D000u)
/** Peripheral PMC base pointer */
#define PMC ((PMC_Type *)PMC_BASE)
#define PMC_PTR (PMC)
/** Array initializer of PMC peripheral base addresses */
#define PMC_BASE_ADDRS { PMC_BASE }
/** Array initializer of PMC peripheral base pointers */
#define PMC_BASE_PTRS { PMC }
/** Interrupt vectors for the PMC peripheral type */
#define PMC IRQS { LVD_LVW_IRQn }

/* -----
-- PMC - Register accessor macros
-----
*/
/*!
 * @addtogroup PMC_Register_Accessor_Macros PMC - Register accessor
macros
 * @{
 */

/* PMC - Register instance definitions */
/* PMC */
#define PMC_LVDSC1 PMC_LVDSC1_REG(PMC)
#define PMC_LVDSC2 PMC_LVDSC2_REG(PMC)
#define PMC_REGSC PMC_REGSC_REG(PMC)

/*!
 * @}
 */
/* end of group PMC_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group PMC_Peripheral_Access_Layer */

/* -----
-- PORT Peripheral Access Layer
-----
```

```

----- */
----- */

/*!
 * @addtogroup PORT_Peripheral_Access_Layer PORT Peripheral Access Layer
 * @{
 */

/** PORT - Register Layout Typedef */
typedef struct {
    __IO uint32_t PCR[32];                                /**< Pin Control
Register n, array offset: 0x0, array step: 0x4 */
    __O uint32_t GPCLR;                                    /**< Global Pin
Control Low Register, offset: 0x80 */
    __O uint32_t GPCHR;                                    /**< Global Pin
Control High Register, offset: 0x84 */
    uint8_t RESERVED_0[24];
    __IO uint32_t ISFR;                                    /**< Interrupt Status
Flag Register, offset: 0xA0 */
} PORT_Type, *PORT_MemMapPtr;

/*
----- */
----- */

-- PORT - Register accessor macros
----- */

/*!
 * @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor
macros
 * @{
 */

/* PORT - Register accessors */
#define PORT_PCR_REG(base,index)          ((base)->PCR[index])
#define PORT_PCR_COUNT                    32
#define PORT_GPCLR_REG(base)             ((base)->GPCLR)
#define PORT_GPCHR_REG(base)             ((base)->GPCHR)
#define PORT_ISFR_REG(base)              ((base)->ISFR)

/*!
 * @}
 */
/* end of group PORT_Register_Accessor_Macros */

/*
----- */
----- */

-- PORT Register Masks
----- */

/*!
 * @addtogroup PORT_Register_Masks PORT Register Masks

```

```

/* @{
 */

/* PCR Bit Fields */
#define PORT_PCR_PS_MASK 0x1u
#define PORT_PCR_PS_SHIFT 0
#define PORT_PCR_PS_WIDTH 1
#define PORT_PCR_PS(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_PS_SHIFT))&PORT_PCR_PS_MASK
#define PORT_PCR_PE_MASK 0x2u
#define PORT_PCR_PE_SHIFT 1
#define PORT_PCR_PE_WIDTH 1
#define PORT_PCR_PE(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_PE_SHIFT))&PORT_PCR_PE_MASK
#define PORT_PCR_SRE_MASK 0x4u
#define PORT_PCR_SRE_SHIFT 2
#define PORT_PCR_SRE_WIDTH 1
#define PORT_PCR_SRE(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_SRE_SHIFT))&PORT_PCR_SRE_MASK
#define PORT_PCR_PFE_MASK 0x10u
#define PORT_PCR_PFE_SHIFT 4
#define PORT_PCR_PFE_WIDTH 1
#define PORT_PCR_PFE(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_PFE_SHIFT))&PORT_PCR_PFE_MASK
#define PORT_PCR_DSE_MASK 0x40u
#define PORT_PCR_DSE_SHIFT 6
#define PORT_PCR_DSE_WIDTH 1
#define PORT_PCR_DSE(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_DSE_SHIFT))&PORT_PCR_DSE_MASK
#define PORT_PCR_MUX_MASK 0x700u
#define PORT_PCR_MUX_SHIFT 8
#define PORT_PCR_MUX_WIDTH 3
#define PORT_PCR_MUX(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_MUX_SHIFT))&PORT_PCR_MUX_MASK
#define PORT_PCR_IRQC_MASK 0xF0000u
#define PORT_PCR_IRQC_SHIFT 16
#define PORT_PCR_IRQC_WIDTH 4
#define PORT_PCR_IRQC(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_IRQC_SHIFT))&PORT_PCR_IRQC_MASK
#define PORT_PCR_ISF_MASK 0x1000000u
#define PORT_PCR_ISF_SHIFT 24
#define PORT_PCR_ISF_WIDTH 1
#define PORT_PCR_ISF(x) (((uint32_t)((uint32_t)(x))<<PORT_PCR_ISF_SHIFT))&PORT_PCR_ISF_MASK
/* GPCLR Bit Fields */
#define PORT_GPCLR_GPWD_MASK 0xFFFFu
#define PORT_GPCLR_GPWD_SHIFT 0
#define PORT_GPCLR_GPWD_WIDTH 16
#define PORT_GPCLR_GPWD(x) (((uint32_t)((uint32_t)(x))<<PORT_GPCLR_GPWD_SHIFT))&PORT_GPCLR_GPWD_MASK
#define PORT_GPCLR_GPWE_MASK 0xFFFF0000u
#define PORT_GPCLR_GPWE_SHIFT 16
#define PORT_GPCLR_GPWE_WIDTH 16

```

```

#define PORT_GPCLR_GPWE(x)
(((uint32_t)((uint32_t)(x))<<PORT_GPCLR_GPWE_SHIFT))&PORT_GPCLR_GPWE_MASK
/*
 * GPCHR Bit Fields */
#define PORT_GPCHR_GPWD_MASK 0xFFFFFu
#define PORT_GPCHR_GPWD_SHIFT 0
#define PORT_GPCHR_GPWD_WIDTH 16
#define PORT_GPCHR_GPWD(x)
(((uint32_t)((uint32_t)(x))<<PORT_GPCHR_GPWD_SHIFT))&PORT_GPCHR_GPWD_MASK
/*
 * GPCHR Bit Fields */
#define PORT_GPCHR_GPWE_MASK 0xFFFFF0000u
#define PORT_GPCHR_GPWE_SHIFT 16
#define PORT_GPCHR_GPWE_WIDTH 16
#define PORT_GPCHR_GPWE(x)
(((uint32_t)((uint32_t)(x))<<PORT_GPCHR_GPWE_SHIFT))&PORT_GPCHR_GPWE_MASK
/*
 * ISFR Bit Fields */
#define PORT_ISFR_ISF_MASK 0xFFFFFFFFu
#define PORT_ISFR_ISF_SHIFT 0
#define PORT_ISFR_ISF_WIDTH 32
#define PORT_ISFR_ISF(x)
(((uint32_t)((uint32_t)(x))<<PORT_ISFR_ISF_SHIFT))&PORT_ISFR_ISF_MASK

/*!
 * @}
 */ /* end of group PORT_Register_Masks */

/* PORT - Peripheral instance base addresses */
/** Peripheral PORTA base address */
#define PORTA_BASE (0x40049000u)
/** Peripheral PORTA base pointer */
#define PORTA ((PORT_Type *)PORTA_BASE)
#define PORTA_BASE_PTR (PORTA)
/** Peripheral PORTB base address */
#define PORTB_BASE (0x4004A000u)
/** Peripheral PORTB base pointer */
#define PORTB ((PORT_Type *)PORTB_BASE)
#define PORTB_BASE_PTR (PORTB)
/** Peripheral PORTC base address */
#define PORTC_BASE (0x4004B000u)
/** Peripheral PORTC base pointer */
#define PORTC ((PORT_Type *)PORTC_BASE)
#define PORTC_BASE_PTR (PORTC)
/** Peripheral PORTD base address */
#define PORTD_BASE (0x4004C000u)
/** Peripheral PORTD base pointer */
#define PORTD ((PORT_Type *)PORTD_BASE)
#define PORTD_BASE_PTR (PORTD)
/** Peripheral PORTE base address */

```

```

#define PORTE_BASE (0x4004D000u)
/** Peripheral PORTE base pointer */
#define PORTE ((PORT_Type
*)PORTE_BASE)
#define PORTE_BASE_PTR (PORTE)
/** Array initializer of PORT peripheral base addresses */
#define PORT_BASE_ADDRS { PORTA_BASE,
PORTB_BASE, PORTC_BASE, PORTD_BASE, PORTE_BASE }
/** Array initializer of PORT peripheral base pointers */
#define PORT_BASE_PTRS { PORTA, PORTB, PORTC,
PORTD, PORTE }
/** Interrupt vectors for the PORT peripheral type */
#define PORT IRQS { PORTA_IRQn,
NotAvail_IRQn, PORTD_IRQn, NotAvail_IRQn }

/* -----
-- PORT - Register accessor macros
-----
----- */

/* !
* @addtogroup PORT_Register_Accessor_Macros PORT - Register accessor
macros
* @{
*/
/* PORT - Register instance definitions */
/* PORTA */
#define PORTA_PCR0 PORT_PCR_REG(PORTA, 0)
#define PORTA_PCR1 PORT_PCR_REG(PORTA, 1)
#define PORTA_PCR2 PORT_PCR_REG(PORTA, 2)
#define PORTA_PCR3 PORT_PCR_REG(PORTA, 3)
#define PORTA_PCR4 PORT_PCR_REG(PORTA, 4)
#define PORTA_PCR5 PORT_PCR_REG(PORTA, 5)
#define PORTA_PCR6 PORT_PCR_REG(PORTA, 6)
#define PORTA_PCR7 PORT_PCR_REG(PORTA, 7)
#define PORTA_PCR8 PORT_PCR_REG(PORTA, 8)
#define PORTA_PCR9 PORT_PCR_REG(PORTA, 9)
#define PORTA_PCR10 PORT_PCR_REG(PORTA, 10)
#define PORTA_PCR11 PORT_PCR_REG(PORTA, 11)
#define PORTA_PCR12 PORT_PCR_REG(PORTA, 12)
#define PORTA_PCR13 PORT_PCR_REG(PORTA, 13)
#define PORTA_PCR14 PORT_PCR_REG(PORTA, 14)
#define PORTA_PCR15 PORT_PCR_REG(PORTA, 15)
#define PORTA_PCR16 PORT_PCR_REG(PORTA, 16)
#define PORTA_PCR17 PORT_PCR_REG(PORTA, 17)
#define PORTA_PCR18 PORT_PCR_REG(PORTA, 18)
#define PORTA_PCR19 PORT_PCR_REG(PORTA, 19)
#define PORTA_PCR20 PORT_PCR_REG(PORTA, 20)
#define PORTA_PCR21 PORT_PCR_REG(PORTA, 21)
#define PORTA_PCR22 PORT_PCR_REG(PORTA, 22)
#define PORTA_PCR23 PORT_PCR_REG(PORTA, 23)

```

```

#define PORTA_PCR24
#define PORTA_PCR25
#define PORTA_PCR26
#define PORTA_PCR27
#define PORTA_PCR28
#define PORTA_PCR29
#define PORTA_PCR30
#define PORTA_PCR31
#define PORTA_GPCLR
#define PORTA_GPCHR
#define PORTA_ISFR
/* PORTB */
#define PORTB_PCR0
#define PORTB_PCR1
#define PORTB_PCR2
#define PORTB_PCR3
#define PORTB_PCR4
#define PORTB_PCR5
#define PORTB_PCR6
#define PORTB_PCR7
#define PORTB_PCR8
#define PORTB_PCR9
#define PORTB_PCR10
#define PORTB_PCR11
#define PORTB_PCR12
#define PORTB_PCR13
#define PORTB_PCR14
#define PORTB_PCR15
#define PORTB_PCR16
#define PORTB_PCR17
#define PORTB_PCR18
#define PORTB_PCR19
#define PORTB_PCR20
#define PORTB_PCR21
#define PORTB_PCR22
#define PORTB_PCR23
#define PORTB_PCR24
#define PORTB_PCR25
#define PORTB_PCR26
#define PORTB_PCR27
#define PORTB_PCR28
#define PORTB_PCR29
#define PORTB_PCR30
#define PORTB_PCR31
#define PORTB_GPCLR
#define PORTB_GPCHR
#define PORTB_ISFR
/* PORTC */
#define PORTC_PCR0
#define PORTC_PCR1
#define PORTC_PCR2
#define PORTC_PCR3
#define PORTC_PCR4
#define PORTC_PCR5
PORT_PCR_REG(PORTA, 24)
PORT_PCR_REG(PORTA, 25)
PORT_PCR_REG(PORTA, 26)
PORT_PCR_REG(PORTA, 27)
PORT_PCR_REG(PORTA, 28)
PORT_PCR_REG(PORTA, 29)
PORT_PCR_REG(PORTA, 30)
PORT_PCR_REG(PORTA, 31)
PORT_GPCLR_REG(PORTA)
PORT_GPCHR_REG(PORTA)
PORT_ISFR_REG(PORTA)
PORT_PCR_REG(PORTB, 0)
PORT_PCR_REG(PORTB, 1)
PORT_PCR_REG(PORTB, 2)
PORT_PCR_REG(PORTB, 3)
PORT_PCR_REG(PORTB, 4)
PORT_PCR_REG(PORTB, 5)
PORT_PCR_REG(PORTB, 6)
PORT_PCR_REG(PORTB, 7)
PORT_PCR_REG(PORTB, 8)
PORT_PCR_REG(PORTB, 9)
PORT_PCR_REG(PORTB, 10)
PORT_PCR_REG(PORTB, 11)
PORT_PCR_REG(PORTB, 12)
PORT_PCR_REG(PORTB, 13)
PORT_PCR_REG(PORTB, 14)
PORT_PCR_REG(PORTB, 15)
PORT_PCR_REG(PORTB, 16)
PORT_PCR_REG(PORTB, 17)
PORT_PCR_REG(PORTB, 18)
PORT_PCR_REG(PORTB, 19)
PORT_PCR_REG(PORTB, 20)
PORT_PCR_REG(PORTB, 21)
PORT_PCR_REG(PORTB, 22)
PORT_PCR_REG(PORTB, 23)
PORT_PCR_REG(PORTB, 24)
PORT_PCR_REG(PORTB, 25)
PORT_PCR_REG(PORTB, 26)
PORT_PCR_REG(PORTB, 27)
PORT_PCR_REG(PORTB, 28)
PORT_PCR_REG(PORTB, 29)
PORT_PCR_REG(PORTB, 30)
PORT_PCR_REG(PORTB, 31)
PORT_GPCLR_REG(PORTB)
PORT_GPCHR_REG(PORTB)
PORT_ISFR_REG(PORTB)
PORT_PCR_REG(PORTC, 0)
PORT_PCR_REG(PORTC, 1)
PORT_PCR_REG(PORTC, 2)
PORT_PCR_REG(PORTC, 3)
PORT_PCR_REG(PORTC, 4)
PORT_PCR_REG(PORTC, 5)

```

```

#define PORTC_PCR6          PORT_PCR_REG(PORTC, 6)
#define PORTC_PCR7          PORT_PCR_REG(PORTC, 7)
#define PORTC_PCR8          PORT_PCR_REG(PORTC, 8)
#define PORTC_PCR9          PORT_PCR_REG(PORTC, 9)
#define PORTC_PCR10         PORT_PCR_REG(PORTC, 10)
#define PORTC_PCR11         PORT_PCR_REG(PORTC, 11)
#define PORTC_PCR12         PORT_PCR_REG(PORTC, 12)
#define PORTC_PCR13         PORT_PCR_REG(PORTC, 13)
#define PORTC_PCR14         PORT_PCR_REG(PORTC, 14)
#define PORTC_PCR15         PORT_PCR_REG(PORTC, 15)
#define PORTC_PCR16         PORT_PCR_REG(PORTC, 16)
#define PORTC_PCR17         PORT_PCR_REG(PORTC, 17)
#define PORTC_PCR18         PORT_PCR_REG(PORTC, 18)
#define PORTC_PCR19         PORT_PCR_REG(PORTC, 19)
#define PORTC_PCR20         PORT_PCR_REG(PORTC, 20)
#define PORTC_PCR21         PORT_PCR_REG(PORTC, 21)
#define PORTC_PCR22         PORT_PCR_REG(PORTC, 22)
#define PORTC_PCR23         PORT_PCR_REG(PORTC, 23)
#define PORTC_PCR24         PORT_PCR_REG(PORTC, 24)
#define PORTC_PCR25         PORT_PCR_REG(PORTC, 25)
#define PORTC_PCR26         PORT_PCR_REG(PORTC, 26)
#define PORTC_PCR27         PORT_PCR_REG(PORTC, 27)
#define PORTC_PCR28         PORT_PCR_REG(PORTC, 28)
#define PORTC_PCR29         PORT_PCR_REG(PORTC, 29)
#define PORTC_PCR30         PORT_PCR_REG(PORTC, 30)
#define PORTC_PCR31         PORT_PCR_REG(PORTC, 31)
#define PORTC_GPCLR         PORT_GPCLR_REG(PORTC)
#define PORTC_GPCHR         PORT_GPCHR_REG(PORTC)
#define PORTC_ISFR          PORT_ISFR_REG(PORTC)

/* PORTD */
#define PORTD_PCR0          PORT_PCR_REG(PORTD, 0)
#define PORTD_PCR1          PORT_PCR_REG(PORTD, 1)
#define PORTD_PCR2          PORT_PCR_REG(PORTD, 2)
#define PORTD_PCR3          PORT_PCR_REG(PORTD, 3)
#define PORTD_PCR4          PORT_PCR_REG(PORTD, 4)
#define PORTD_PCR5          PORT_PCR_REG(PORTD, 5)
#define PORTD_PCR6          PORT_PCR_REG(PORTD, 6)
#define PORTD_PCR7          PORT_PCR_REG(PORTD, 7)
#define PORTD_PCR8          PORT_PCR_REG(PORTD, 8)
#define PORTD_PCR9          PORT_PCR_REG(PORTD, 9)
#define PORTD_PCR10         PORT_PCR_REG(PORTD, 10)
#define PORTD_PCR11         PORT_PCR_REG(PORTD, 11)
#define PORTD_PCR12         PORT_PCR_REG(PORTD, 12)
#define PORTD_PCR13         PORT_PCR_REG(PORTD, 13)
#define PORTD_PCR14         PORT_PCR_REG(PORTD, 14)
#define PORTD_PCR15         PORT_PCR_REG(PORTD, 15)
#define PORTD_PCR16         PORT_PCR_REG(PORTD, 16)
#define PORTD_PCR17         PORT_PCR_REG(PORTD, 17)
#define PORTD_PCR18         PORT_PCR_REG(PORTD, 18)
#define PORTD_PCR19         PORT_PCR_REG(PORTD, 19)
#define PORTD_PCR20         PORT_PCR_REG(PORTD, 20)
#define PORTD_PCR21         PORT_PCR_REG(PORTD, 21)
#define PORTD_PCR22         PORT_PCR_REG(PORTD, 22)
#define PORTD_PCR23         PORT_PCR_REG(PORTD, 23)

```

```

#define PORTD_PCR24 PORT_PCR_REG(PORTD, 24)
#define PORTD_PCR25 PORT_PCR_REG(PORTD, 25)
#define PORTD_PCR26 PORT_PCR_REG(PORTD, 26)
#define PORTD_PCR27 PORT_PCR_REG(PORTD, 27)
#define PORTD_PCR28 PORT_PCR_REG(PORTD, 28)
#define PORTD_PCR29 PORT_PCR_REG(PORTD, 29)
#define PORTD_PCR30 PORT_PCR_REG(PORTD, 30)
#define PORTD_PCR31 PORT_PCR_REG(PORTD, 31)
#define PORTD_GPCLR PORT_GPCLR_REG(PORTD)
#define PORTD_GPCHR PORT_GPCHR_REG(PORTD)
#define PORTD_ISFR PORT_ISFR_REG(PORTD)
/* PORTE */
#define PORTE_PCR0 PORT_PCR_REG(PORTE, 0)
#define PORTE_PCR1 PORT_PCR_REG(PORTE, 1)
#define PORTE_PCR2 PORT_PCR_REG(PORTE, 2)
#define PORTE_PCR3 PORT_PCR_REG(PORTE, 3)
#define PORTE_PCR4 PORT_PCR_REG(PORTE, 4)
#define PORTE_PCR5 PORT_PCR_REG(PORTE, 5)
#define PORTE_PCR6 PORT_PCR_REG(PORTE, 6)
#define PORTE_PCR7 PORT_PCR_REG(PORTE, 7)
#define PORTE_PCR8 PORT_PCR_REG(PORTE, 8)
#define PORTE_PCR9 PORT_PCR_REG(PORTE, 9)
#define PORTE_PCR10 PORT_PCR_REG(PORTE, 10)
#define PORTE_PCR11 PORT_PCR_REG(PORTE, 11)
#define PORTE_PCR12 PORT_PCR_REG(PORTE, 12)
#define PORTE_PCR13 PORT_PCR_REG(PORTE, 13)
#define PORTE_PCR14 PORT_PCR_REG(PORTE, 14)
#define PORTE_PCR15 PORT_PCR_REG(PORTE, 15)
#define PORTE_PCR16 PORT_PCR_REG(PORTE, 16)
#define PORTE_PCR17 PORT_PCR_REG(PORTE, 17)
#define PORTE_PCR18 PORT_PCR_REG(PORTE, 18)
#define PORTE_PCR19 PORT_PCR_REG(PORTE, 19)
#define PORTE_PCR20 PORT_PCR_REG(PORTE, 20)
#define PORTE_PCR21 PORT_PCR_REG(PORTE, 21)
#define PORTE_PCR22 PORT_PCR_REG(PORTE, 22)
#define PORTE_PCR23 PORT_PCR_REG(PORTE, 23)
#define PORTE_PCR24 PORT_PCR_REG(PORTE, 24)
#define PORTE_PCR25 PORT_PCR_REG(PORTE, 25)
#define PORTE_PCR26 PORT_PCR_REG(PORTE, 26)
#define PORTE_PCR27 PORT_PCR_REG(PORTE, 27)
#define PORTE_PCR28 PORT_PCR_REG(PORTE, 28)
#define PORTE_PCR29 PORT_PCR_REG(PORTE, 29)
#define PORTE_PCR30 PORT_PCR_REG(PORTE, 30)
#define PORTE_PCR31 PORT_PCR_REG(PORTE, 31)
#define PORTE_GPCLR PORT_GPCLR_REG(PORTE)
#define PORTE_GPCHR PORT_GPCHR_REG(PORTE)
#define PORTE_ISFR PORT_ISFR_REG(PORTE)

/* PORT - Register array accessors */
#define PORTA_PCR(index) PORT_PCR_REG(PORTA, index)
#define PORTB_PCR(index) PORT_PCR_REG(PORTB, index)

```

```

#define PORTC_PCR(index)
PORT_PCR_REG(PORTC, index)
#define PORTD_PCR(index)
PORT_PCR_REG(PORTD, index)
#define PORTE_PCR(index)
PORT_PCR_REG(PORTE, index)

/*!
 * @}
 */ /* end of group PORT_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group PORT_Peripheral_Access_Layer */

/* -----
-- RCM Peripheral Access Layer
-----
*/
/*!
 * @addtogroup RCM_Peripheral_Access_Layer RCM Peripheral Access Layer
 * @{
 */

/** RCM - Register Layout Typedef */
typedef struct {
    __I uint8_t SRS0;                                /**< System Reset
Status Register 0, offset: 0x0 */
    __I uint8_t SRS1;                                /**< System Reset
Status Register 1, offset: 0x1 */
    uint8_t RESERVED_0[2];
    __IO uint8_t RPFC;                               /**< Reset Pin Filter
Control register, offset: 0x4 */
    __IO uint8_t RPFW;                               /**< Reset Pin Filter
Width register, offset: 0x5 */
} RCM_Type, *RCM_MemMapPtr;

/* -----
-- RCM - Register accessor macros
-----
*/
/*!
 * @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor
macros
 * @{
 */

```

```

/* RCM - Register accessors */
#define RCM_SRS0_REG(base)          ((base)->SRS0)
#define RCM_SRS1_REG(base)          ((base)->SRS1)
#define RCM_RPFC_REG(base)          ((base)->RPFC)
#define RCM_RPFW_REG(base)          ((base)->RPFW)

/* !
 * @}
 */ /* end of group RCM_Register_Accessor_Macros */

/*
-----  

-- RCM Register Masks  

-----  

---- */

/* !
 * @addtogroup RCM_Register_Masks RCM Register Masks
 * @{
 */

/* SRS0 Bit Fields */
#define RCM_SRS0_WAKEUP_MASK           0x1u
#define RCM_SRS0_WAKEUP_SHIFT          0
#define RCM_SRS0_WAKEUP_WIDTH          1
#define RCM_SRS0_WAKEUP(x)            (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WAKEUP_SHIFT))&RCM_SRS0_WAKEUP_MASK)
#define RCM_SRS0_LVD_MASK              0x2u
#define RCM_SRS0_LVD_SHIFT             1
#define RCM_SRS0_LVD_WIDTH             1
#define RCM_SRS0_LVD(x)                (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LVD_SHIFT))&RCM_SRS0_LVD_MASK)
#define RCM_SRS0_LOC_MASK              0x4u
#define RCM_SRS0_LOC_SHIFT             2
#define RCM_SRS0_LOC_WIDTH             1
#define RCM_SRS0_LOC(x)                (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOC_SHIFT))&RCM_SRS0_LOC_MASK)
#define RCM_SRS0_LOL_MASK              0x8u
#define RCM_SRS0_LOL_SHIFT             3
#define RCM_SRS0_LOL_WIDTH             1
#define RCM_SRS0_LOL(x)                (((uint8_t)((uint8_t)(x)<<RCM_SRS0_LOL_SHIFT))&RCM_SRS0_LOL_MASK)
#define RCM_SRS0_WDOG_MASK             0x20u
#define RCM_SRS0_WDOG_SHIFT            5
#define RCM_SRS0_WDOG_WIDTH            1
#define RCM_SRS0_WDOG(x)                (((uint8_t)((uint8_t)(x)<<RCM_SRS0_WDOG_SHIFT))&RCM_SRS0_WDOG_MASK)
#define RCM_SRS0_PIN_MASK              0x40u
#define RCM_SRS0_PIN_SHIFT             6
#define RCM_SRS0_PIN_WIDTH             1
#define RCM_SRS0_PIN(x)                (((uint8_t)((uint8_t)(x)<<RCM_SRS0_PIN_SHIFT))&RCM_SRS0_PIN_MASK)
#define RCM_SRS0_POR_MASK              0x80u

```

```

#define RCM_SRS0_POR_SHIFT 7
#define RCM_SRS0_POR_WIDTH 1
#define RCM_SRS0_POR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS0_POR_SHIFT))&RCM_SRS0_POR_MASK)
/* SRS1 Bit Fields */
#define RCM_SRS1_LOCKUP_MASK 0x2u
#define RCM_SRS1_LOCKUP_SHIFT 1
#define RCM_SRS1_LOCKUP_WIDTH 1
#define RCM_SRS1_LOCKUP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_LOCKUP_SHIFT))&RCM_SRS1_LOCKUP_MASK)
#define RCM_SRS1_SW_MASK 0x4u
#define RCM_SRS1_SW_SHIFT 2
#define RCM_SRS1_SW_WIDTH 1
#define RCM_SRS1_SW(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SW_SHIFT))&RCM_SRS1_SW_MASK)
#define RCM_SRS1_MDM_AP_MASK 0x8u
#define RCM_SRS1_MDM_AP_SHIFT 3
#define RCM_SRS1_MDM_AP_WIDTH 1
#define RCM_SRS1_MDM_AP(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_MDM_AP_SHIFT))&RCM_SRS1_MDM_AP_MASK)
#define RCM_SRS1_SACKERR_MASK 0x20u
#define RCM_SRS1_SACKERR_SHIFT 5
#define RCM_SRS1_SACKERR_WIDTH 1
#define RCM_SRS1_SACKERR(x)
(((uint8_t)((uint8_t)(x)<<RCM_SRS1_SACKERR_SHIFT))&RCM_SRS1_SACKERR_MASK)
/* RPFC Bit Fields */
#define RCM_RPFC_RSTFLTSRW_MASK 0x3u
#define RCM_RPFC_RSTFLTSRW_SHIFT 0
#define RCM_RPFC_RSTFLTSRW_WIDTH 2
#define RCM_RPFC_RSTFLTSRW(x)
(((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSRW_SHIFT))&RCM_RPFC_RSTFLTSRW_MASK)
#define RCM_RPFC_RSTFLTSS_MASK 0x4u
#define RCM_RPFC_RSTFLTSS_SHIFT 2
#define RCM_RPFC_RSTFLTSS_WIDTH 1
#define RCM_RPFC_RSTFLTSS(x)
(((uint8_t)((uint8_t)(x)<<RCM_RPFC_RSTFLTSS_SHIFT))&RCM_RPFC_RSTFLTSS_MASK)
/* RPFW Bit Fields */
#define RCM_RPFW_RSTFLTSEL_MASK 0x1Fu
#define RCM_RPFW_RSTFLTSEL_SHIFT 0
#define RCM_RPFW_RSTFLTSEL_WIDTH 5
#define RCM_RPFW_RSTFLTSEL(x)
(((uint8_t)((uint8_t)(x)<<RCM_RPFW_RSTFLTSEL_SHIFT))&RCM_RPFW_RSTFLTSEL_MASK)

/*
 * @}
 */ /* end of group RCM_Register_Masks */

/* RCM - Peripheral instance base addresses */
/** Peripheral RCM base address */

```

```

#define RCM_BASE (0x4007F000u)
/** Peripheral RCM base pointer */
#define RCM ((RCM_Type *) RCM_BASE)
#define RCM_BASE_PTR (RCM)
/** Array initializer of RCM peripheral base addresses */
#define RCM_BASE_ADDRS { RCM_BASE }
/** Array initializer of RCM peripheral base pointers */
#define RCM_BASE_PTRS { RCM }

/* -----
-- RCM - Register accessor macros
-----
*/
/*!
 * @addtogroup RCM_Register_Accessor_Macros RCM - Register accessor
macros
 * @{
 */

/* RCM - Register instance definitions */
/* RCM */
#define RCM_SRS0 RCM_SRS0_REG(RCM)
#define RCM_SRS1 RCM_SRS1_REG(RCM)
#define RCM_RPFC RCM_RPFC_REG(RCM)
#define RCM_RPFW RCM_RPFW_REG(RCM)

/*!
 * @}
 */
/* end of group RCM_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group RCM_Peripheral_Access_Layer */

/* -----
-- ROM Peripheral Access Layer
-----
*/
/*!
 * @addtogroup ROM_Peripheral_Access_Layer ROM Peripheral Access Layer
 * @{
 */

/** ROM - Register Layout Typedef */
typedef struct {
    __I uint32_t ENTRY[3];                                /**< Entry, array
offset: 0x0, array step: 0x4 */

```

```

    __I uint32_t TABLEMARK;                                /**< End of Table
Marker Register, offset: 0xC */
    uint8_t RESERVED_0[4028];
    __I uint32_t SYSACCESS;                             /**< System Access
Register, offset: 0xFCC */
    __I uint32_t PERIPHID4;                            /**< Peripheral ID
Register, offset: 0xFD0 */
    __I uint32_t PERIPHID5;                            /**< Peripheral ID
Register, offset: 0xFD4 */
    __I uint32_t PERIPHID6;                            /**< Peripheral ID
Register, offset: 0xFD8 */
    __I uint32_t PERIPHID7;                            /**< Peripheral ID
Register, offset: 0xFDC */
    __I uint32_t PERIPHID0;                            /**< Peripheral ID
Register, offset: 0xFE0 */
    __I uint32_t PERIPHID1;                            /**< Peripheral ID
Register, offset: 0xFE4 */
    __I uint32_t PERIPHID2;                            /**< Peripheral ID
Register, offset: 0xFE8 */
    __I uint32_t PERIPHID3;                            /**< Peripheral ID
Register, offset: 0xFEC */
    __I uint32_t COMPID[4];                           /**< Component ID
Register, array offset: 0xFF0, array step: 0x4 */
} ROM_Type, *ROM_MemMapPtr;

/*
-----
-- ROM - Register accessor macros
-----
*/
/*!
 * @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor
macros
 * @{
 */

/* ROM - Register accessors */
#define ROM_ENTRY_REG(base, index)          ((base) ->ENTRY[index])
#define ROM_ENTRY_COUNT                     3
#define ROM_TABLEMARK_REG(base)            ((base) ->TABLEMARK)
#define ROM_SYSACCESS_REG(base)           ((base) ->SYSACCESS)
#define ROM_PERIPHID4_REG(base)           ((base) ->PERIPHID4)
#define ROM_PERIPHID5_REG(base)           ((base) ->PERIPHID5)
#define ROM_PERIPHID6_REG(base)           ((base) ->PERIPHID6)
#define ROM_PERIPHID7_REG(base)           ((base) ->PERIPHID7)
#define ROM_PERIPHID0_REG(base)           ((base) ->PERIPHID0)
#define ROM_PERIPHID1_REG(base)           ((base) ->PERIPHID1)
#define ROM_PERIPHID2_REG(base)           ((base) ->PERIPHID2)
#define ROM_PERIPHID3_REG(base)           ((base) ->PERIPHID3)
#define ROM_COMPID_REG(base, index)        ((base) ->COMPID[index])
#define ROM_COMPID_COUNT                  4

```

```

/*!
 * @}
 */
/* end of group ROM_Register_Accessor_Macros */

/*
-----  

-- ROM Register Masks  

-----  

*/
/*!  

 * @addtogroup ROM_Register_Masks ROM Register Masks  

 * @{
 *  

 */

/* ENTRY Bit Fields */
#define ROM_ENTRY_ENTRY_MASK          0xFFFFFFFFu
#define ROM_ENTRY_ENTRY_SHIFT         0
#define ROM_ENTRY_ENTRY_WIDTH        32
#define ROM_ENTRY_ENTRY(x)  

(((uint32_t) (((uint32_t)(x))<<ROM_ENTRY_ENTRY_SHIFT)) & ROM_ENTRY_ENTRY_MASK)

/* TABLEMARK Bit Fields */
#define ROM_TABLEMARK_MARK_MASK       0xFFFFFFFFu
#define ROM_TABLEMARK_MARK_SHIFT     0
#define ROM_TABLEMARK_MARK_WIDTH    32
#define ROM_TABLEMARK_MARK(x)  

(((uint32_t) (((uint32_t)(x))<<ROM_TABLEMARK_MARK_SHIFT)) & ROM_TABLEMARK_MARK_MASK)

/* SYSACCESS Bit Fields */
#define ROM_SYSACCESS_SYSACCESS_MASK  0xFFFFFFFFu
#define ROM_SYSACCESS_SYSACCESS_SHIFT 0
#define ROM_SYSACCESS_SYSACCESS_WIDTH 32
#define ROM_SYSACCESS_SYSACCESS(x)  

(((uint32_t) (((uint32_t)(x))<<ROM_SYSACCESS_SYSACCESS_SHIFT)) & ROM_SYSACCESS_SYSACCESS_MASK)

/* PERIPHID4 Bit Fields */
#define ROM_PERIPHID4_PERIPHID_MASK  0xFFFFFFFFu
#define ROM_PERIPHID4_PERIPHID_SHIFT 0
#define ROM_PERIPHID4_PERIPHID_WIDTH 32
#define ROM_PERIPHID4_PERIPHID(x)  

(((uint32_t) (((uint32_t)(x))<<ROM_PERIPHID4_PERIPHID_SHIFT)) & ROM_PERIPHID4_PERIPHID_MASK)

/* PERIPHID5 Bit Fields */
#define ROM_PERIPHID5_PERIPHID_MASK  0xFFFFFFFFu
#define ROM_PERIPHID5_PERIPHID_SHIFT 0
#define ROM_PERIPHID5_PERIPHID_WIDTH 32
#define ROM_PERIPHID5_PERIPHID(x)  

(((uint32_t) (((uint32_t)(x))<<ROM_PERIPHID5_PERIPHID_SHIFT)) & ROM_PERIPHID5_PERIPHID_MASK)

/* PERIPHID6 Bit Fields */
#define ROM_PERIPHID6_PERIPHID_MASK  0xFFFFFFFFu
#define ROM_PERIPHID6_PERIPHID_SHIFT 0

```

```

#define ROM_PERIPHID6_PERIPHID_WIDTH          32
#define ROM_PERIPHID6_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID6_PERIPHID_SHIFT))&ROM_PERIPHID6_PERIPHID_MASK
/* PERIPHID7 Bit Fields */
#define ROM_PERIPHID7_PERIPHID_MASK           0xFFFFFFFFu
#define ROM_PERIPHID7_PERIPHID_SHIFT          0
#define ROM_PERIPHID7_PERIPHID_WIDTH          32
#define ROM_PERIPHID7_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID7_PERIPHID_SHIFT))&ROM_PERIPHID7_PERIPHID_MASK
/* PERIPHID0 Bit Fields */
#define ROM_PERIPHID0_PERIPHID_MASK           0xFFFFFFFFu
#define ROM_PERIPHID0_PERIPHID_SHIFT          0
#define ROM_PERIPHID0_PERIPHID_WIDTH          32
#define ROM_PERIPHID0_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID0_PERIPHID_SHIFT))&ROM_PERIPHID0_PERIPHID_MASK
/* PERIPHID1 Bit Fields */
#define ROM_PERIPHID1_PERIPHID_MASK           0xFFFFFFFFu
#define ROM_PERIPHID1_PERIPHID_SHIFT          0
#define ROM_PERIPHID1_PERIPHID_WIDTH          32
#define ROM_PERIPHID1_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID1_PERIPHID_SHIFT))&ROM_PERIPHID1_PERIPHID_MASK
/* PERIPHID2 Bit Fields */
#define ROM_PERIPHID2_PERIPHID_MASK           0xFFFFFFFFu
#define ROM_PERIPHID2_PERIPHID_SHIFT          0
#define ROM_PERIPHID2_PERIPHID_WIDTH          32
#define ROM_PERIPHID2_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID2_PERIPHID_SHIFT))&ROM_PERIPHID2_PERIPHID_MASK
/* PERIPHID3 Bit Fields */
#define ROM_PERIPHID3_PERIPHID_MASK           0xFFFFFFFFu
#define ROM_PERIPHID3_PERIPHID_SHIFT          0
#define ROM_PERIPHID3_PERIPHID_WIDTH          32
#define ROM_PERIPHID3_PERIPHID(x)              (((uint32_t)((uint32_t)(x))<<ROM_PERIPHID3_PERIPHID_SHIFT))&ROM_PERIPHID3_PERIPHID_MASK
/* COMPID Bit Fields */
#define ROM_COMPID_COMPID_MASK                0xFFFFFFFFu
#define ROM_COMPID_COMPID_SHIFT               0
#define ROM_COMPID_COMPID_WIDTH              32
#define ROM_COMPID_COMPID(x)                 (((uint32_t)((uint32_t)(x))<<ROM_COMPID_COMPID_SHIFT))&ROM_COMPID_COMPID_MASK
/*
 * @}
 */ /* end of group ROM_Register_Masks */

/* ROM - Peripheral instance base addresses */
/** Peripheral ROM base address */

```

```

#define ROM_BASE (0xF0002000u)
/** Peripheral ROM base pointer */
#define ROM ((ROM_Type *)ROM_BASE)
#define ROM_BASE_PTR (ROM)
/** Array initializer of ROM peripheral base addresses */
#define ROM_BASE_ADDRS { ROM_BASE }
/** Array initializer of ROM peripheral base pointers */
#define ROM_BASE_PTRS { ROM }

/* -----
-- ROM - Register accessor macros
-----
*/
/*!
 * @addtogroup ROM_Register_Accessor_Macros ROM - Register accessor
macros
 * @{
 */

/* ROM - Register instance definitions */
/* ROM */
#define ROM_ENTRY0 ROM_ENTRY_REG(ROM, 0)
#define ROM_ENTRY1 ROM_ENTRY_REG(ROM, 1)
#define ROM_ENTRY2 ROM_ENTRY_REG(ROM, 2)
#define ROM_TABLEMARK ROM_TABLEMARK_REG(ROM)
#define ROM_SYSACCESS ROM_SYSACCESS_REG(ROM)
#define ROM_PERIPHID4 ROM_PERIPHID4_REG(ROM)
#define ROM_PERIPHID5 ROM_PERIPHID5_REG(ROM)
#define ROM_PERIPHID6 ROM_PERIPHID6_REG(ROM)
#define ROM_PERIPHID7 ROM_PERIPHID7_REG(ROM)
#define ROM_PERIPHIDO ROM_PERIPHIDO_REG(ROM)
#define ROM_PERIPHID1 ROM_PERIPHID1_REG(ROM)
#define ROM_PERIPHID2 ROM_PERIPHID2_REG(ROM)
#define ROM_PERIPHID3 ROM_PERIPHID3_REG(ROM)
#define ROM_COMPID0 ROM_COMPID_REG(ROM, 0)
#define ROM_COMPID1 ROM_COMPID_REG(ROM, 1)
#define ROM_COMPID2 ROM_COMPID_REG(ROM, 2)
#define ROM_COMPID3 ROM_COMPID_REG(ROM, 3)

/* ROM - Register array accessors */
#define ROM_ENTRY(index) ROM_ENTRY_REG(ROM, index)
#define ROM_COMPID(index) ROM_COMPID_REG(ROM, index)

/*!
 * @}
 */
/* /* end of group ROM_Register_Accessor_Macros */

/*
 * @{
 */

```

```

 */ /* end of group ROM_Peripheral_Access_Layer */

/*
-----
-- RTC Peripheral Access Layer
-----
*/

/*!
 * @addtogroup RTC_Peripheral_Access_Layer RTC Peripheral Access Layer
 * @{
 */

/** RTC - Register Layout Typedef */
typedef struct {
    __IO uint32_t TSR;                                /**< RTC Time Seconds
Register, offset: 0x0 */
    __IO uint32_t TPR;                                /**< RTC Time
Prescaler Register, offset: 0x4 */
    __IO uint32_t TAR;                                /**< RTC Time Alarm
Register, offset: 0x8 */
    __IO uint32_t TCR;                                /**< RTC Time
Compensation Register, offset: 0xC */
    __IO uint32_t CR;                                 /**< RTC Control
Register, offset: 0x10 */
    __IO uint32_t SR;                                 /**< RTC Status
Register, offset: 0x14 */
    __IO uint32_t LR;                                 /**< RTC Lock
Register, offset: 0x18 */
    __IO uint32_t IER;                               /**< RTC Interrupt
Enable Register, offset: 0x1C */
} RTC_Type, *RTC_MemMapPtr;

/*
-----
-- RTC - Register accessor macros
-----
*/

/*!
 * @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor
macros
 * @{
 */

/* RTC - Register accessors */
#define RTC_TSREG(base)          ((base)->TSR)
#define RTC_TPREG(base)          ((base)->TPR)
#define RTC_TARREG(base)         ((base)->TAR)
#define RTC_TCRREG(base)         ((base)->TCR)
#define RTC_CRRREG(base)         ((base)->CR)
#define RTC_SRREG(base)          ((base)->SR)

```

```

#define RTC_LR_REG(base)          ((base)->LR)
#define RTC_IER_REG(base)         ((base)->IER)

/*!
 * @}
 */
/* end of group RTC_Register_Accessor_Macros */

/*
-----  

-- RTC Register Masks  

-----  

----- */

/*!  

 * @addtogroup RTC_Register_Masks RTC Register Masks
 * @{
 */

/* TSR Bit Fields */
#define RTC_TSR_TSRS_MASK          0xFFFFFFFFu
#define RTC_TSR_TSRS_SHIFT         0
#define RTC_TSR_TSRS_WIDTH         32
#define RTC_TSR_TSRS(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TSRS_SHIFT))&RTC_TSRS_MASK)

/* TPR Bit Fields */
#define RTC_TPR_TPR_MASK           0xFFFFu
#define RTC_TPR_TPR_SHIFT          0
#define RTC_TPR_TPR_WIDTH          16
#define RTC_TPR_TPR(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TPR_TPR_SHIFT))&RTC_TPR_TPR_MASK

/* TAR Bit Fields */
#define RTC_TAR_TAR_MASK           0xFFFFFFFFu
#define RTC_TAR_TAR_SHIFT          0
#define RTC_TAR_TAR_WIDTH          32
#define RTC_TAR_TAR(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TAR_TAR_SHIFT))&RTC_TAR_TAR_MASK

/* TCR Bit Fields */
#define RTC_TCR_TCR_MASK           0xFFu
#define RTC_TCR_TCR_SHIFT          0
#define RTC_TCR_TCR_WIDTH          8
#define RTC_TCR_TCR(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TCR_TCR_SHIFT))&RTC_TCR_TCR_MASK

#define RTC_TCR_CIR_MASK           0xFF00u
#define RTC_TCR_CIR_SHIFT          8
#define RTC_TCR_CIR_WIDTH          8
#define RTC_TCR_CIR(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TCR_CIR_SHIFT))&RTC_TCR_CIR_MASK

#define RTC_TCR_TCV_MASK           0xFF000u
#define RTC_TCR_TCV_SHIFT          16
#define RTC_TCR_TCV_WIDTH          8
#define RTC_TCR_TCV(x)  

(((uint32_t)((uint32_t)(x))<<RTC_TCR_TCV_SHIFT))&RTC_TCR_TCV_MASK

#define RTC_TCR_CIC_MASK           0xFF00000u

```

#define RTC_TCR_CIC_SHIFT	24
#define RTC_TCR_CIC_WIDTH	8
#define RTC_TCR_CIC(x)	
((uint32_t)((uint32_t)(x))<<RTC_TCR_CIC_SHIFT))&RTC_TCR_CIC_MASK)	
/* CR Bit Fields */	
#define RTC_CR_SWR_MASK	0x1u
#define RTC_CR_SWR_SHIFT	0
#define RTC_CR_SWR_WIDTH	1
#define RTC_CR_SWR(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_SWR_SHIFT))&RTC_CR_SWR_MASK)	
#define RTC_CR_WPE_MASK	0x2u
#define RTC_CR_WPE_SHIFT	1
#define RTC_CR_WPE_WIDTH	1
#define RTC_CR_WPE(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_WPE_SHIFT))&RTC_CR_WPE_MASK)	
#define RTC_CR_SUP_MASK	0x4u
#define RTC_CR_SUP_SHIFT	2
#define RTC_CR_SUP_WIDTH	1
#define RTC_CR_SUP(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_SUP_SHIFT))&RTC_CR_SUP_MASK)	
#define RTC_CR_UM_MASK	0x8u
#define RTC_CR_UM_SHIFT	3
#define RTC_CR_UM_WIDTH	1
#define RTC_CR_UM(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_UM_SHIFT))&RTC_CR_UM_MASK)	
#define RTC_CR_OSCE_MASK	0x100u
#define RTC_CR_OSCE_SHIFT	8
#define RTC_CR_OSCE_WIDTH	1
#define RTC_CR_OSCE(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_OSCE_SHIFT))&RTC_CR_OSCE_MASK)	
#define RTC_CR_CLKO_MASK	0x200u
#define RTC_CR_CLKO_SHIFT	9
#define RTC_CR_CLKO_WIDTH	1
#define RTC_CR_CLKO(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_CLKO_SHIFT))&RTC_CR_CLKO_MASK)	
#define RTC_CR_SC16P_MASK	0x400u
#define RTC_CR_SC16P_SHIFT	10
#define RTC_CR_SC16P_WIDTH	1
#define RTC_CR_SC16P(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_SC16P_SHIFT))&RTC_CR_SC16P_MASK)	
#define RTC_CR_SC8P_MASK	0x800u
#define RTC_CR_SC8P_SHIFT	11
#define RTC_CR_SC8P_WIDTH	1
#define RTC_CR_SC8P(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_SC8P_SHIFT))&RTC_CR_SC8P_MASK)	
#define RTC_CR_SC4P_MASK	0x1000u
#define RTC_CR_SC4P_SHIFT	12
#define RTC_CR_SC4P_WIDTH	1
#define RTC_CR_SC4P(x)	
((uint32_t)((uint32_t)(x))<<RTC_CR_SC4P_SHIFT))&RTC_CR_SC4P_MASK)	
#define RTC_CR_SC2P_MASK	0x2000u
#define RTC_CR_SC2P_SHIFT	13
#define RTC_CR_SC2P_WIDTH	1

```

#define RTC_CR_SC2P(x)
(((uint32_t)((uint32_t)(x))<<RTC_CR_SC2P_SHIFT))&RTC_CR_SC2P_MASK
/* SR Bit Fields */
#define RTC_SR_TIF_MASK 0x1u
#define RTC_SR_TIF_SHIFT 0
#define RTC_SR_TIF_WIDTH 1
#define RTC_SR_TIF(x)
(((uint32_t)((uint32_t)(x))<<RTC_SR_TIF_SHIFT))&RTC_SR_TIF_MASK
#define RTC_SR_TOF_MASK 0x2u
#define RTC_SR_TOF_SHIFT 1
#define RTC_SR_TOF_WIDTH 1
#define RTC_SR_TOF(x)
(((uint32_t)((uint32_t)(x))<<RTC_SR_TOF_SHIFT))&RTC_SR_TOF_MASK
#define RTC_SR_TAF_MASK 0x4u
#define RTC_SR_TAF_SHIFT 2
#define RTC_SR_TAF_WIDTH 1
#define RTC_SR_TAF(x)
(((uint32_t)((uint32_t)(x))<<RTC_SR_TAF_SHIFT))&RTC_SR_TAF_MASK
#define RTC_SR_TCE_MASK 0x10u
#define RTC_SR_TCE_SHIFT 4
#define RTC_SR_TCE_WIDTH 1
#define RTC_SR_TCE(x)
(((uint32_t)((uint32_t)(x))<<RTC_SR_TCE_SHIFT))&RTC_SR_TCE_MASK
/* LR Bit Fields */
#define RTC_LR_TCL_MASK 0x8u
#define RTC_LR_TCL_SHIFT 3
#define RTC_LR_TCL_WIDTH 1
#define RTC_LR_TCL(x)
(((uint32_t)((uint32_t)(x))<<RTC_LR_TCL_SHIFT))&RTC_LR_TCL_MASK
#define RTC_LR_CRL_MASK 0x10u
#define RTC_LR_CRL_SHIFT 4
#define RTC_LR_CRL_WIDTH 1
#define RTC_LR_CRL(x)
(((uint32_t)((uint32_t)(x))<<RTC_LR_CRL_SHIFT))&RTC_LR_CRL_MASK
#define RTC_LR_SRL_MASK 0x20u
#define RTC_LR_SRL_SHIFT 5
#define RTC_LR_SRL_WIDTH 1
#define RTC_LR_SRL(x)
(((uint32_t)((uint32_t)(x))<<RTC_LR_SRL_SHIFT))&RTC_LR_SRL_MASK
#define RTC_LR_LRL_MASK 0x40u
#define RTC_LR_LRL_SHIFT 6
#define RTC_LR_LRL_WIDTH 1
#define RTC_LR_LRL(x)
(((uint32_t)((uint32_t)(x))<<RTC_LR_LRL_SHIFT))&RTC_LR_LRL_MASK
/* IER Bit Fields */
#define RTC_IER_TIIE_MASK 0x1u
#define RTC_IER_TIIE_SHIFT 0
#define RTC_IER_TIIE_WIDTH 1
#define RTC_IER_TIIE(x)
(((uint32_t)((uint32_t)(x))<<RTC_IER_TIIE_SHIFT))&RTC_IER_TIIE_MASK
#define RTC_IER_TOIE_MASK 0x2u
#define RTC_IER_TOIE_SHIFT 1
#define RTC_IER_TOIE_WIDTH 1

```

```

#define RTC_IER_TOIE(x)
(((uint32_t)((uint32_t)(x))<<RTC_IER_TOIE_SHIFT))&RTC_IER_TOIE_MASK)
#define RTC_IER_TAIE_MASK 0x4u
#define RTC_IER_TAIE_SHIFT 2
#define RTC_IER_TAIE_WIDTH 1
#define RTC_IER_TAIE(x)
(((uint32_t)((uint32_t)(x))<<RTC_IER_TAIE_SHIFT))&RTC_IER_TAIE_MASK)
#define RTC_IER_TSIE_MASK 0x10u
#define RTC_IER_TSIE_SHIFT 4
#define RTC_IER_TSIE_WIDTH 1
#define RTC_IER_TSIE(x)
(((uint32_t)((uint32_t)(x))<<RTC_IER_TSIE_SHIFT))&RTC_IER_TSIE_MASK)
#define RTC_IER_WPON_MASK 0x80u
#define RTC_IER_WPON_SHIFT 7
#define RTC_IER_WPON_WIDTH 1
#define RTC_IER_WPON(x)
(((uint32_t)((uint32_t)(x))<<RTC_IER_WPON_SHIFT))&RTC_IER_WPON_MASK)

/*!
 * @}
 */ /* end of group RTC_Register_Masks */

/* RTC - Peripheral instance base addresses */
/** Peripheral RTC base address */
#define RTC_BASE (0x4003D000u)
/** Peripheral RTC base pointer */
#define RTC ((RTC_Type *)RTC_BASE)
#define RTC_PTR (RTC)
/** Array initializer of RTC peripheral base addresses */
#define RTC_BASE_ADDRS { RTC_BASE }
/** Array initializer of RTC peripheral base pointers */
#define RTC_BASE_PTRS { RTC }
/** Interrupt vectors for the RTC peripheral type */
#define RTC IRQS { RTC IRQn }
#define RTC_SECONDS_IRQS { RTC_Seconds IRQn }

/* -----
-- RTC - Register accessor macros
----- */
/*!
 * @addtogroup RTC_Register_Accessor_Macros RTC - Register accessor
macros
 * @{
 */

/* RTC - Register instance definitions */
/* RTC */
#define RTC_TSR RTC_TSR_REG(RTC)
#define RTC_TPR RTC_TPR_REG(RTC)

```

```

#define RTC_TAR           RTC_TAR_REG(RTC)
#define RTC_TCR           RTC_TCR_REG(RTC)
#define RTC_CR            RTC_CR_REG(RTC)
#define RTC_SR            RTC_SR_REG(RTC)
#define RTC_LR            RTC_LR_REG(RTC)
#define RTC_IER           RTC_IER_REG(RTC)

/*!
 * @}
 */ /* end of group RTC_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group RTC_Peripheral_Access_Layer */

/* -----
-- SIM Peripheral Access Layer
-----
*/
/*!
 * @addtogroup SIM_Peripheral_Access_Layer SIM Peripheral Access Layer
 * @{
 */

/** SIM - Register Layout Typedef */
typedef struct {
    __IO uint32_t SOPT1;                                /**< System Options
Register 1, offset: 0x0 */
    __IO uint32_t SOPT1CFG;                             /**< SOPT1
Configuration Register, offset: 0x4 */
    uint8_t RESERVED_0[4092];
    __IO uint32_t SOPT2;                                /**< System Options
Register 2, offset: 0x1004 */
    uint8_t RESERVED_1[4];
    __IO uint32_t SOPT4;                                /**< System Options
Register 4, offset: 0x100C */
    __IO uint32_t SOPT5;                                /**< System Options
Register 5, offset: 0x1010 */
    uint8_t RESERVED_2[4];
    __IO uint32_t SOPT7;                                /**< System Options
Register 7, offset: 0x1018 */
    uint8_t RESERVED_3[8];
    __I uint32_t SDID;                                 /**< System Device
Identification Register, offset: 0x1024 */
    uint8_t RESERVED_4[12];
    __IO uint32_t SCGC4;                               /**< System Clock
Gating Control Register 4, offset: 0x1034 */
    __IO uint32_t SCGC5;                               /**< System Clock
Gating Control Register 5, offset: 0x1038 */

```

```

    __IO uint32_t SCGC6;                                /**< System Clock
Gating Control Register 6, offset: 0x103C */
    __IO uint32_t SCGC7;                                /**< System Clock
Gating Control Register 7, offset: 0x1040 */
    __IO uint32_t CLKDIV1;                             /**< System Clock
Divider Register 1, offset: 0x1044 */
        uint8_t RESERVED_5[4];
    __IO uint32_t FCFG1;                               /**< Flash
Configuration Register 1, offset: 0x104C */
        I uint32_t FCFG2;                            /**< Flash
Configuration Register 2, offset: 0x1050 */
        uint8_t RESERVED_6[4];
        I uint32_t UIDMH;                           /**< Unique
Identification Register Mid-High, offset: 0x1058 */
        I uint32_t UIDML;                           /**< Unique
Identification Register Mid Low, offset: 0x105C */
        I uint32_t UIDL;                            /**< Unique
Identification Register Low, offset: 0x1060 */
        uint8_t RESERVED_7[156];
    __IO uint32_t COPC;                               /**< COP Control
Register, offset: 0x1100 */
    __O uint32_t SRVCOP;                            /**< Service COP
Register, offset: 0x1104 */
} SIM_Type, *SIM_MemMapPtr;

/* -----
-- SIM - Register accessor macros
-----
*/
/*!
 * @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor
macros
 * @{
 */

/* SIM - Register accessors */
#define SIM_SOPT1_REG(base)                      ((base) -> SOPT1)
#define SIM_SOPT1CFG_REG(base)                   ((base) -> SOPT1CFG)
#define SIM_SOPT2_REG(base)                      ((base) -> SOPT2)
#define SIM_SOPT4_REG(base)                      ((base) -> SOPT4)
#define SIM_SOPT5_REG(base)                      ((base) -> SOPT5)
#define SIM_SOPT7_REG(base)                      ((base) -> SOPT7)
#define SIM_SDID_REG(base)                      ((base) -> SDID)
#define SIM_SCGC4_REG(base)                     ((base) -> SCGC4)
#define SIM_SCGC5_REG(base)                     ((base) -> SCGC5)
#define SIM_SCGC6_REG(base)                     ((base) -> SCGC6)
#define SIM_SCGC7_REG(base)                     ((base) -> SCGC7)
#define SIM_CLKDIV1_REG(base)                   ((base) -> CLKDIV1)
#define SIM_FCFG1_REG(base)                     ((base) -> FCFG1)
#define SIM_FCFG2_REG(base)                     ((base) -> FCFG2)
#define SIM_UIDMH_REG(base)                    ((base) -> UIDMH)

```

```

#define SIM_UIDML_REG(base)          ((base)->UIDML)
#define SIM_UIDL_REG(base)           ((base)->UIDL)
#define SIM_COPC_REG(base)           ((base)->COPC)
#define SIM_SRVCOP_REG(base)         ((base)->SRVCOP)

/* !
 * @}
 */ /* end of group SIM_Register_Accessor_Macros */

/*
-----  

-- SIM Register Masks  

-----  

---- */

/* !
 * @addtogroup SIM_Register_Masks SIM Register Masks
 * @{
 */

/* SOPT1 Bit Fields */
#define SIM_SOPT1_OSC32KSEL_MASK      0xC0000u
#define SIM_SOPT1_OSC32KSEL_SHIFT     18
#define SIM_SOPT1_OSC32KSEL_WIDTH    2
#define SIM_SOPT1_OSC32KSEL(x)        (((uint32_t)((uint32_t)(x))<<SIM_SOPT1_OSC32KSEL_SHIFT))&SIM_SOPT1_OSC32KSEL_MASK
#define SIM_SOPT1_USBVSTBY_MASK       0x20000000u
#define SIM_SOPT1_USBVSTBY_SHIFT     29
#define SIM_SOPT1_USBVSTBY_WIDTH    1
#define SIM_SOPT1_USBVSTBY(x)        (((uint32_t)((uint32_t)(x))<<SIM_SOPT1_USBVSTBY_SHIFT))&SIM_SOPT1_USBVSTBY_MASK
#define SIM_SOPT1_USBSSTBY_MASK       0x40000000u
#define SIM_SOPT1_USBSSTBY_SHIFT     30
#define SIM_SOPT1_USBSSTBY_WIDTH    1
#define SIM_SOPT1_USBSSTBY(x)        (((uint32_t)((uint32_t)(x))<<SIM_SOPT1_USBSSTBY_SHIFT))&SIM_SOPT1_USBSSTBY_MASK
#define SIM_SOPT1_USBREGEN_MASK       0x80000000u
#define SIM_SOPT1_USBREGEN_SHIFT     31
#define SIM_SOPT1_USBREGEN_WIDTH    1
#define SIM_SOPT1_USBREGEN(x)        (((uint32_t)((uint32_t)(x))<<SIM_SOPT1_USBREGEN_SHIFT))&SIM_SOPT1_USBREGEN_MASK
/* SOPT1CFG Bit Fields */
#define SIM_SOPT1CFG_URWE_MASK        0x1000000u
#define SIM_SOPT1CFG_URWE_SHIFT      24
#define SIM_SOPT1CFG_URWE_WIDTH     1
#define SIM_SOPT1CFG_URWE(x)         (((uint32_t)((uint32_t)(x))<<SIM_SOPT1CFG_URWE_SHIFT))&SIM_SOPT1CFG_URWE_MASK
#define SIM_SOPT1CFG_UVSWE_MASK       0x2000000u

```

```

#define SIM_SOPT1CFG_UVSWE_SHIFT 25
#define SIM_SOPT1CFG_UVSWE_WIDTH 1
#define SIM_SOPT1CFG_UVSWE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT1CFG_UVSWE_SHIFT))&SIM_SOPT1CFG_UVSWE_MASK
#define SIM_SOPT1CFG_USSWE_MASK 0x4000000u
#define SIM_SOPT1CFG_USSWE_SHIFT 26
#define SIM_SOPT1CFG_USSWE_WIDTH 1
#define SIM_SOPT1CFG_USSWE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT1CFG_USSWE_SHIFT))&SIM_SOPT1CFG_USSWE_MASK
/* SOPT2 Bit Fields */
#define SIM_SOPT2_RTCCLKOUTSEL_MASK 0x10u
#define SIM_SOPT2_RTCCLKOUTSEL_SHIFT 4
#define SIM_SOPT2_RTCCLKOUTSEL_WIDTH 1
#define SIM_SOPT2_RTCCLKOUTSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_RTCCLKOUTSEL_SHIFT))&SIM_SOPT2_RTCCLKOUTSEL_MASK
#define SIM_SOPT2_CLKOUTSEL_MASK 0xE0u
#define SIM_SOPT2_CLKOUTSEL_SHIFT 5
#define SIM_SOPT2_CLKOUTSEL_WIDTH 3
#define SIM_SOPT2_CLKOUTSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_CLKOUTSEL_SHIFT))&SIM_SOPT2_CLKOUTSEL_MASK
#define SIM_SOPT2_PLLFLLSEL_MASK 0x10000u
#define SIM_SOPT2_PLLFLLSEL_SHIFT 16
#define SIM_SOPT2_PLLFLLSEL_WIDTH 1
#define SIM_SOPT2_PLLFLLSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_PLLFLLSEL_SHIFT))&SIM_SOPT2_PLLFLLSEL_MASK
#define SIM_SOPT2_USBSRC_MASK 0x40000u
#define SIM_SOPT2_USBSRC_SHIFT 18
#define SIM_SOPT2_USBSRC_WIDTH 1
#define SIM_SOPT2_USBSRC(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_USBSRC_SHIFT))&SIM_SOPT2_USBSRC_MASK
#define SIM_SOPT2_TPMSRC_MASK 0x3000000u
#define SIM_SOPT2_TPMSRC_SHIFT 24
#define SIM_SOPT2_TPMSRC_WIDTH 2
#define SIM_SOPT2_TPMSRC(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_TPMSRC_SHIFT))&SIM_SOPT2_TPMSRC_MASK
#define SIM_SOPT2_UART0SRC_MASK 0xC000000u
#define SIM_SOPT2_UART0SRC_SHIFT 26
#define SIM_SOPT2_UART0SRC_WIDTH 2
#define SIM_SOPT2_UART0SRC(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT2_UART0SRC_SHIFT))&SIM_SOPT2_UART0SRC_MASK
/* SOPT4 Bit Fields */
#define SIM_SOPT4 TPM1CH0SRC_MASK 0x40000u
#define SIM_SOPT4 TPM1CH0SRC_SHIFT 18
#define SIM_SOPT4 TPM1CH0SRC_WIDTH 1

```

```

#define SIM_SOPT4_TPM1CH0SRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT4_TPM1CH0SRC_SHIFT))&SIM_SOPT4_TPM1
CH0SRC_MASK)
#define SIM_SOPT4_TPM2CH0SRC_MASK 0x100000u
#define SIM_SOPT4_TPM2CH0SRC_SHIFT 20
#define SIM_SOPT4_TPM2CH0SRC_WIDTH 1
#define SIM_SOPT4_TPM2CH0SRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT4_TPM2CH0SRC_SHIFT))&SIM_SOPT4_TPM2
CH0SRC_MASK)
#define SIM_SOPT4_TPM0CLKSEL_MASK 0x1000000u
#define SIM_SOPT4_TPM0CLKSEL_SHIFT 24
#define SIM_SOPT4_TPM0CLKSEL_WIDTH 1
#define SIM_SOPT4_TPM0CLKSEL(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT4_TPM0CLKSEL_SHIFT))&SIM_SOPT4_TPM0
CLKSEL_MASK)
#define SIM_SOPT4_TPM1CLKSEL_MASK 0x2000000u
#define SIM_SOPT4_TPM1CLKSEL_SHIFT 25
#define SIM_SOPT4_TPM1CLKSEL_WIDTH 1
#define SIM_SOPT4_TPM1CLKSEL(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT4_TPM1CLKSEL_SHIFT))&SIM_SOPT4_TPM1
CLKSEL_MASK)
#define SIM_SOPT4_TPM2CLKSEL_MASK 0x4000000u
#define SIM_SOPT4_TPM2CLKSEL_SHIFT 26
#define SIM_SOPT4_TPM2CLKSEL_WIDTH 1
#define SIM_SOPT4_TPM2CLKSEL(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT4_TPM2CLKSEL_SHIFT))&SIM_SOPT4_TPM2
CLKSEL_MASK)
/* SOPT5 Bit Fields */
#define SIM_SOPT5_UART0TXSRC_MASK 0x3u
#define SIM_SOPT5_UART0TXSRC_SHIFT 0
#define SIM_SOPT5_UART0TXSRC_WIDTH 2
#define SIM_SOPT5_UART0TXSRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT5_UART0TXSRC_SHIFT))&SIM_SOPT5_UART
0TXSRC_MASK)
#define SIM_SOPT5_UART0RXSRC_MASK 0x4u
#define SIM_SOPT5_UART0RXSRC_SHIFT 2
#define SIM_SOPT5_UART0RXSRC_WIDTH 1
#define SIM_SOPT5_UART0RXSRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT5_UART0RXSRC_SHIFT))&SIM_SOPT5_UART
0RXSRC_MASK)
#define SIM_SOPT5_UART1TXSRC_MASK 0x30u
#define SIM_SOPT5_UART1TXSRC_SHIFT 4
#define SIM_SOPT5_UART1TXSRC_WIDTH 2
#define SIM_SOPT5_UART1TXSRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT5_UART1TXSRC_SHIFT))&SIM_SOPT5_UART
1TXSRC_MASK)
#define SIM_SOPT5_UART1RXSRC_MASK 0x40u
#define SIM_SOPT5_UART1RXSRC_SHIFT 6
#define SIM_SOPT5_UART1RXSRC_WIDTH 1
#define SIM_SOPT5_UART1RXSRC(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SOPT5_UART1RXSRC_SHIFT))&SIM_SOPT5_UART
1RXSRC_MASK)
#define SIM_SOPT5_UART0ODE_MASK 0x10000u
#define SIM_SOPT5_UART0ODE_SHIFT 16

```

```

#define SIM_SOPT5_UART0ODE_WIDTH 1
#define SIM_SOPT5_UART0ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART0ODE_SHIFT))&SIM_SOPT5_UART0ODE_MASK
#define SIM_SOPT5_UART1ODE_MASK 0x20000u
#define SIM_SOPT5_UART1ODE_SHIFT 17
#define SIM_SOPT5_UART1ODE_WIDTH 1
#define SIM_SOPT5_UART1ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART1ODE_SHIFT))&SIM_SOPT5_UART1ODE_MASK
#define SIM_SOPT5_UART2ODE_MASK 0x40000u
#define SIM_SOPT5_UART2ODE_SHIFT 18
#define SIM_SOPT5_UART2ODE_WIDTH 1
#define SIM_SOPT5_UART2ODE(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT5_UART2ODE_SHIFT))&SIM_SOPT5_UART2ODE_MASK
/* SOPT7 Bit Fields */
#define SIM_SOPT7_ADC0TRGSEL_MASK 0xFFu
#define SIM_SOPT7_ADC0TRGSEL_SHIFT 0
#define SIM_SOPT7_ADC0TRGSEL_WIDTH 4
#define SIM_SOPT7_ADC0TRGSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADC0TRGSEL_SHIFT))&SIM_SOPT7_ADC0TRGSEL_MASK
#define SIM_SOPT7_ADC0PRETRGSEL_MASK 0x10u
#define SIM_SOPT7_ADC0PRETRGSEL_SHIFT 4
#define SIM_SOPT7_ADC0PRETRGSEL_WIDTH 1
#define SIM_SOPT7_ADC0PRETRGSEL(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADC0PRETRGSEL_SHIFT))&SIM_SOPT7_ADC0PRETRGSEL_MASK
#define SIM_SOPT7_ADC0ALTTGEN_MASK 0x80u
#define SIM_SOPT7_ADC0ALTTGEN_SHIFT 7
#define SIM_SOPT7_ADC0ALTTGEN_WIDTH 1
#define SIM_SOPT7_ADC0ALTTGEN(x) (((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADC0ALTTGEN_SHIFT))&SIM_SOPT7_ADC0ALTTGEN_MASK
/* SDID Bit Fields */
#define SIM_SDID_PINID_MASK 0xFFu
#define SIM_SDID_PINID_SHIFT 0
#define SIM_SDID_PINID_WIDTH 4
#define SIM_SDID_PINID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_PINID_SHIFT))&SIM_SDID_PINID_MASK
#define SIM_SDID_DIEID_MASK 0xF80u
#define SIM_SDID_DIEID_SHIFT 7
#define SIM_SDID_DIEID_WIDTH 5
#define SIM_SDID_DIEID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_DIEID_SHIFT))&SIM_SDID_DIEID_MASK
#define SIM_SDID_REVID_MASK 0xF000u
#define SIM_SDID_REVID_SHIFT 12
#define SIM_SDID_REVID_WIDTH 4
#define SIM_SDID_REVID(x) (((uint32_t)((uint32_t)(x))<<SIM_SDID_REVID_SHIFT))&SIM_SDID_REVID_MASK
#define SIM_SDID_SRAMSIZE_MASK 0xF0000u
#define SIM_SDID_SRAMSIZE_SHIFT 16
#define SIM_SDID_SRAMSIZE_WIDTH 4

```

```

#define SIM_SDID_SRAMSIZE(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SDID_SRAMSIZE_SHIFT))&SIM_SDID_SRAMSIZE
_MASK)
#define SIM_SDID_SERIESID_MASK 0xF00000u
#define SIM_SDID_SERIESID_SHIFT 20
#define SIM_SDID_SERIESID_WIDTH 4
#define SIM_SDID_SERIESID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SDID_SERIESID_SHIFT))&SIM_SDID_SERIESID
_MASK)
#define SIM_SDID_SUBFAMID_MASK 0xF000000u
#define SIM_SDID_SUBFAMID_SHIFT 24
#define SIM_SDID_SUBFAMID_WIDTH 4
#define SIM_SDID_SUBFAMID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SDID_SUBFAMID_SHIFT))&SIM_SDID_SUBFAMID
_MASK)
#define SIM_SDID_FAMID_MASK 0xF0000000u
#define SIM_SDID_FAMID_SHIFT 28
#define SIM_SDID_FAMID_WIDTH 4
#define SIM_SDID_FAMID(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SDID_FAMID_SHIFT))&SIM_SDID_FAMID_MASK)
/* SCGC4 Bit Fields */
#define SIM_SCGC4_I2C0_MASK 0x40u
#define SIM_SCGC4_I2C0_SHIFT 6
#define SIM_SCGC4_I2C0_WIDTH 1
#define SIM_SCGC4_I2C0(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SCGC4_I2C0_SHIFT))&SIM_SCGC4_I2C0_MASK)
#define SIM_SCGC4_I2C1_MASK 0x80u
#define SIM_SCGC4_I2C1_SHIFT 7
#define SIM_SCGC4_I2C1_WIDTH 1
#define SIM_SCGC4_I2C1(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SCGC4_I2C1_SHIFT))&SIM_SCGC4_I2C1_MASK)
#define SIM_SCGC4_UART0_MASK 0x400u
#define SIM_SCGC4_UART0_SHIFT 10
#define SIM_SCGC4_UART0_WIDTH 1
#define SIM_SCGC4_UART0(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SCGC4_UART0_SHIFT))&SIM_SCGC4_UART0_MAS
K)
#define SIM_SCGC4_UART1_MASK 0x800u
#define SIM_SCGC4_UART1_SHIFT 11
#define SIM_SCGC4_UART1_WIDTH 1
#define SIM_SCGC4_UART1(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SCGC4_UART1_SHIFT))&SIM_SCGC4_UART1_MAS
K)
#define SIM_SCGC4_UART2_MASK 0x1000u
#define SIM_SCGC4_UART2_SHIFT 12
#define SIM_SCGC4_UART2_WIDTH 1
#define SIM_SCGC4_UART2(x)
(((uint32_t) (((uint32_t)(x))<<SIM_SCGC4_UART2_SHIFT))&SIM_SCGC4_UART2_MAS
K)
#define SIM_SCGC4_USBOTG_MASK 0x40000u
#define SIM_SCGC4_USBOTG_SHIFT 18
#define SIM_SCGC4_USBOTG_WIDTH 1

```

```

#define SIM_SCGC4_USBOTG(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC4_USBOTG_SHIFT)) & SIM_SCGC4_USBOTG_M
ASK)

#define SIM_SCGC4_CMP_MASK          0x80000u
#define SIM_SCGC4_CMP_SHIFT        19
#define SIM_SCGC4_CMP_WIDTH         1
#define SIM_SCGC4_CMP(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC4_CMP_SHIFT)) & SIM_SCGC4_CMP_MASK)

#define SIM_SCGC4_SPI0_MASK         0x400000u
#define SIM_SCGC4_SPI0_SHIFT        22
#define SIM_SCGC4_SPI0_WIDTH         1
#define SIM_SCGC4_SPI0(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC4_SPI0_SHIFT)) & SIM_SCGC4_SPI0_MASK)

#define SIM_SCGC4_SPI1_MASK         0x800000u
#define SIM_SCGC4_SPI1_SHIFT        23
#define SIM_SCGC4_SPI1_WIDTH         1
#define SIM_SCGC4_SPI1(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC4_SPI1_SHIFT)) & SIM_SCGC4_SPI1_MASK)

/* SCGC5 Bit Fields */

#define SIM_SCGC5_LPTMR_MASK          0x1u
#define SIM_SCGC5_LPTMR_SHIFT        0
#define SIM_SCGC5_LPTMR_WIDTH         1
#define SIM_SCGC5_LPTMR(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_LPTMR_SHIFT)) & SIM_SCGC5_LPTMR_MAS
K)

#define SIM_SCGC5_TSI_MASK           0x20u
#define SIM_SCGC5_TSI_SHIFT          5
#define SIM_SCGC5_TSI_WIDTH           1
#define SIM_SCGC5_TSI(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_TSI_SHIFT)) & SIM_SCGC5_TSI_MASK)

#define SIM_SCGC5_PORTA_MASK          0x200u
#define SIM_SCGC5_PORTA_SHIFT         9
#define SIM_SCGC5_PORTA_WIDTH          1
#define SIM_SCGC5_PORTA(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_PORTA_SHIFT)) & SIM_SCGC5_PORTA_MAS
K)

#define SIM_SCGC5_PORTB_MASK          0x400u
#define SIM_SCGC5_PORTB_SHIFT         10
#define SIM_SCGC5_PORTB_WIDTH          1
#define SIM_SCGC5_PORTB(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_PORTB_SHIFT)) & SIM_SCGC5_PORTB_MAS
K)

#define SIM_SCGC5_PORTC_MASK          0x800u
#define SIM_SCGC5_PORTC_SHIFT         11
#define SIM_SCGC5_PORTC_WIDTH          1
#define SIM_SCGC5_PORTC(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_PORTC_SHIFT)) & SIM_SCGC5_PORTC_MAS
K)

#define SIM_SCGC5_PORTD_MASK          0x1000u
#define SIM_SCGC5_PORTD_SHIFT         12
#define SIM_SCGC5_PORTD_WIDTH          1
#define SIM_SCGC5_PORTD(x)
(((uint32_t) (((uint32_t) (x)) << SIM_SCGC5_PORTD_SHIFT)) & SIM_SCGC5_PORTD_MAS
K)

```

```

#define SIM_SCGC5_PORTE_MASK          0x2000u
#define SIM_SCGC5_PORTE_SHIFT         13
#define SIM_SCGC5_PORTE_WIDTH         1
#define SIM_SCGC5_PORTE(x)           (((uint32_t)((uint32_t)(x))<<SIM_SCGC5_PORTE_SHIFT))&SIM_SCGC5_PORTE_MASK
/* SCGC6 Bit Fields */
#define SIM_SCGC6_FTF_MASK            0x1u
#define SIM_SCGC6_FTF_SHIFT           0
#define SIM_SCGC6_FTF_WIDTH           1
#define SIM_SCGC6_FTF(x)             (((uint32_t)((uint32_t)(x))<<SIM_SCGC6_FTF_SHIFT))&SIM_SCGC6_FTF_MASK
#define SIM_SCGC6_DMAMUX_MASK         0x2u
#define SIM_SCGC6_DMAMUX_SHIFT        1
#define SIM_SCGC6_DMAMUX_WIDTH        1
#define SIM_SCGC6_DMAMUX(x)          (((uint32_t)((uint32_t)(x))<<SIM_SCGC6_DMAMUX_SHIFT))&SIM_SCGC6_DMAMUX_MASK
#define SIM_SCGC6_PIT_MASK            0x800000u
#define SIM_SCGC6_PIT_SHIFT           23
#define SIM_SCGC6_PIT_WIDTH           1
#define SIM_SCGC6_PIT(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6_PIT_SHIFT))&SIM_SCGC6_PIT_MASK
#define SIM_SCGC6 TPM0_MASK           0x1000000u
#define SIM_SCGC6 TPM0_SHIFT           24
#define SIM_SCGC6 TPM0_WIDTH           1
#define SIM_SCGC6 TPM0(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 TPM0_SHIFT))&SIM_SCGC6 TPM0_MASK
#define SIM_SCGC6 TPM1_MASK           0x2000000u
#define SIM_SCGC6 TPM1_SHIFT           25
#define SIM_SCGC6 TPM1_WIDTH           1
#define SIM_SCGC6 TPM1(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 TPM1_SHIFT))&SIM_SCGC6 TPM1_MASK
#define SIM_SCGC6 TPM2_MASK           0x4000000u
#define SIM_SCGC6 TPM2_SHIFT           26
#define SIM_SCGC6 TPM2_WIDTH           1
#define SIM_SCGC6 TPM2(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 TPM2_SHIFT))&SIM_SCGC6 TPM2_MASK
#define SIM_SCGC6 ADC0_MASK            0x800000u
#define SIM_SCGC6 ADC0_SHIFT           27
#define SIM_SCGC6 ADC0_WIDTH           1
#define SIM_SCGC6 ADC0(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 ADC0_SHIFT))&SIM_SCGC6 ADC0_MASK
#define SIM_SCGC6 RTC_MASK             0x20000000u
#define SIM_SCGC6 RTC_SHIFT            29
#define SIM_SCGC6 RTC_WIDTH            1
#define SIM_SCGC6 RTC(x)               (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 RTC_SHIFT))&SIM_SCGC6 RTC_MASK
#define SIM_SCGC6 DAC0_MASK            0x80000000u
#define SIM_SCGC6 DAC0_SHIFT           31
#define SIM_SCGC6 DAC0_WIDTH           1
#define SIM_SCGC6 DAC0(x)              (((uint32_t)((uint32_t)(x))<<SIM_SCGC6 DAC0_SHIFT))&SIM_SCGC6 DAC0_MASK
/* SCGC7 Bit Fields */

```

```

#define SIM_SCGC7_DMA_MASK 0x100u
#define SIM_SCGC7_DMA_SHIFT 8
#define SIM_SCGC7_DMA_WIDTH 1
#define SIM_SCGC7_DMA(x) (((uint32_t)((uint32_t)(x))<<SIM_SCGC7_DMA_SHIFT))&SIM_SCGC7_DMA_MASK
/* CLKDIV1 Bit Fields */
#define SIM_CLKDIV1_OUTDIV4_MASK 0x70000u
#define SIM_CLKDIV1_OUTDIV4_SHIFT 16
#define SIM_CLKDIV1_OUTDIV4_WIDTH 3
#define SIM_CLKDIV1_OUTDIV4(x) (((uint32_t)((uint32_t)(x))<<SIM_CLKDIV1_OUTDIV4_SHIFT))&SIM_CLKDIV1_OUTDIV4_MASK
#define SIM_CLKDIV1_OUTDIV1_MASK 0xF0000000u
#define SIM_CLKDIV1_OUTDIV1_SHIFT 28
#define SIM_CLKDIV1_OUTDIV1_WIDTH 4
#define SIM_CLKDIV1_OUTDIV1(x) (((uint32_t)((uint32_t)(x))<<SIM_CLKDIV1_OUTDIV1_SHIFT))&SIM_CLKDIV1_OUTDIV1_MASK
/* FCFG1 Bit Fields */
#define SIM_FCFG1_FLASHDIS_MASK 0x1u
#define SIM_FCFG1_FLASHDIS_SHIFT 0
#define SIM_FCFG1_FLASHDIS_WIDTH 1
#define SIM_FCFG1_FLASHDIS(x) (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDIS_SHIFT))&SIM_FCFG1_FLASHDIS_MASK
#define SIM_FCFG1_FLASHDOZE_MASK 0x2u
#define SIM_FCFG1_FLASHDOZE_SHIFT 1
#define SIM_FCFG1_FLASHDOZE_WIDTH 1
#define SIM_FCFG1_FLASHDOZE(x) (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_FLASHDOZE_SHIFT))&SIM_FCFG1_FLASHDOZE_MASK
#define SIM_FCFG1_PFSIZE_MASK 0xF000000u
#define SIM_FCFG1_PFSIZE_SHIFT 24
#define SIM_FCFG1_PFSIZE_WIDTH 4
#define SIM_FCFG1_PFSIZE(x) (((uint32_t)((uint32_t)(x))<<SIM_FCFG1_PFSIZE_SHIFT))&SIM_FCFG1_PFSIZE_MASK
/* FCFG2 Bit Fields */
#define SIM_FCFG2_MAXADDR0_MASK 0x7F000000u
#define SIM_FCFG2_MAXADDR0_SHIFT 24
#define SIM_FCFG2_MAXADDR0_WIDTH 7
#define SIM_FCFG2_MAXADDR0(x) (((uint32_t)((uint32_t)(x))<<SIM_FCFG2_MAXADDR0_SHIFT))&SIM_FCFG2_MAXADDR0_MASK
/* UIDMH Bit Fields */
#define SIM_UIDMH_UID_MASK 0xFFFFu
#define SIM_UIDMH_UID_SHIFT 0
#define SIM_UIDMH_UID_WIDTH 16
#define SIM_UIDMH_UID(x) (((uint32_t)((uint32_t)(x))<<SIM_UIDMH_UID_SHIFT))&SIM_UIDMH_UID_MASK
/* UIDML Bit Fields */
#define SIM_UIDML_UID_MASK 0xFFFFFFFFu
#define SIM_UIDML_UID_SHIFT 0
#define SIM_UIDML_UID_WIDTH 32

```

```

#define SIM_UIDML_UID(x)
(((uint32_t)((uint32_t)(x))<<SIM_UIDML_UID_SHIFT))&SIM_UIDML_UID_MASK)
/* UIDL Bit Fields */
#define SIM_UIDL_UID_MASK          0xFFFFFFFFu
#define SIM_UIDL_UID_SHIFT         0
#define SIM_UIDL_UID_WIDTH         32
#define SIM_UIDL_UID(x)
(((uint32_t)((uint32_t)(x))<<SIM_UIDL_UID_SHIFT))&SIM_UIDL_UID_MASK)
/* COPC Bit Fields */
#define SIM_COPC_COPW_MASK         0x1u
#define SIM_COPC_COPW_SHIFT        0
#define SIM_COPC_COPW_WIDTH        1
#define SIM_COPC_COPW(x)
(((uint32_t)((uint32_t)(x))<<SIM_COPC_COPW_SHIFT))&SIM_COPC_COPW_MASK)
#define SIM_COPC_COPCLKS_MASK      0x2u
#define SIM_COPC_COPCLKS_SHIFT     1
#define SIM_COPC_COPCLKS_WIDTH     1
#define SIM_COPC_COPCLKS(x)
(((uint32_t)((uint32_t)(x))<<SIM_COPC_COPCLKS_SHIFT))&SIM_COPC_COPCLKS_MASK)
#define SIM_COPC_COPT_MASK         0xCu
#define SIM_COPC_COPT_SHIFT        2
#define SIM_COPC_COPT_WIDTH        2
#define SIM_COPC_COPT(x)
(((uint32_t)((uint32_t)(x))<<SIM_COPC_COPT_SHIFT))&SIM_COPC_COPT_MASK)
/* SRVCOP Bit Fields */
#define SIM_SRVCOP_SRVCOP_MASK     0xFFu
#define SIM_SRVCOP_SRVCOP_SHIFT    0
#define SIM_SRVCOP_SRVCOP_WIDTH    8
#define SIM_SRVCOP_SRVCOP(x)
(((uint32_t)((uint32_t)(x))<<SIM_SRVCOP_SRVCOP_SHIFT))&SIM_SRVCOP_SRVCOP_MASK)

/*!
 * @}
 */ /* end of group SIM_Register_Masks */


```

```

/* SIM - Peripheral instance base addresses */
/** Peripheral SIM base address */
#define SIM_BASE                  (0x40047000u)
/** Peripheral SIM base pointer */
#define SIM                         ((SIM_Type *)SIM_BASE)
#define SIM_BASE_PTR                (SIM)
/** Array initializer of SIM peripheral base addresses */
#define SIM_BASE_ADDRS             { SIM_BASE }
/** Array initializer of SIM peripheral base pointers */
#define SIM_BASE_PTRS               { SIM }

/* -----
-- SIM - Register accessor macros
-----
*/

```

```

/*!
 * @addtogroup SIM_Register_Accessor_Macros SIM - Register accessor
macros
 * @{
 */

/* SIM - Register instance definitions */
/* SIM */
#define SIM_SOPT1 SIM_SOPT1_REG(SIM)
#define SIM_SOPT1CFG SIM_SOPT1CFG_REG(SIM)
#define SIM_SOPT2 SIM_SOPT2_REG(SIM)
#define SIM_SOPT4 SIM_SOPT4_REG(SIM)
#define SIM_SOPT5 SIM_SOPT5_REG(SIM)
#define SIM_SOPT7 SIM_SOPT7_REG(SIM)
#define SIM_SDID SIM_SDID_REG(SIM)
#define SIM_SCGC4 SIM_SCGC4_REG(SIM)
#define SIM_SCGC5 SIM_SCGC5_REG(SIM)
#define SIM_SCGC6 SIM_SCGC6_REG(SIM)
#define SIM_SCGC7 SIM_SCGC7_REG(SIM)
#define SIM_CLKDIV1 SIM_CLKDIV1_REG(SIM)
#define SIM_FCFG1 SIM_FCFG1_REG(SIM)
#define SIM_FCFG2 SIM_FCFG2_REG(SIM)
#define SIM_UIDMH SIM_UIDMH_REG(SIM)
#define SIM_UIDML SIM_UIDML_REG(SIM)
#define SIM_UIDL SIM_UIDL_REG(SIM)
#define SIM_COPC SIM_COPC_REG(SIM)
#define SIM_SRVCOP SIM_SRVCOP_REG(SIM)

/*!
 * @}
 */
/* end of group SIM_Register_Accessor_Macros */

/*!
 * @}
 */
/* end of group SIM_Peripheral_Access_Layer */

/*
-----
-- SMC Peripheral Access Layer
-----
*/
/*!
 * @addtogroup SMC_Peripheral_Access_Layer SMC Peripheral Access Layer
 * @{
 */
/* ** SMC - Register Layout Typedef */
typedef struct {


```

```

    __IO uint8_t PMPROT;                                /**< Power Mode
Protection register, offset: 0x0 */
    __IO uint8_t PMCTRL;                               /**< Power Mode
Control register, offset: 0x1 */
    __IO uint8_t STOPCTRL;                            /**< Stop Control
Register, offset: 0x2 */
    __I  uint8_t PMSTAT;                             /**< Power Mode Status
register, offset: 0x3 */
} SMC_Type, *SMC_MemMapPtr;

/* -----
-- SMC - Register accessor macros
-----
*/
/*!
* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor
macros
* @{
*/
/* SMC - Register accessors */
#define SMC_PMPROT_REG(base)          ((base) -> PMPROT)
#define SMC_PMCTRL_REG(base)          ((base) -> PMCTRL)
#define SMC_STOPCTRL_REG(base)        ((base) -> STOPCTRL)
#define SMC_PMSTAT_REG(base)          ((base) -> PMSTAT)

/*!
* @}
*/
/* end of group SMC_Register_Accessor_Macros */

/* -----
-- SMC Register Masks
-----
*/
/*!
* @addtogroup SMC_Register_Masks SMC Register Masks
* @{
*/
/* PMPROT Bit Fields */
#define SMC_PMPROT_AVLLS_MASK          0x2u
#define SMC_PMPROT_AVLLS_SHIFT         1
#define SMC_PMPROT_AVLLS_WIDTH         1
#define SMC_PMPROT_AVLLS(x)           (((uint8_t)((uint8_t)(x)) << SMC_PMPROT_AVLLS_SHIFT)) & SMC_PMPROT_AVLLS_MASK
#define SMC_PMPROT_ALLS_MASK           0x8u
#define SMC_PMPROT_ALLS_SHIFT          3

```

```

#define SMC_PMPROT_ALLS_WIDTH 1
#define SMC_PMPROT_ALLS(x) (((uint8_t)((uint8_t)(x))<<SMC_PMPROT_ALLS_SHIFT))&SMC_PMPROT_ALLS_MASK)
#define SMC_PMPROT_AVLP_MASK 0x20u
#define SMC_PMPROT_AVLP_SHIFT 5
#define SMC_PMPROT_AVLP_WIDTH 1
#define SMC_PMPROT_AVLP(x) (((uint8_t)((uint8_t)(x))<<SMC_PMPROT_AVLP_SHIFT))&SMC_PMPROT_AVLP_MASK)
/* PMCTRL Bit Fields */
#define SMC_PMCTRL_STOPM_MASK 0x7u
#define SMC_PMCTRL_STOPM_SHIFT 0
#define SMC_PMCTRL_STOPM_WIDTH 3
#define SMC_PMCTRL_STOPM(x) (((uint8_t)((uint8_t)(x))<<SMC_PMCTRL_STOPM_SHIFT))&SMC_PMCTRL_STOPM_MASK)
#define SMC_PMCTRL_STOPA_MASK 0x8u
#define SMC_PMCTRL_STOPA_SHIFT 3
#define SMC_PMCTRL_STOPA_WIDTH 1
#define SMC_PMCTRL_STOPA(x) (((uint8_t)((uint8_t)(x))<<SMC_PMCTRL_STOPA_SHIFT))&SMC_PMCTRL_STOPA_MASK)
#define SMC_PMCTRL_RUNM_MASK 0x60u
#define SMC_PMCTRL_RUNM_SHIFT 5
#define SMC_PMCTRL_RUNM_WIDTH 2
#define SMC_PMCTRL_RUNM(x) (((uint8_t)((uint8_t)(x))<<SMC_PMCTRL_RUNM_SHIFT))&SMC_PMCTRL_RUNM_MASK)
/* STOPCTRL Bit Fields */
#define SMC_STOPCTRL_VLLSM_MASK 0x7u
#define SMC_STOPCTRL_VLLSM_SHIFT 0
#define SMC_STOPCTRL_VLLSM_WIDTH 3
#define SMC_STOPCTRL_VLLSM(x) (((uint8_t)((uint8_t)(x))<<SMC_STOPCTRL_VLLSM_SHIFT))&SMC_STOPCTRL_VLLSM_MASK)
#define SMC_STOPCTRL_PORPO_MASK 0x20u
#define SMC_STOPCTRL_PORPO_SHIFT 5
#define SMC_STOPCTRL_PORPO_WIDTH 1
#define SMC_STOPCTRL_PORPO(x) (((uint8_t)((uint8_t)(x))<<SMC_STOPCTRL_PORPO_SHIFT))&SMC_STOPCTRL_PORPO_MASK)
#define SMC_STOPCTRL_PSTOPO_MASK 0xC0u
#define SMC_STOPCTRL_PSTOPO_SHIFT 6
#define SMC_STOPCTRL_PSTOPO_WIDTH 2
#define SMC_STOPCTRL_PSTOPO(x) (((uint8_t)((uint8_t)(x))<<SMC_STOPCTRL_PSTOPO_SHIFT))&SMC_STOPCTRL_PSTOPO_MASK)
/* PMSTAT Bit Fields */
#define SMC_PMSTAT_PMSTAT_MASK 0x7Fu
#define SMC_PMSTAT_PMSTAT_SHIFT 0
#define SMC_PMSTAT_PMSTAT_WIDTH 7
#define SMC_PMSTAT_PMSTAT(x) (((uint8_t)((uint8_t)(x))<<SMC_PMSTAT_PMSTAT_SHIFT))&SMC_PMSTAT_PMSTAT_MASK)

```

/\* !

```

* @}
/*/* end of group SMC_Register_Masks */

/* SMC - Peripheral instance base addresses */
/** Peripheral SMC base address */
#define SMC_BASE (0x4007E000u)
/** Peripheral SMC base pointer */
#define SMC ((SMC_Type *) SMC_BASE)
#define SMC_BASE_PTR ((SMC) SMC)
/** Array initializer of SMC peripheral base addresses */
#define SMC_BASE_ADDRS { SMC_BASE }
/** Array initializer of SMC peripheral base pointers */
#define SMC_BASE_PTRS { SMC }

/* -----
-- SMC - Register accessor macros
-----
*/
/*!
* @addtogroup SMC_Register_Accessor_Macros SMC - Register accessor
macros
* @{
*/
/* SMC - Register instance definitions */
/* SMC */
#define SMC_PMPROT SMC_PMPROT_REG(SMC)
#define SMC_PMCTRL SMC_PMCTRL_REG(SMC)
#define SMC_STOPCTRL SMC_STOPCTRL_REG(SMC)
#define SMC_PMSTAT SMC_PMSTAT_REG(SMC)

/*!
* @}
*/
/*/* end of group SMC_Register_Accessor_Macros */

/*!
* @}
*/
/*/* end of group SMC_Peripheral_Access_Layer */

/* -----
-- SPI Peripheral Access Layer
-----
*/
/*!
* @addtogroup SPI_Peripheral_Access_Layer SPI Peripheral Access Layer
* @{
*/

```

```

        */

/** SPI - Register Layout Typedef */
typedef struct {
    __IO uint8_t C1;                                /**< SPI control
register 1, offset: 0x0 */
    __IO uint8_t C2;                                /**< SPI control
register 2, offset: 0x1 */
    __IO uint8_t BR;                               /**< SPI baud rate
register, offset: 0x2 */
    __IO uint8_t S;                                 /**< SPI status
register, offset: 0x3 */
    uint8_t RESERVED_0[1];
    __IO uint8_t D;                                /**< SPI data
register, offset: 0x5 */
    uint8_t RESERVED_1[1];
    __IO uint8_t M;                               /**< SPI match
register, offset: 0x7 */
} SPI_Type, *SPI_MemMapPtr;

/* -----
-- SPI - Register accessor macros
-----
*/
/*
 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor
macros
 * @{
 */

/* SPI - Register accessors */
#define SPI_C1_REG(base)          ((base)->C1)
#define SPI_C2_REG(base)          ((base)->C2)
#define SPI_BR_REG(base)          ((base)->BR)
#define SPI_S_REG(base)           ((base)->S)
#define SPI_D_REG(base)           ((base)->D)
#define SPI_M_REG(base)           ((base)->M)

/*!
 * @}
 */
/* end of group SPI_Register_Accessor_Macros */

/* -----
-- SPI Register Masks
-----
*/
/*
 * @addtogroup SPI_Register_Masks SPI Register Masks
*/

```

```

/* @{
 */

/* C1 Bit Fields */
#define SPI_C1_LSBFE_MASK          0x1u
#define SPI_C1_LSBFE_SHIFT         0
#define SPI_C1_LSBFE_WIDTH         1
#define SPI_C1_LSBFE(x) (((uint8_t)(x))<<SPI_C1_LSBFE_SHIFT)&SPI_C1_LSBFE_MASK)
#define SPI_C1_SSOE_MASK           0x2u
#define SPI_C1_SSOE_SHIFT          1
#define SPI_C1_SSOE_WIDTH          1
#define SPI_C1_SSOE(x) (((uint8_t)(x))<<SPI_C1_SSOE_SHIFT)&SPI_C1_SSOE_MASK)
#define SPI_C1_CPHA_MASK           0x4u
#define SPI_C1_CPHA_SHIFT          2
#define SPI_C1_CPHA_WIDTH          1
#define SPI_C1_CPHA(x) (((uint8_t)(x))<<SPI_C1_CPHA_SHIFT)&SPI_C1_CPHA_MASK)
#define SPI_C1_CPOL_MASK           0x8u
#define SPI_C1_CPOL_SHIFT          3
#define SPI_C1_CPOL_WIDTH          1
#define SPI_C1_CPOL(x) (((uint8_t)(x))<<SPI_C1_CPOL_SHIFT)&SPI_C1_CPOL_MASK)
#define SPI_C1_MSTR_MASK           0x10u
#define SPI_C1_MSTR_SHIFT          4
#define SPI_C1_MSTR_WIDTH          1
#define SPI_C1_MSTR(x) (((uint8_t)(x))<<SPI_C1_MSTR_SHIFT)&SPI_C1_MSTR_MASK)
#define SPI_C1_SPTIE_MASK          0x20u
#define SPI_C1_SPTIE_SHIFT          5
#define SPI_C1_SPTIE_WIDTH          1
#define SPI_C1_SPTIE(x) (((uint8_t)(x))<<SPI_C1_SPTIE_SHIFT)&SPI_C1_SPTIE_MASK)
#define SPI_C1_SPE_MASK             0x40u
#define SPI_C1_SPE_SHIFT              6
#define SPI_C1_SPE_WIDTH              1
#define SPI_C1_SPE(x) (((uint8_t)(x))<<SPI_C1_SPE_SHIFT)&SPI_C1_SPE_MASK)
#define SPI_C1_SPIE_MASK             0x80u
#define SPI_C1_SPIE_SHIFT              7
#define SPI_C1_SPIE_WIDTH              1
#define SPI_C1_SPIE(x) (((uint8_t)(x))<<SPI_C1_SPIE_SHIFT)&SPI_C1_SPIE_MASK)

/* C2 Bit Fields */
#define SPI_C2_SPC0_MASK           0x1u
#define SPI_C2_SPC0_SHIFT          0
#define SPI_C2_SPC0_WIDTH          1
#define SPI_C2_SPC0(x) (((uint8_t)(x))<<SPI_C2_SPC0_SHIFT)&SPI_C2_SPC0_MASK)
#define SPI_C2_SPISWAI_MASK          0x2u
#define SPI_C2_SPISWAI_SHIFT         1
#define SPI_C2_SPISWAI_WIDTH         1

```

```

#define SPI_C2_SPISWAI(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_SPISWAI_SHIFT))&SPI_C2_SPISWAI_MASK)
#define SPI_C2_RXDMAE_MASK 0x4u
#define SPI_C2_RXDMAE_SHIFT 2
#define SPI_C2_RXDMAE_WIDTH 1
#define SPI_C2_RXDMAE(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_RXDMAE_SHIFT))&SPI_C2_RXDMAE_MASK)
#define SPI_C2_BIDIROE_MASK 0x8u
#define SPI_C2_BIDIROE_SHIFT 3
#define SPI_C2_BIDIROE_WIDTH 1
#define SPI_C2_BIDIROE(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_BIDIROE_SHIFT))&SPI_C2_BIDIROE_MASK)
#define SPI_C2_MODFEN_MASK 0x10u
#define SPI_C2_MODFEN_SHIFT 4
#define SPI_C2_MODFEN_WIDTH 1
#define SPI_C2_MODFEN(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_MODFEN_SHIFT))&SPI_C2_MODFEN_MASK)
#define SPI_C2_TXDMAE_MASK 0x20u
#define SPI_C2_TXDMAE_SHIFT 5
#define SPI_C2_TXDMAE_WIDTH 1
#define SPI_C2_TXDMAE(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_TXDMAE_SHIFT))&SPI_C2_TXDMAE_MASK)
#define SPI_C2_SPMIE_MASK 0x80u
#define SPI_C2_SPMIE_SHIFT 7
#define SPI_C2_SPMIE_WIDTH 1
#define SPI_C2_SPMIE(x)
(((uint8_t)((uint8_t)(x))<<SPI_C2_SPMIE_SHIFT))&SPI_C2_SPMIE_MASK)
/* BR Bit Fields */
#define SPI_BR_SPR_MASK 0xFu
#define SPI_BR_SPR_SHIFT 0
#define SPI_BR_SPR_WIDTH 4
#define SPI_BR_SPR(x)
(((uint8_t)((uint8_t)(x))<<SPI_BR_SPR_SHIFT))&SPI_BR_SPR_MASK)
#define SPI_BR_SPPR_MASK 0x70u
#define SPI_BR_SPPR_SHIFT 4
#define SPI_BR_SPPR_WIDTH 3
#define SPI_BR_SPPR(x)
(((uint8_t)((uint8_t)(x))<<SPI_BR_SPPR_SHIFT))&SPI_BR_SPPR_MASK)
/* S Bit Fields */
#define SPI_S_MODF_MASK 0x10u
#define SPI_S_MODF_SHIFT 4
#define SPI_S_MODF_WIDTH 1
#define SPI_S_MODF(x)
(((uint8_t)((uint8_t)(x))<<SPI_S_MODF_SHIFT))&SPI_S_MODF_MASK)
#define SPI_S_SPTEF_MASK 0x20u
#define SPI_S_SPTEF_SHIFT 5
#define SPI_S_SPTEF_WIDTH 1
#define SPI_S_SPTEF(x)
(((uint8_t)((uint8_t)(x))<<SPI_S_SPTEF_SHIFT))&SPI_S_SPTEF_MASK)
#define SPI_S_SPMF_MASK 0x40u
#define SPI_S_SPMF_SHIFT 6
#define SPI_S_SPMF_WIDTH 1
#define SPI_S_SPMF(x)
(((uint8_t)((uint8_t)(x))<<SPI_S_SPMF_SHIFT))&SPI_S_SPMF_MASK)

```

```

#define SPI_S_SPRF_MASK          0x80u
#define SPI_S_SPRF_SHIFT         7
#define SPI_S_SPRF_WIDTH         1
#define SPI_S_SPRF(x) (((uint8_t)((uint8_t)(x))<<SPI_S_SPRF_SHIFT))&SPI_S_SPRF_MASK)
/* D Bit Fields */
#define SPI_D_Bits_MASK          0xFFu
#define SPI_D_Bits_SHIFT         0
#define SPI_D_Bits_WIDTH         8
#define SPI_D_Bits(x) (((uint8_t)((uint8_t)(x))<<SPI_D_Bits_SHIFT))&SPI_D_Bits_MASK)
/* M Bit Fields */
#define SPI_M_Bits_MASK          0xFFu
#define SPI_M_Bits_SHIFT         0
#define SPI_M_Bits_WIDTH         8
#define SPI_M_Bits(x) (((uint8_t)((uint8_t)(x))<<SPI_M_Bits_SHIFT))&SPI_M_Bits_MASK)

/*!
 * @}
 */ /* end of group SPI_Register_Masks */

/* SPI - Peripheral instance base addresses */
/** Peripheral SPI0 base address */
#define SPI0_BASE                (0x40076000u)
/** Peripheral SPI0 base pointer */
#define SPI0                      ((SPI_Type *)SPI0_BASE)
#define SPI0_PTR                  (SPI0)
/** Peripheral SPI1 base address */
#define SPI1_BASE                (0x40077000u)
/** Peripheral SPI1 base pointer */
#define SPI1                      ((SPI_Type *)SPI1_BASE)
#define SPI1_PTR                  (SPI1)
/** Array initializer of SPI peripheral base addresses */
#define SPI_BASE_ADDRS           { SPI0_BASE, SPI1_BASE }
/** Array initializer of SPI peripheral base pointers */
#define SPI_BASE_PTRS             { SPI0, SPI1 }
/** Interrupt vectors for the SPI peripheral type */
#define SPI IRQS                  { SPI0_IRQn, SPI1_IRQn }

/* -----
-- SPI - Register accessor macros
-----
*/
/*!
 * @addtogroup SPI_Register_Accessor_Macros SPI - Register accessor
macros
 * @{
 */

```

```

/* SPI - Register instance definitions */
/* SPI0 */
#define SPI0_C1                      SPI_C1_REG(SPI0)
#define SPI0_C2                      SPI_C2_REG(SPI0)
#define SPI0_BR                      SPI_BR_REG(SPI0)
#define SPI0_S                       SPI_S_REG(SPI0)
#define SPI0_D                       SPI_D_REG(SPI0)
#define SPI0_M                       SPI_M_REG(SPI0)
/* SPI1 */
#define SPI1_C1                      SPI_C1_REG(SPI1)
#define SPI1_C2                      SPI_C2_REG(SPI1)
#define SPI1_BR                      SPI_BR_REG(SPI1)
#define SPI1_S                       SPI_S_REG(SPI1)
#define SPI1_D                       SPI_D_REG(SPI1)
#define SPI1_M                       SPI_M_REG(SPI1)

/*!
 * @}
 */ /* end of group SPI_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group SPI_Peripheral_Access_Layer */

/*
-----  

-- TPM Peripheral Access Layer  

-----  

----- */

/*!  

 * @addtogroup TPM_Peripheral_Access_Layer TPM Peripheral Access Layer  

 * @{
 */

/** TPM - Register Layout Typedef */
typedef struct {
    __IO uint32_t SC;                                /**< Status and
Control, offset: 0x0 */  

    __IO uint32_t CNT;                               /**< Counter, offset:
0x4 */  

    __IO uint32_t MOD;                               /**< Modulo, offset:
0x8 */  

    struct {  

        __IO uint32_t CnSC;                          /**< Channel (n)  

Status and Control, array offset: 0xC, array step: 0x8 */  

        __IO uint32_t CnV;                           /**< Channel (n)  

Value, array offset: 0x10, array step: 0x8 */  

    } CONTROLS[6];
    uint8_t RESERVED_0[20];
}

```

```

    __IO uint32_t STATUS;                                /**< Capture and
Compare_Status, offset: 0x50 */
    uint8_t RESERVED_1[48];
    __IO uint32_t CONF;                                /**< Configuration,
offset: 0x84 */
} TPM_Type, *TPM_MemMapPtr;

/* -----
-- TPM - Register accessor macros
-----
*/
/*!
* @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor
macros
* @{
*/
/* TPM - Register accessors */
#define TPM_SC_REG(base)                                ((base)->SC)
#define TPM_CNT_REG(base)                               ((base)->CNT)
#define TPM_MOD_REG(base)                               ((base)->MOD)
#define TPM_CnSC_REG(base, index)                      ((base)-
>CONTROLS[index].CnSC)
#define TPM_CnSC_COUNT                                6
#define TPM_CnV_REG(base, index)                      ((base)-
>CONTROLS[index].CnV)
#define TPM_CnV_COUNT                                 6
#define TPM_STATUS_REG(base)                           ((base)->STATUS)
#define TPM_CONF_REG(base)                            ((base)->CONF)

/*!
* @}
*/
/* end of group TPM_Register_Accessor_Macros */

/* -----
-- TPM Register Masks
-----
*/
/*!
* @addtogroup TPM_Register_Masks TPM Register Masks
* @{
*/
/* SC Bit Fields */
#define TPM_SC_PS_MASK                                0x7u
#define TPM_SC_PS_SHIFT                               0
#define TPM_SC_PS_WIDTH                              3

```

```

#define TPM_SC_PS(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_PS_SHIFT))&TPM_SC_PS_MASK)
#define TPM_SC_CMOD_MASK 0x18u
#define TPM_SC_CMOD_SHIFT 3
#define TPM_SC_CMOD_WIDTH 2
#define TPM_SC_CMOD(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_CMOD_SHIFT))&TPM_SC_CMOD_MASK)
#define TPM_SC_CPWMS_MASK 0x20u
#define TPM_SC_CPWMS_SHIFT 5
#define TPM_SC_CPWMS_WIDTH 1
#define TPM_SC_CPWMS(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_CPWMS_SHIFT))&TPM_SC_CPWMS_MASK)
#define TPM_SC_TOIE_MASK 0x40u
#define TPM_SC_TOIE_SHIFT 6
#define TPM_SC_TOIE_WIDTH 1
#define TPM_SC_TOIE(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_TOIE_SHIFT))&TPM_SC_TOIE_MASK)
#define TPM_SC_TOF_MASK 0x80u
#define TPM_SC_TOF_SHIFT 7
#define TPM_SC_TOF_WIDTH 1
#define TPM_SC_TOF(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_TOF_SHIFT))&TPM_SC_TOF_MASK)
#define TPM_SC_DMA_MASK 0x100u
#define TPM_SC_DMA_SHIFT 8
#define TPM_SC_DMA_WIDTH 1
#define TPM_SC_DMA(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_DMA_SHIFT))&TPM_SC_DMA_MASK)
/* CNT Bit Fields */
#define TPM_CNT_COUNT_MASK 0xFFFFu
#define TPM_CNT_COUNT_SHIFT 0
#define TPM_CNT_COUNT_WIDTH 16
#define TPM_CNT_COUNT(x)
(((uint32_t)((uint32_t)(x))<<TPM_CNT_COUNT_SHIFT))&TPM_CNT_COUNT_MASK)
/* MOD Bit Fields */
#define TPM_MOD_MOD_MASK 0xFFFFu
#define TPM_MOD_MOD_SHIFT 0
#define TPM_MOD_MOD_WIDTH 16
#define TPM_MOD_MOD(x)
(((uint32_t)((uint32_t)(x))<<TPM_MOD_MOD_SHIFT))&TPM_MOD_MOD_MASK)
/* CnSC Bit Fields */
#define TPM_CnSC_DMA_MASK 0x1u
#define TPM_CnSC_DMA_SHIFT 0
#define TPM_CnSC_DMA_WIDTH 1
#define TPM_CnSC_DMA(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_DMA_SHIFT))&TPM_CnSC_DMA_MASK)
#define TPM_CnSC_ELSA_MASK 0x4u
#define TPM_CnSC_ELSA_SHIFT 2
#define TPM_CnSC_ELSA_WIDTH 1
#define TPM_CnSC_ELSA(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_ELSA_SHIFT))&TPM_CnSC_ELSA_MASK)
#define TPM_CnSC_ELSB_MASK 0x8u
#define TPM_CnSC_ELSB_SHIFT 3
#define TPM_CnSC_ELSB_WIDTH 1

```

```

#define TPM_CnSC_ELSB(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_ELSB_SHIFT) )&TPM_CnSC_ELSB_MASK)
#define TPM_CnSC_MSA_MASK 0x10u
#define TPM_CnSC_MSA_SHIFT 4
#define TPM_CnSC_MSA_WIDTH 1
#define TPM_CnSC_MSA(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_MSA_SHIFT) )&TPM_CnSC_MSA_MASK)
#define TPM_CnSC_MSB_MASK 0x20u
#define TPM_CnSC_MSB_SHIFT 5
#define TPM_CnSC_MSB_WIDTH 1
#define TPM_CnSC_MSB(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_MSB_SHIFT) )&TPM_CnSC_MSB_MASK)
#define TPM_CnSC_CHIE_MASK 0x40u
#define TPM_CnSC_CHIE_SHIFT 6
#define TPM_CnSC_CHIE_WIDTH 1
#define TPM_CnSC_CHIE(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_CHIE_SHIFT) )&TPM_CnSC_CHIE_MASK)
#define TPM_CnSC_CHF_MASK 0x80u
#define TPM_CnSC_CHF_SHIFT 7
#define TPM_CnSC_CHF_WIDTH 1
#define TPM_CnSC_CHF(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnSC_CHF_SHIFT) )&TPM_CnSC_CHF_MASK)
/* CnV Bit Fields */
#define TPM_CnV_VAL_MASK 0xFFFFu
#define TPM_CnV_VAL_SHIFT 0
#define TPM_CnV_VAL_WIDTH 16
#define TPM_CnV_VAL(x)
(((uint32_t)((uint32_t)(x))<<TPM_CnV_VAL_SHIFT) )&TPM_CnV_VAL_MASK)
/* STATUS Bit Fields */
#define TPM_STATUS_CHOF_MASK 0x1u
#define TPM_STATUS_CHOF_SHIFT 0
#define TPM_STATUS_CHOF_WIDTH 1
#define TPM_STATUS_CHOF(x)
(((uint32_t)((uint32_t)(x))<<TPM_STATUS_CHOF_SHIFT) )&TPM_STATUS_CHOF_MASK)
#define TPM_STATUS_CH1F_MASK 0x2u
#define TPM_STATUS_CH1F_SHIFT 1
#define TPM_STATUS_CH1F_WIDTH 1
#define TPM_STATUS_CH1F(x)
(((uint32_t)((uint32_t)(x))<<TPM_STATUS_CH1F_SHIFT) )&TPM_STATUS_CH1F_MASK)
#define TPM_STATUS_CH2F_MASK 0x4u
#define TPM_STATUS_CH2F_SHIFT 2
#define TPM_STATUS_CH2F_WIDTH 1
#define TPM_STATUS_CH2F(x)
(((uint32_t)((uint32_t)(x))<<TPM_STATUS_CH2F_SHIFT) )&TPM_STATUS_CH2F_MASK)
#define TPM_STATUS_CH3F_MASK 0x8u
#define TPM_STATUS_CH3F_SHIFT 3
#define TPM_STATUS_CH3F_WIDTH 1
#define TPM_STATUS_CH3F(x)
(((uint32_t)((uint32_t)(x))<<TPM_STATUS_CH3F_SHIFT) )&TPM_STATUS_CH3F_MASK)
#define TPM_STATUS_CH4F_MASK 0x10u

```

```

#define TPM_STATUS_CH4F_SHIFT 4
#define TPM_STATUS_CH4F_WIDTH 1
#define TPM_STATUS_CH4F(x) (((uint32_t)((uint32_t)(x))<<TPM_STATUS_CH4F_SHIFT))&TPM_STATUS_CH4F_MASK
#define TPM_STATUS_CH5F_MASK 0x20u
#define TPM_STATUS_CH5F_SHIFT 5
#define TPM_STATUS_CH5F_WIDTH 1
#define TPM_STATUS_CH5F(x) (((uint32_t)((uint32_t)(x))<<TPM_STATUS_CH5F_SHIFT))&TPM_STATUS_CH5F_MASK
#define TPM_STATUS_TOF_MASK 0x100u
#define TPM_STATUS_TOF_SHIFT 8
#define TPM_STATUS_TOF_WIDTH 1
#define TPM_STATUS_TOF(x) (((uint32_t)((uint32_t)(x))<<TPM_STATUS_TOF_SHIFT))&TPM_STATUS_TOF_MASK
/* CONF Bit Fields */
#define TPM_CONF_DOZEEN_MASK 0x20u
#define TPM_CONF_DOZEEN_SHIFT 5
#define TPM_CONF_DOZEEN_WIDTH 1
#define TPM_CONF_DOZEEN(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_DOZEEN_SHIFT))&TPM_CONF_DOZEEN_MASK
#define TPM_CONF_DBGMODE_MASK 0xC0u
#define TPM_CONF_DBGMODE_SHIFT 6
#define TPM_CONF_DBGMODE_WIDTH 2
#define TPM_CONF_DBGMODE(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_DBGMODE_SHIFT))&TPM_CONF_DBGMODE_MASK
#define TPM_CONF_GTBEEN_MASK 0x200u
#define TPM_CONF_GTBEEN_SHIFT 9
#define TPM_CONF_GTBEEN_WIDTH 1
#define TPM_CONF_GTBEEN(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_GTBEEN_SHIFT))&TPM_CONF_GTBEEN_MASK
#define TPM_CONF_CSOT_MASK 0x10000u
#define TPM_CONF_CSOT_SHIFT 16
#define TPM_CONF_CSOT_WIDTH 1
#define TPM_CONF_CSOT(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_CSOT_SHIFT))&TPM_CONF_CSOT_MASK
#define TPM_CONF_CSOO_MASK 0x20000u
#define TPM_CONF_CSOO_SHIFT 17
#define TPM_CONF_CSOO_WIDTH 1
#define TPM_CONF_CSOO(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_CSOO_SHIFT))&TPM_CONF_CSOO_MASK
#define TPM_CONF_CROT_MASK 0x40000u
#define TPM_CONF_CROT_SHIFT 18
#define TPM_CONF_CROT_WIDTH 1
#define TPM_CONF_CROT(x) (((uint32_t)((uint32_t)(x))<<TPM_CONF_CROT_SHIFT))&TPM_CONF_CROT_MASK
#define TPM_CONF_TRGSEL_MASK 0xF000000u
#define TPM_CONF_TRGSEL_SHIFT 24
#define TPM_CONF_TRGSEL_WIDTH 4

```

```

#define TPM_CONF_TRGSEL(x)
(((uint32_t)((uint32_t)(x))<<TPM_CONF_TRGSEL_SHIFT))&TPM_CONF_TRGSEL_MSK)

/*!
 * @}
 */
/* end of group TPM_Register_Masks */

/* TPM - Peripheral instance base addresses */
/** Peripheral TPM0 base address */
#define TPM0_BASE (0x40038000u)
/** Peripheral TPM0 base pointer */
#define TPM0 ((TPM_Type *)TPM0_BASE)
#define TPM0_BASE_PTR (TPM0)
/** Peripheral TPM1 base address */
#define TPM1_BASE (0x40039000u)
#define TPM1 ((TPM_Type *)TPM1_BASE)
#define TPM1_BASE_PTR (TPM1)
/** Peripheral TPM2 base address */
#define TPM2_BASE (0x4003A000u)
#define TPM2 ((TPM_Type *)TPM2_BASE)
#define TPM2_BASE_PTR (TPM2)
/** Array initializer of TPM peripheral base addresses */
#define TPM_BASE_ADDRS { TPM0_BASE, TPM1_BASE,
TPM2_BASE }
/** Array initializer of TPM peripheral base pointers */
#define TPM_BASE_PTRS { TPM0, TPM1, TPM2 }
/** Interrupt vectors for the TPM peripheral type */
#define TPM IRQS { TPM0_IRQn, TPM1_IRQn,
TPM2_IRQn }

/* -----
-- TPM - Register accessor macros
-----
*/
/*!
 * @addtogroup TPM_Register_Accessor_Macros TPM - Register accessor
macros
 * @{
 */
/* TPM - Register instance definitions */
/* TPM0 */
#define TPM0_SC TPM_SC_REG(TPM0)
#define TPM0_CNT TPM_CNT_REG(TPM0)
#define TPM0_MOD TPM_MOD_REG(TPM0)
#define TPM0_COSC TPM_CnSC_REG(TPM0, 0)
#define TPM0_COV TPM_CnV_REG(TPM0, 0)

```

```

#define TPM0_C1SC TPM_CnSC_REG(TPM0,1)
#define TPM0_C1V TPM_CnV_REG(TPM0,1)
#define TPM0_C2SC TPM_CnSC_REG(TPM0,2)
#define TPM0_C2V TPM_CnV_REG(TPM0,2)
#define TPM0_C3SC TPM_CnSC_REG(TPM0,3)
#define TPM0_C3V TPM_CnV_REG(TPM0,3)
#define TPM0_C4SC TPM_CnSC_REG(TPM0,4)
#define TPM0_C4V TPM_CnV_REG(TPM0,4)
#define TPM0_C5SC TPM_CnSC_REG(TPM0,5)
#define TPM0_C5V TPM_CnV_REG(TPM0,5)
#define TPM0_STATUS TPM_STATUS_REG(TPM0)
#define TPM0_CONF TPM_CONF_REG(TPM0)
/* TPM1 */
#define TPM1_SC TPM_SC_REG(TPM1)
#define TPM1_CNT TPM_CNT_REG(TPM1)
#define TPM1_MOD TPM_MOD_REG(TPM1)
#define TPM1_C0SC TPM_CnSC_REG(TPM1,0)
#define TPM1_C0V TPM_CnV_REG(TPM1,0)
#define TPM1_C1SC TPM_CnSC_REG(TPM1,1)
#define TPM1_C1V TPM_CnV_REG(TPM1,1)
#define TPM1_STATUS TPM_STATUS_REG(TPM1)
#define TPM1_CONF TPM_CONF_REG(TPM1)
/* TPM2 */
#define TPM2_SC TPM_SC_REG(TPM2)
#define TPM2_CNT TPM_CNT_REG(TPM2)
#define TPM2_MOD TPM_MOD_REG(TPM2)
#define TPM2_C0SC TPM_CnSC_REG(TPM2,0)
#define TPM2_C0V TPM_CnV_REG(TPM2,0)
#define TPM2_C1SC TPM_CnSC_REG(TPM2,1)
#define TPM2_C1V TPM_CnV_REG(TPM2,1)
#define TPM2_STATUS TPM_STATUS_REG(TPM2)
#define TPM2_CONF TPM_CONF_REG(TPM2)

/* TPM - Register array accessors */
#define TPM0_CnSC(index) TPM_CnSC_REG(TPM0,index)
#define TPM1_CnSC(index) TPM_CnSC_REG(TPM1,index)
#define TPM2_CnSC(index) TPM_CnSC_REG(TPM2,index)
#define TPM0_CnV(index) TPM_CnV_REG(TPM0,index)
#define TPM1_CnV(index) TPM_CnV_REG(TPM1,index)
#define TPM2_CnV(index) TPM_CnV_REG(TPM2,index)

/*!
 * @}
 */ /* end of group TPM_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group TPM_Peripheral_Access_Layer */

/*
-----  

-- TSI Peripheral Access Layer

```

```

----- */
----- */

/*!
 * @addtogroup TSI_Peripheral_Access_Layer TSI Peripheral Access Layer
 * @{
 */

/** TSI - Register Layout Typedef */
typedef struct {
    __IO uint32_t GENCS;                                /**< TSI General
Control and Status Register, offset: 0x0 */
    __IO uint32_t DATA;                                 /**< TSI DATA
Register, offset: 0x4 */
    __IO uint32_t TSHD;                                /**< TSI Threshold
Register, offset: 0x8 */
} TSI_Type, *TSI_MemMapPtr;

/*
----- */
----- */

-- TSI - Register accessor macros
----- */

/*!
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor
macros
 * @{
 */

/* TSI - Register accessors */
#define TSI_GENCS_REG(base)          ((base) -> GENCS)
#define TSI_DATA_REG(base)           ((base) -> DATA)
#define TSI_TSHD_REG(base)           ((base) -> TSHD)

/*!
 * @}
 */
/* end of group TSI_Register_Accessor_Macros */

/*
----- */
----- */

-- TSI Register Masks
----- */

/*!
 * @addtogroup TSI_Register_Masks TSI Register Masks
 * @{
 */

/* GENCS Bit Fields */
#define TSI_GENCS_CURSW_MASK        0x2u

```

```

#define TSI_GENCS_CURSW_SHIFT 1
#define TSI_GENCS_CURSW_WIDTH 1
#define TSI_GENCS_CURSW(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_CURSW_SHIFT))&TSI_GENCS_CURSW_MASK
#define TSI_GENCS_EOSF_MASK 0x4u
#define TSI_GENCS_EOSF_SHIFT 2
#define TSI_GENCS_EOSF_WIDTH 1
#define TSI_GENCS_EOSF(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_EOSF_SHIFT))&TSI_GENCS_EOSF_MASK
#define TSI_GENCS_SCNIP_MASK 0x8u
#define TSI_GENCS_SCNIP_SHIFT 3
#define TSI_GENCS_SCNIP_WIDTH 1
#define TSI_GENCS_SCNIP(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_SCNIP_SHIFT))&TSI_GENCS_SCNIP_MASK
#define TSI_GENCS_STM_MASK 0x10u
#define TSI_GENCS_STM_SHIFT 4
#define TSI_GENCS_STM_WIDTH 1
#define TSI_GENCS_STM(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_STM_SHIFT))&TSI_GENCS_STM_MASK
#define TSI_GENCS_STPE_MASK 0x20u
#define TSI_GENCS_STPE_SHIFT 5
#define TSI_GENCS_STPE_WIDTH 1
#define TSI_GENCS_STPE(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_STPE_SHIFT))&TSI_GENCS_STPE_MASK
#define TSI_GENCS_TSIIEN_MASK 0x40u
#define TSI_GENCS_TSIIEN_SHIFT 6
#define TSI_GENCS_TSIIEN_WIDTH 1
#define TSI_GENCS_TSIIEN(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_TSIIEN_SHIFT))&TSI_GENCS_TSIIEN_MASK
#define TSI_GENCS_TSIEN_MASK 0x80u
#define TSI_GENCS_TSIEN_SHIFT 7
#define TSI_GENCS_TSIEN_WIDTH 1
#define TSI_GENCS_TSIEN(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_TSIEN_SHIFT))&TSI_GENCS_TSIEN_MASK
#define TSI_GENCS_NSCN_MASK 0x1F00u
#define TSI_GENCS_NSCN_SHIFT 8
#define TSI_GENCS_NSCN_WIDTH 5
#define TSI_GENCS_NSCN(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_NSCN_SHIFT))&TSI_GENCS_NSCN_MASK
#define TSI_GENCS_PS_MASK 0xE000u
#define TSI_GENCS_PS_SHIFT 13
#define TSI_GENCS_PS_WIDTH 3
#define TSI_GENCS_PS(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_PS_SHIFT))&TSI_GENCS_PS_MASK
#define TSI_GENCS_EXTCHRG_MASK 0x70000u
#define TSI_GENCS_EXTCHRG_SHIFT 16
#define TSI_GENCS_EXTCHRG_WIDTH 3
#define TSI_GENCS_EXTCHRG(x) (((uint32_t)((uint32_t)(x))<<TSI_GENCS_EXTCHRG_SHIFT))&TSI_GENCS_EXTCHRG_MASK

```

```

#define TSI_GENCS_DVOLT_MASK          0x180000u
#define TSI_GENCS_DVOLT_SHIFT         19
#define TSI_GENCS_DVOLT_WIDTH         2
#define TSI_GENCS_DVOLT(x)           (((uint32_t)((uint32_t)(x))<<TSI_GENCS_DVOLT_SHIFT))&TSI_GENCS_DVOLT_MASK)
#define TSI_GENCS_REFCHRG_MASK        0xE00000u
#define TSI_GENCS_REFCHRG_SHIFT       21
#define TSI_GENCS_REFCHRG_WIDTH       3
#define TSI_GENCS_REFCHRG(x)         (((uint32_t)((uint32_t)(x))<<TSI_GENCS_REFCHRG_SHIFT))&TSI_GENCS_REFCHRG_MASK)
#define TSI_GENCS_MODE_MASK           0xF000000u
#define TSI_GENCS_MODE_SHIFT          24
#define TSI_GENCS_MODE_WIDTH          4
#define TSI_GENCS_MODE(x)            (((uint32_t)((uint32_t)(x))<<TSI_GENCS_MODE_SHIFT))&TSI_GENCS_MODE_MASK)
#define TSI_GENCS_ESOR_MASK           0x10000000u
#define TSI_GENCS_ESOR_SHIFT          28
#define TSI_GENCS_ESOR_WIDTH          1
#define TSI_GENCS_ESOR(x)             (((uint32_t)((uint32_t)(x))<<TSI_GENCS_ESOR_SHIFT))&TSI_GENCS_ESOR_MASK)
#define TSI_GENCS_OUTRGF_MASK         0x80000000u
#define TSI_GENCS_OUTRGF_SHIFT        31
#define TSI_GENCS_OUTRGF_WIDTH        1
#define TSI_GENCS_OUTRGF(x)           (((uint32_t)((uint32_t)(x))<<TSI_GENCS_OUTRGF_SHIFT))&TSI_GENCS_OUTRGF_MASK)
/* DATA Bit Fields */
#define TSI_DATA_TSICNT_MASK          0xFFFFu
#define TSI_DATA_TSICNT_SHIFT         0
#define TSI_DATA_TSICNT_WIDTH         16
#define TSI_DATA_TSICNT(x)            (((uint32_t)((uint32_t)(x))<<TSI_DATA_TSICNT_SHIFT))&TSI_DATA_TSICNT_MASK)
#define TSI_DATA_SWTS_MASK            0x400000u
#define TSI_DATA_SWTS_SHIFT           22
#define TSI_DATA_SWTS_WIDTH           1
#define TSI_DATA_SWTS(x)              (((uint32_t)((uint32_t)(x))<<TSI_DATA_SWTS_SHIFT))&TSI_DATA_SWTS_MASK)
#define TSI_DATA_DMAEN_MASK           0x800000u
#define TSI_DATA_DMAEN_SHIFT          23
#define TSI_DATA_DMAEN_WIDTH          1
#define TSI_DATA_DMAEN(x)              (((uint32_t)((uint32_t)(x))<<TSI_DATA_DMAEN_SHIFT))&TSI_DATA_DMAEN_MASK)
#define TSI_DATA_TSICH_MASK           0xF0000000u
#define TSI_DATA_TSICH_SHIFT          28
#define TSI_DATA_TSICH_WIDTH          4
#define TSI_DATA_TSICH(x)              (((uint32_t)((uint32_t)(x))<<TSI_DATA_TSICH_SHIFT))&TSI_DATA_TSICH_MASK)
/* TSHD Bit Fields */
#define TSI_TSHD_THRESL_MASK          0xFFFFu
#define TSI_TSHD_THRESL_SHIFT         0
#define TSI_TSHD_THRESL_WIDTH         16

```

```

#define TSI_TSHD_THRESL(x)
(((uint32_t)((uint32_t)(x))<<TSI_TSHD_THRESL_SHIFT))&TSI_TSHD_THRESL_MASK
K)
#define TSI_TSHD_THRESH_MASK          0xFFFF0000u
#define TSI_TSHD_THRESH_SHIFT        16
#define TSI_TSHD_THRESH_WIDTH        16
#define TSI_TSHD_THRESH(x)
(((uint32_t)((uint32_t)(x))<<TSI_TSHD_THRESH_SHIFT))&TSI_TSHD_THRESH_MASK
K)

/*!
 * @}
 */ /* end of group TSI_Register_Masks */

/* TSI - Peripheral instance base addresses */
/** Peripheral TSI0 base address */
#define TSI0_BASE                    (0x40045000u)
/** Peripheral TSI0 base pointer */
#define TSI0                         ((TSI_Type *)TSI0_BASE)
#define TSI0_PTR                     (TSI0)
/** Array initializer of TSI peripheral base addresses */
#define TSI_BASE_ADDRS               { TSI0_BASE }
/** Array initializer of TSI peripheral base pointers */
#define TSI_BASE_PTRS                { TSI0 }
/** Interrupt vectors for the TSI peripheral type */
#define TSI IRQS                      { TSI0_IRQn }

/* -----
-- TSI - Register accessor macros
-----
*/
/*!
 * @addtogroup TSI_Register_Accessor_Macros TSI - Register accessor
macros
 * @{
 */
/* TSI - Register instance definitions */
/* TSI0 */
#define TSI0_GENCS                  TSI_GENCS_REG(TSI0)
#define TSI0_DATA                   TSI_DATA_REG(TSI0)
#define TSI0_TSHD                   TSI_TSHD_REG(TSI0)

/*!
 * @}
 */
/* end of group TSI_Register_Accessor_Macros */

/*!
 * @{
 */

```

```

 */ /* end of group TSI_Peripheral_Access_Layer */

/*
-----
-- UART Peripheral Access Layer
-----
*/

/*!
 * @addtogroup UART_Peripheral_Access_Layer UART Peripheral Access Layer
 * @{
 */

/** UART - Register Layout Typedef */
typedef struct {
    __IO uint8_t BDH;                                /**< UART Baud Rate
Register: High, offset: 0x0 */
    __IO uint8_t BDL;                                /**< UART Baud Rate
Register: Low, offset: 0x1 */
    __IO uint8_t C1;                                 /**< UART Control
Register 1, offset: 0x2 */
    __IO uint8_t C2;                                 /**< UART Control
Register 2, offset: 0x3 */
    __I  uint8_t S1;                                /**< UART Status
Register 1, offset: 0x4 */
    __IO uint8_t S2;                                /**< UART Status
Register 2, offset: 0x5 */
    __IO uint8_t C3;                                 /**< UART Control
Register 3, offset: 0x6 */
    __IO uint8_t D;                                 /**< UART Data
Register, offset: 0x7 */
    __IO uint8_t C4;                                 /**< UART Control
Register 4, offset: 0x8 */
} UART_Type, *UART_MemMapPtr;

/*
-----
-- UART - Register accessor macros
-----
*/

/*!
 * @addtogroup UART_Register_Accessor_Macros UART - Register accessor
macros
 * @{
 */

/* UART - Register accessors */
#define UART_BDH_REG(base)          ((base)->BDH)
#define UART_BDL_REG(base)          ((base)->BDL)
#define UART_C1_REG(base)           ((base)->C1)
#define UART_C2_REG(base)           ((base)->C2)

```

```

#define UART_S1_REG(base)          ((base)->S1)
#define UART_S2_REG(base)          ((base)->S2)
#define UART_C3_REG(base)          ((base)->C3)
#define UART_D_REG(base)           ((base)->D)
#define UART_C4_REG(base)          ((base)->C4)

/* !
 * @}
 */ /* end of group UART_Register_Accessor_Macros */

/*
-----  

-- UART Register Masks  

-----  

---- */

/*!  

 * @addtogroup UART_Register_Masks  

 * @{
 */
/* BDH Bit Fields */
#define UART_BDH_SBR_MASK          0x1Fu
#define UART_BDH_SBR_SHIFT         0
#define UART_BDH_SBR_WIDTH         5
#define UART_BDH_SBR(x)  

((uint8_t)((uint8_t)(x)<<UART_BDH_SBR_SHIFT))&UART_BDH_SBR_MASK)
#define UART_BDH_SBNS_MASK          0x20u
#define UART_BDH_SBNS_SHIFT         5
#define UART_BDH_SBNS_WIDTH         1
#define UART_BDH_SBNS(x)  

((uint8_t)((uint8_t)(x)<<UART_BDH_SBNS_SHIFT))&UART_BDH_SBNS_MASK)
#define UART_BDH_RXEDGIE_MASK        0x40u
#define UART_BDH_RXEDGIE_SHIFT       6
#define UART_BDH_RXEDGIE_WIDTH       1
#define UART_BDH_RXEDGIE(x)  

((uint8_t)((uint8_t)(x)<<UART_BDH_RXEDGIE_SHIFT))&UART_BDH_RXEDGIE_MASK)
#define UART_BDH_LBKDIE_MASK         0x80u
#define UART_BDH_LBKDIE_SHIFT        7
#define UART_BDH_LBKDIE_WIDTH        1
#define UART_BDH_LBKDIE(x)  

((uint8_t)((uint8_t)(x)<<UART_BDH_LBKDIE_SHIFT))&UART_BDH_LBKDIE_MASK)
/* BDL Bit Fields */
#define UART_BDL_SBR_MASK           0xFFu
#define UART_BDL_SBR_SHIFT          0
#define UART_BDL_SBR_WIDTH          8
#define UART_BDL_SBR(x)  

((uint8_t)((uint8_t)(x)<<UART_BDL_SBR_SHIFT))&UART_BDL_SBR_MASK)
/* C1 Bit Fields */
#define UART_C1_PT_MASK              0x1u
#define UART_C1_PT_SHIFT             0
#define UART_C1_PT_WIDTH             1

```

```

#define UART_C1_PT(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_PT_SHIFT))&UART_C1_PT_MASK)
#define UART_C1_PE_MASK 0x2u
#define UART_C1_PE_SHIFT 1
#define UART_C1_PE_WIDTH 1
#define UART_C1_PE(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_PE_SHIFT))&UART_C1_PE_MASK)
#define UART_C1_I LT_MASK 0x4u
#define UART_C1_I LT_SHIFT 2
#define UART_C1_I LT_WIDTH 1
#define UART_C1_I LT(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_I LT_SHIFT))&UART_C1_I LT_MASK)
#define UART_C1_WAKE_MASK 0x8u
#define UART_C1_WAKE_SHIFT 3
#define UART_C1_WAKE_WIDTH 1
#define UART_C1_WAKE(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_WAKE_SHIFT))&UART_C1_WAKE_MASK)
#define UART_C1_M_MASK 0x10u
#define UART_C1_M_SHIFT 4
#define UART_C1_M_WIDTH 1
#define UART_C1_M(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_M_SHIFT))&UART_C1_M_MASK)
#define UART_C1_RSRC_MASK 0x20u
#define UART_C1_RSRC_SHIFT 5
#define UART_C1_RSRC_WIDTH 1
#define UART_C1_RSRC(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_RSRC_SHIFT))&UART_C1_RSRC_MASK)
#define UART_C1_UARTSWAI_MASK 0x40u
#define UART_C1_UARTSWAI_SHIFT 6
#define UART_C1_UARTSWAI_WIDTH 1
#define UART_C1_UARTSWAI(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_UARTSWAI_SHIFT))&UART_C1_UARTSWAI_MASK)
#define UART_C1_LOOPS_MASK 0x80u
#define UART_C1_LOOPS_SHIFT 7
#define UART_C1_LOOPS_WIDTH 1
#define UART_C1_LOOPS(x)
(((uint8_t)((uint8_t)(x))<<UART_C1_LOOPS_SHIFT))&UART_C1_LOOPS_MASK)
/* C2 Bit Fields */
#define UART_C2_SBK_MASK 0x1u
#define UART_C2_SBK_SHIFT 0
#define UART_C2_SBK_WIDTH 1
#define UART_C2_SBK(x)
(((uint8_t)((uint8_t)(x))<<UART_C2_SBK_SHIFT))&UART_C2_SBK_MASK)
#define UART_C2_RWU_MASK 0x2u
#define UART_C2_RWU_SHIFT 1
#define UART_C2_RWU_WIDTH 1
#define UART_C2_RWU(x)
(((uint8_t)((uint8_t)(x))<<UART_C2_RWU_SHIFT))&UART_C2_RWU_MASK)
#define UART_C2_RE_MASK 0x4u
#define UART_C2_RE_SHIFT 2
#define UART_C2_RE_WIDTH 1
#define UART_C2_RE(x)
(((uint8_t)((uint8_t)(x))<<UART_C2_RE_SHIFT))&UART_C2_RE_MASK)

```

```

#define UART_C2_TE_MASK          0x8u
#define UART_C2_TE_SHIFT         3
#define UART_C2_TE_WIDTH         1
#define UART_C2_TE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_TE_SHIFT)&UART_C2_TE_MASK)
#define UART_C2_ILIE_MASK        0x10u
#define UART_C2_ILIE_SHIFT       4
#define UART_C2_ILIE_WIDTH       1
#define UART_C2_ILIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_ILIE_SHIFT)&UART_C2_ILIE_MASK)
#define UART_C2_RIE_MASK         0x20u
#define UART_C2_RIE_SHIFT        5
#define UART_C2_RIE_WIDTH        1
#define UART_C2_RIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_RIE_SHIFT)&UART_C2_RIE_MASK)
#define UART_C2_TCIE_MASK        0x40u
#define UART_C2_TCIE_SHIFT       6
#define UART_C2_TCIE_WIDTH       1
#define UART_C2_TCIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_TCIE_SHIFT)&UART_C2_TCIE_MASK)
#define UART_C2_TIE_MASK         0x80u
#define UART_C2_TIE_SHIFT        7
#define UART_C2_TIE_WIDTH        1
#define UART_C2_TIE(x) (((uint8_t)((uint8_t)(x))<<UART_C2_TIE_SHIFT)&UART_C2_TIE_MASK)
/* S1 Bit Fields */
#define UART_S1_PF_MASK          0x1u
#define UART_S1_PF_SHIFT         0
#define UART_S1_PF_WIDTH         1
#define UART_S1_PF(x) (((uint8_t)((uint8_t)(x))<<UART_S1_PF_SHIFT)&UART_S1_PF_MASK)
#define UART_S1_FE_MASK          0x2u
#define UART_S1_FE_SHIFT         1
#define UART_S1_FE_WIDTH         1
#define UART_S1_FE(x) (((uint8_t)((uint8_t)(x))<<UART_S1_FE_SHIFT)&UART_S1_FE_MASK)
#define UART_S1_NF_MASK          0x4u
#define UART_S1_NF_SHIFT         2
#define UART_S1_NF_WIDTH         1
#define UART_S1_NF(x) (((uint8_t)((uint8_t)(x))<<UART_S1_NF_SHIFT)&UART_S1_NF_MASK)
#define UART_S1_OR_MASK          0x8u
#define UART_S1_OR_SHIFT         3
#define UART_S1_OR_WIDTH         1
#define UART_S1_OR(x) (((uint8_t)((uint8_t)(x))<<UART_S1_OR_SHIFT)&UART_S1_OR_MASK)
#define UART_S1_IDLE_MASK         0x10u
#define UART_S1_IDLE_SHIFT        4
#define UART_S1_IDLE_WIDTH        1
#define UART_S1_IDLE(x) (((uint8_t)((uint8_t)(x))<<UART_S1_IDLE_SHIFT)&UART_S1_IDLE_MASK)
#define UART_S1_RDRF_MASK        0x20u
#define UART_S1_RDRF_SHIFT       5
#define UART_S1_RDRF_WIDTH       1

```

```

#define UART_S1_RDRF(x)
(((uint8_t)((uint8_t)(x))<<UART_S1_RDRF_SHIFT))&UART_S1_RDRF_MASK
#define UART_S1_TC_MASK 0x40u
#define UART_S1_TC_SHIFT 6
#define UART_S1_TC_WIDTH 1
#define UART_S1_TC(x)
(((uint8_t)((uint8_t)(x))<<UART_S1_TC_SHIFT))&UART_S1_TC_MASK
#define UART_S1_TDRE_MASK 0x80u
#define UART_S1_TDRE_SHIFT 7
#define UART_S1_TDRE_WIDTH 1
#define UART_S1_TDRE(x)
(((uint8_t)((uint8_t)(x))<<UART_S1_TDRE_SHIFT))&UART_S1_TDRE_MASK
/* S2 Bit Fields */
#define UART_S2_RAF_MASK 0x1u
#define UART_S2_RAF_SHIFT 0
#define UART_S2_RAF_WIDTH 1
#define UART_S2_RAF(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_RAF_SHIFT))&UART_S2_RAF_MASK
#define UART_S2_LBKDE_MASK 0x2u
#define UART_S2_LBKDE_SHIFT 1
#define UART_S2_LBKDE_WIDTH 1
#define UART_S2_LBKDE(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_LBKDE_SHIFT))&UART_S2_LBKDE_MASK
#define UART_S2_BRK13_MASK 0x4u
#define UART_S2_BRK13_SHIFT 2
#define UART_S2_BRK13_WIDTH 1
#define UART_S2_BRK13(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_BRK13_SHIFT))&UART_S2_BRK13_MASK
#define UART_S2_RWUID_MASK 0x8u
#define UART_S2_RWUID_SHIFT 3
#define UART_S2_RWUID_WIDTH 1
#define UART_S2_RWUID(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_RWUID_SHIFT))&UART_S2_RWUID_MASK
#define UART_S2_RXINV_MASK 0x10u
#define UART_S2_RXINV_SHIFT 4
#define UART_S2_RXINV_WIDTH 1
#define UART_S2_RXINV(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_RXINV_SHIFT))&UART_S2_RXINV_MASK
#define UART_S2_RXEDGIF_MASK 0x40u
#define UART_S2_RXEDGIF_SHIFT 6
#define UART_S2_RXEDGIF_WIDTH 1
#define UART_S2_RXEDGIF(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_RXEDGIF_SHIFT))&UART_S2_RXEDGIF_MASK
#define UART_S2_LBKDIF_MASK 0x80u
#define UART_S2_LBKDIF_SHIFT 7
#define UART_S2_LBKDIF_WIDTH 1
#define UART_S2_LBKDIF(x)
(((uint8_t)((uint8_t)(x))<<UART_S2_LBKDIF_SHIFT))&UART_S2_LBKDIF_MASK
/* C3 Bit Fields */
#define UART_C3_PEIE_MASK 0x1u
#define UART_C3_PEIE_SHIFT 0
#define UART_C3_PEIE_WIDTH 1
#define UART_C3_PEIE(x)
(((uint8_t)((uint8_t)(x))<<UART_C3_PEIE_SHIFT))&UART_C3_PEIE_MASK)

```

```

#define UART_C3_FEIE_MASK          0x2u
#define UART_C3_FEIE_SHIFT         1
#define UART_C3_FEIE_WIDTH         1
#define UART_C3_FEIE(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_FEIE_SHIFT))&UART_C3_FEIE_MASK)
#define UART_C3_NEIE_MASK          0x4u
#define UART_C3_NEIE_SHIFT         2
#define UART_C3_NEIE_WIDTH         1
#define UART_C3_NEIE(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_NEIE_SHIFT))&UART_C3_NEIE_MASK)
#define UART_C3_ORIE_MASK          0x8u
#define UART_C3_ORIE_SHIFT         3
#define UART_C3_ORIE_WIDTH         1
#define UART_C3_ORIE(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_ORIE_SHIFT))&UART_C3_ORIE_MASK)
#define UART_C3_TXINV_MASK          0x10u
#define UART_C3_TXINV_SHIFT         4
#define UART_C3_TXINV_WIDTH         1
#define UART_C3_TXINV(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_TXINV_SHIFT))&UART_C3_TXINV_MASK)
#define UART_C3_TXDIR_MASK          0x20u
#define UART_C3_TXDIR_SHIFT         5
#define UART_C3_TXDIR_WIDTH         1
#define UART_C3_TXDIR(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_TXDIR_SHIFT))&UART_C3_TXDIR_MASK)
#define UART_C3_T8_MASK             0x40u
#define UART_C3_T8_SHIFT              6
#define UART_C3_T8_WIDTH              1
#define UART_C3_T8(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_T8_SHIFT))&UART_C3_T8_MASK)
#define UART_C3_R8_MASK             0x80u
#define UART_C3_R8_SHIFT              7
#define UART_C3_R8_WIDTH              1
#define UART_C3_R8(x)
  (((uint8_t)((uint8_t)(x))<<UART_C3_R8_SHIFT))&UART_C3_R8_MASK)
/* D Bit Fields */
#define UART_D_R0T0_MASK            0x1u
#define UART_D_R0T0_SHIFT              0
#define UART_D_R0T0_WIDTH              1
#define UART_D_R0T0(x)
  (((uint8_t)((uint8_t)(x))<<UART_D_R0T0_SHIFT))&UART_D_R0T0_MASK)
#define UART_D_R1T1_MASK            0x2u
#define UART_D_R1T1_SHIFT              1
#define UART_D_R1T1_WIDTH              1
#define UART_D_R1T1(x)
  (((uint8_t)((uint8_t)(x))<<UART_D_R1T1_SHIFT))&UART_D_R1T1_MASK)
#define UART_D_R2T2_MASK            0x4u
#define UART_D_R2T2_SHIFT              2
#define UART_D_R2T2_WIDTH              1
#define UART_D_R2T2(x)
  (((uint8_t)((uint8_t)(x))<<UART_D_R2T2_SHIFT))&UART_D_R2T2_MASK)
#define UART_D_R3T3_MASK            0x8u
#define UART_D_R3T3_SHIFT              3
#define UART_D_R3T3_WIDTH              1

```

```

#define UART_D_R3T3(x)
(((uint8_t)((uint8_t)(x))<<UART_D_R3T3_SHIFT))&UART_D_R3T3_MASK)
#define UART_D_R4T4_MASK 0x10u
#define UART_D_R4T4_SHIFT 4
#define UART_D_R4T4_WIDTH 1
#define UART_D_R4T4(x)
(((uint8_t)((uint8_t)(x))<<UART_D_R4T4_SHIFT))&UART_D_R4T4_MASK)
#define UART_D_R5T5_MASK 0x20u
#define UART_D_R5T5_SHIFT 5
#define UART_D_R5T5_WIDTH 1
#define UART_D_R5T5(x)
(((uint8_t)((uint8_t)(x))<<UART_D_R5T5_SHIFT))&UART_D_R5T5_MASK)
#define UART_D_R6T6_MASK 0x40u
#define UART_D_R6T6_SHIFT 6
#define UART_D_R6T6_WIDTH 1
#define UART_D_R6T6(x)
(((uint8_t)((uint8_t)(x))<<UART_D_R6T6_SHIFT))&UART_D_R6T6_MASK)
#define UART_D_R7T7_MASK 0x80u
#define UART_D_R7T7_SHIFT 7
#define UART_D_R7T7_WIDTH 1
#define UART_D_R7T7(x)
(((uint8_t)((uint8_t)(x))<<UART_D_R7T7_SHIFT))&UART_D_R7T7_MASK)
/* C4 Bit Fields */
#define UART_C4_RDMAS_MASK 0x20u
#define UART_C4_RDMAS_SHIFT 5
#define UART_C4_RDMAS_WIDTH 1
#define UART_C4_RDMAS(x)
(((uint8_t)((uint8_t)(x))<<UART_C4_RDMAS_SHIFT))&UART_C4_RDMAS_MASK)
#define UART_C4_TDMAS_MASK 0x80u
#define UART_C4_TDMAS_SHIFT 7
#define UART_C4_TDMAS_WIDTH 1
#define UART_C4_TDMAS(x)
(((uint8_t)((uint8_t)(x))<<UART_C4_TDMAS_SHIFT))&UART_C4_TDMAS_MASK)

/*
 * @}
 */ /* end of group UART_Register_Masks */

```

```

/* UART - Peripheral instance base addresses */
/** Peripheral UART1 base address */
#define UART1_BASE (0x4006B000u)
/** Peripheral UART1 base pointer */
#define UART1 ((UART_Type
*)UART1_BASE)
#define UART1_BASE_PTR (UART1)
/** Peripheral UART2 base address */
#define UART2_BASE (0x4006C000u)
/** Peripheral UART2 base pointer */
#define UART2 ((UART_Type
*)UART2_BASE)
#define UART2_BASE_PTR (UART2)
/** Array initializer of UART peripheral base addresses */

```

```

#define UART_BASE_ADDRS           { 0u, UART1_BASE,
UART2_BASE }
/** Array initializer of UART peripheral base pointers */
#define UART_BASE_PTRS            { (UART_Type *)0u,
UART1, UART2 }
/** Interrupt vectors for the UART peripheral type */
#define UART_RX_TX IRQS          { NotAvail_IRQn,
UART1_IRQn, UART2_IRQn }
#define UART_ERR IRQS             { NotAvail_IRQn,
UART1_IRQn, UART2_IRQn }

/* -----
-- UART - Register accessor macros
-----
*/
/*!
* @addtogroup UART_Register_Accessor_Macros UART - Register accessor
macros
* @{
*/
/* UART - Register instance definitions */
/* UART1 */
#define UART1_BDH                 UART_BDH_REG(UART1)
#define UART1_BDL                 UART_BDL_REG(UART1)
#define UART1_C1                  UART_C1_REG(UART1)
#define UART1_C2                  UART_C2_REG(UART1)
#define UART1_S1                  UART_S1_REG(UART1)
#define UART1_S2                  UART_S2_REG(UART1)
#define UART1_C3                  UART_C3_REG(UART1)
#define UART1_D                   UART_D_REG(UART1)
#define UART1_C4                  UART_C4_REG(UART1)
/* UART2 */
#define UART2_BDH                 UART_BDH_REG(UART2)
#define UART2_BDL                 UART_BDL_REG(UART2)
#define UART2_C1                  UART_C1_REG(UART2)
#define UART2_C2                  UART_C2_REG(UART2)
#define UART2_S1                  UART_S1_REG(UART2)
#define UART2_S2                  UART_S2_REG(UART2)
#define UART2_C3                  UART_C3_REG(UART2)
#define UART2_D                   UART_D_REG(UART2)
#define UART2_C4                  UART_C4_REG(UART2)

/*!
* @}
*/
/* end of group UART_Register_Accessor_Macros */

/*!
* @}
*/
/* end of group UART_Peripheral_Access_Layer */

```

```

/* -----
-- UART0 Peripheral Access Layer
-----
---- */

/*!
 * @addtogroup UART0_Peripheral_Access_Layer UART0 Peripheral Access
Layer
 * @{
 */

/** UART0 - Register Layout Typedef */
typedef struct {
    __IO uint8_t BDH;                                /**< UART Baud Rate
Register High, offset: 0x0 */
    __IO uint8_t BDL;                                /**< UART Baud Rate
Register Low, offset: 0x1 */
    __IO uint8_t C1;                                 /**< UART Control
Register 1, offset: 0x2 */
    __IO uint8_t C2;                                 /**< UART Control
Register 2, offset: 0x3 */
    __IO uint8_t S1;                                 /**< UART Status
Register 1, offset: 0x4 */
    __IO uint8_t S2;                                 /**< UART Status
Register 2, offset: 0x5 */
    __IO uint8_t C3;                                 /**< UART Control
Register 3, offset: 0x6 */
    __IO uint8_t D;                                  /**< UART Data
Register, offset: 0x7 */
    __IO uint8_t MA1;                               /**< UART Match
Address Registers 1, offset: 0x8 */
    __IO uint8_t MA2;                               /**< UART Match
Address Registers 2, offset: 0x9 */
    __IO uint8_t C4;                                 /**< UART Control
Register 4, offset: 0xA */
    __IO uint8_t C5;                                 /**< UART Control
Register 5, offset: 0xB */
} UART0_Type, *UART0_MemMapPtr;

/* -----
-- UART0 - Register accessor macros
-----
---- */

/*!
 * @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor
macros
 * @{
 */

```

```

/* UART0 - Register accessors */
#define UART0_BDH_REG(base)          ((base)->BDH)
#define UART0_BDL_REG(base)          ((base)->BDL)
#define UART0_C1_REG(base)           ((base)->C1)
#define UART0_C2_REG(base)           ((base)->C2)
#define UART0_S1_REG(base)           ((base)->S1)
#define UART0_S2_REG(base)           ((base)->S2)
#define UART0_C3_REG(base)           ((base)->C3)
#define UART0_D_REG(base)            ((base)->D)
#define UART0_MA1_REG(base)          ((base)->MA1)
#define UART0_MA2_REG(base)          ((base)->MA2)
#define UART0_C4_REG(base)           ((base)->C4)
#define UART0_C5_REG(base)           ((base)->C5)

/*!
 * @}
 */ /* end of group UART0_Register_Accessor_Macros */

/* -----
-- UART0 Register Masks
-----
*/
/*!
 * @addtogroup UART0_Register_Masks UART0 Register Masks
 * @{
 */

/* BDH Bit Fields */
#define UART0_BDH_SBR_MASK          0x1Fu
#define UART0_BDH_SBR_SHIFT         0
#define UART0_BDH_SBR_WIDTH         5
#define UART0_BDH_SBR(x)            (((uint8_t)(x))<<UART0_BDH_SBR_SHIFT)&UART0_BDH_SBR_MASK)
#define UART0_BDH_SBNS_MASK          0x20u
#define UART0_BDH_SBNS_SHIFT         5
#define UART0_BDH_SBNS_WIDTH         1
#define UART0_BDH_SBNS(x)            (((uint8_t)(x))<<UART0_BDH_SBNS_SHIFT)&UART0_BDH_SBNS_MASK)
#define UART0_BDH_RXEDGIE_MASK        0x40u
#define UART0_BDH_RXEDGIE_SHIFT       6
#define UART0_BDH_RXEDGIE_WIDTH       1
#define UART0_BDH_RXEDGIE(x)          (((uint8_t)(x))<<UART0_BDH_RXEDGIE_SHIFT)&UART0_BDH_RXEDGIE_MASK)
#define UART0_BDH_LBKDIE_MASK         0x80u
#define UART0_BDH_LBKDIE_SHIFT        7
#define UART0_BDH_LBKDIE_WIDTH        1
#define UART0_BDH_LBKDIE(x)           (((uint8_t)(x))<<UART0_BDH_LBKDIE_SHIFT)&UART0_BDH_LBKDIE_MASK)

```

```

/* BDL Bit Fields */
#define UART0_BDL_SBR_MASK          0xFFu
#define UART0_BDL_SBR_SHIFT         0
#define UART0_BDL_SBR_WIDTH         8
#define UART0_BDL_SBR(x)            (((uint8_t)((uint8_t)(x))<<UART0_BDL_SBR_SHIFT))&UART0_BDL_SBR_MASK
/* C1 Bit Fields */
#define UART0_C1_PT_MASK            0x1u
#define UART0_C1_PT_SHIFT           0
#define UART0_C1_PT_WIDTH           1
#define UART0_C1_PT(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_PT_SHIFT))&UART0_C1_PT_MASK
#define UART0_C1_PE_MASK             0x2u
#define UART0_C1_PE_SHIFT            1
#define UART0_C1_PE_WIDTH           1
#define UART0_C1_PE(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_PE_SHIFT))&UART0_C1_PE_MASK
#define UART0_C1_ILT_MASK            0x4u
#define UART0_C1_ILT_SHIFT           2
#define UART0_C1_ILT_WIDTH           1
#define UART0_C1_ILT(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_ILT_SHIFT))&UART0_C1_ILT_MASK
#define UART0_C1_WAKE_MASK           0x8u
#define UART0_C1_WAKE_SHIFT          3
#define UART0_C1_WAKE_WIDTH          1
#define UART0_C1_WAKE(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_WAKE_SHIFT))&UART0_C1_WAKE_MASK
#define UART0_C1_M_MASK              0x10u
#define UART0_C1_M_SHIFT             4
#define UART0_C1_M_WIDTH              1
#define UART0_C1_M(x)                (((uint8_t)((uint8_t)(x))<<UART0_C1_M_SHIFT))&UART0_C1_M_MASK
#define UART0_C1_RSRC_MASK            0x20u
#define UART0_C1_RSRC_SHIFT           5
#define UART0_C1_RSRC_WIDTH           1
#define UART0_C1_RSRC(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_RSRC_SHIFT))&UART0_C1_RSRC_MASK
#define UART0_C1_DOZEEN_MASK          0x40u
#define UART0_C1_DOZEEN_SHIFT         6
#define UART0_C1_DOZEEN_WIDTH         1
#define UART0_C1_DOZEEN(x)            (((uint8_t)((uint8_t)(x))<<UART0_C1_DOZEEN_SHIFT))&UART0_C1_DOZEEN_MASK
#define UART0_C1_LOOPS_MASK           0x80u
#define UART0_C1_LOOPS_SHIFT          7
#define UART0_C1_LOOPS_WIDTH          1
#define UART0_C1_LOOPS(x)              (((uint8_t)((uint8_t)(x))<<UART0_C1_LOOPS_SHIFT))&UART0_C1_LOOPS_MASK
/* C2 Bit Fields */
#define UART0_C2_SBK_MASK             0x1u
#define UART0_C2_SBK_SHIFT            0
#define UART0_C2_SBK_WIDTH             1
#define UART0_C2_SBK(x)                (((uint8_t)((uint8_t)(x))<<UART0_C2_SBK_SHIFT))&UART0_C2_SBK_MASK
#define UART0_C2_RWU_MASK              0x2u

```

```

#define UART0_C2_RWU_SHIFT 1
#define UART0_C2_RWU_WIDTH 1
#define UART0_C2_RWU(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_RWU_SHIFT))&UART0_C2_RWU_MASK)
#define UART0_C2_RE_MASK 0x4u
#define UART0_C2_RE_SHIFT 2
#define UART0_C2_RE_WIDTH 1
#define UART0_C2_RE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_RE_SHIFT))&UART0_C2_RE_MASK)
#define UART0_C2_TE_MASK 0x8u
#define UART0_C2_TE_SHIFT 3
#define UART0_C2_TE_WIDTH 1
#define UART0_C2_TE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_TE_SHIFT))&UART0_C2_TE_MASK)
#define UART0_C2_ILIE_MASK 0x10u
#define UART0_C2_ILIE_SHIFT 4
#define UART0_C2_ILIE_WIDTH 1
#define UART0_C2_ILIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_ILIE_SHIFT))&UART0_C2_ILIE_MASK)
#define UART0_C2_RIE_MASK 0x20u
#define UART0_C2_RIE_SHIFT 5
#define UART0_C2_RIE_WIDTH 1
#define UART0_C2_RIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_RIE_SHIFT))&UART0_C2_RIE_MASK)
#define UART0_C2_TCIE_MASK 0x40u
#define UART0_C2_TCIE_SHIFT 6
#define UART0_C2_TCIE_WIDTH 1
#define UART0_C2_TCIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_TCIE_SHIFT))&UART0_C2_TCIE_MASK)
#define UART0_C2_TIE_MASK 0x80u
#define UART0_C2_TIE_SHIFT 7
#define UART0_C2_TIE_WIDTH 1
#define UART0_C2_TIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C2_TIE_SHIFT))&UART0_C2_TIE_MASK)
/* S1 Bit Fields */
#define UART0_S1_PF_MASK 0x1u
#define UART0_S1_PF_SHIFT 0
#define UART0_S1_PF_WIDTH 1
#define UART0_S1_PF(x) (((uint8_t)((uint8_t)(x)<<UART0_S1_PF_SHIFT))&UART0_S1_PF_MASK)
#define UART0_S1_FE_MASK 0x2u
#define UART0_S1_FE_SHIFT 1
#define UART0_S1_FE_WIDTH 1
#define UART0_S1_FE(x) (((uint8_t)((uint8_t)(x)<<UART0_S1_FE_SHIFT))&UART0_S1_FE_MASK)
#define UART0_S1_NF_MASK 0x4u
#define UART0_S1_NF_SHIFT 2
#define UART0_S1_NF_WIDTH 1
#define UART0_S1_NF(x) (((uint8_t)((uint8_t)(x)<<UART0_S1_NF_SHIFT))&UART0_S1_NF_MASK)
#define UART0_S1_OR_MASK 0x8u
#define UART0_S1_OR_SHIFT 3
#define UART0_S1_OR_WIDTH 1

```

```

#define UART0_S1_OR(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S1_OR_SHIFT)) &UART0_S1_OR_MASK)
#define UART0_S1_IDLE_MASK 0x10u
#define UART0_S1_IDLE_SHIFT 4
#define UART0_S1_IDLE_WIDTH 1
#define UART0_S1_IDLE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S1_IDLE_SHIFT)) &UART0_S1_IDLE_MASK)
#define UART0_S1_RDRF_MASK 0x20u
#define UART0_S1_RDRF_SHIFT 5
#define UART0_S1_RDRF_WIDTH 1
#define UART0_S1_RDRF(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S1_RDRF_SHIFT)) &UART0_S1_RDRF_MASK)
#define UART0_S1_TC_MASK 0x40u
#define UART0_S1_TC_SHIFT 6
#define UART0_S1_TC_WIDTH 1
#define UART0_S1_TC(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S1_TC_SHIFT)) &UART0_S1_TC_MASK)
#define UART0_S1_TDRE_MASK 0x80u
#define UART0_S1_TDRE_SHIFT 7
#define UART0_S1_TDRE_WIDTH 1
#define UART0_S1_TDRE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S1_TDRE_SHIFT)) &UART0_S1_TDRE_MASK)
/* S2 Bit Fields */
#define UART0_S2_RAF_MASK 0x1u
#define UART0_S2_RAF_SHIFT 0
#define UART0_S2_RAF_WIDTH 1
#define UART0_S2_RAF(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_RAF_SHIFT)) &UART0_S2_RAF_MASK)
#define UART0_S2_LBKDE_MASK 0x2u
#define UART0_S2_LBKDE_SHIFT 1
#define UART0_S2_LBKDE_WIDTH 1
#define UART0_S2_LBKDE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_LBKDE_SHIFT)) &UART0_S2_LBKDE_MASK)
#define UART0_S2_BRK13_MASK 0x4u
#define UART0_S2_BRK13_SHIFT 2
#define UART0_S2_BRK13_WIDTH 1
#define UART0_S2_BRK13(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_BRK13_SHIFT)) &UART0_S2_BRK13_MASK)
#define UART0_S2_RWUID_MASK 0x8u
#define UART0_S2_RWUID_SHIFT 3
#define UART0_S2_RWUID_WIDTH 1
#define UART0_S2_RWUID(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_RWUID_SHIFT)) &UART0_S2_RWUID_MASK)
#define UART0_S2_RXINV_MASK 0x10u
#define UART0_S2_RXINV_SHIFT 4
#define UART0_S2_RXINV_WIDTH 1
#define UART0_S2_RXINV(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_RXINV_SHIFT)) &UART0_S2_RXINV_MASK)
#define UART0_S2_MSBF_MASK 0x20u
#define UART0_S2_MSBF_SHIFT 5
#define UART0_S2_MSBF_WIDTH 1
#define UART0_S2_MSBF(x)
(((uint8_t) (((uint8_t)(x))<<UART0_S2_MSBF_SHIFT)) &UART0_S2_MSBF_MASK)
#define UART0_S2_RXEDGIF_MASK 0x40u

```

```

#define UART0_S2_RXEDGIF_SHIFT 6
#define UART0_S2_RXEDGIF_WIDTH 1
#define UART0_S2_RXEDGIF(x) (((uint8_t)((uint8_t)(x)<<UART0_S2_RXEDGIF_SHIFT))&UART0_S2_RXEDGIF_MASK)
#define UART0_S2_LBKDIF_MASK 0x80u
#define UART0_S2_LBKDIF_SHIFT 7
#define UART0_S2_LBKDIF_WIDTH 1
#define UART0_S2_LBKDIF(x) (((uint8_t)((uint8_t)(x)<<UART0_S2_LBKDIF_SHIFT))&UART0_S2_LBKDIF_MASK)
/* C3 Bit Fields */
#define UART0_C3_PEIE_MASK 0x1u
#define UART0_C3_PEIE_SHIFT 0
#define UART0_C3_PEIE_WIDTH 1
#define UART0_C3_PEIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_PEIE_SHIFT))&UART0_C3_PEIE_MASK)
#define UART0_C3_FEIE_MASK 0x2u
#define UART0_C3_FEIE_SHIFT 1
#define UART0_C3_FEIE_WIDTH 1
#define UART0_C3_FEIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_FEIE_SHIFT))&UART0_C3_FEIE_MASK)
#define UART0_C3_NEIE_MASK 0x4u
#define UART0_C3_NEIE_SHIFT 2
#define UART0_C3_NEIE_WIDTH 1
#define UART0_C3_NEIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_NEIE_SHIFT))&UART0_C3_NEIE_MASK)
#define UART0_C3_ORIE_MASK 0x8u
#define UART0_C3_ORIE_SHIFT 3
#define UART0_C3_ORIE_WIDTH 1
#define UART0_C3_ORIE(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_ORIE_SHIFT))&UART0_C3_ORIE_MASK)
#define UART0_C3_TXINV_MASK 0x10u
#define UART0_C3_TXINV_SHIFT 4
#define UART0_C3_TXINV_WIDTH 1
#define UART0_C3_TXINV(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_TXINV_SHIFT))&UART0_C3_TXINV_MASK)
#define UART0_C3_TXDIR_MASK 0x20u
#define UART0_C3_TXDIR_SHIFT 5
#define UART0_C3_TXDIR_WIDTH 1
#define UART0_C3_TXDIR(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_TXDIR_SHIFT))&UART0_C3_TXDIR_MASK)
#define UART0_C3_R9T8_MASK 0x40u
#define UART0_C3_R9T8_SHIFT 6
#define UART0_C3_R9T8_WIDTH 1
#define UART0_C3_R9T8(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_R9T8_SHIFT))&UART0_C3_R9T8_MASK)
#define UART0_C3_R8T9_MASK 0x80u
#define UART0_C3_R8T9_SHIFT 7
#define UART0_C3_R8T9_WIDTH 1
#define UART0_C3_R8T9(x) (((uint8_t)((uint8_t)(x)<<UART0_C3_R8T9_SHIFT))&UART0_C3_R8T9_MASK)
/* D Bit Fields */
#define UART0_D_R0T0_MASK 0x1u
#define UART0_D_R0T0_SHIFT 0

```

```

#define UART0_D_R0T0_WIDTH 1
#define UART0_D_R0T0(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R0T0_SHIFT))&UART0_D_R0T0_MASK
#define UART0_D_R1T1_MASK 0x2u
#define UART0_D_R1T1_SHIFT 1
#define UART0_D_R1T1_WIDTH 1
#define UART0_D_R1T1(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R1T1_SHIFT))&UART0_D_R1T1_MASK
#define UART0_D_R2T2_MASK 0x4u
#define UART0_D_R2T2_SHIFT 2
#define UART0_D_R2T2_WIDTH 1
#define UART0_D_R2T2(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R2T2_SHIFT))&UART0_D_R2T2_MASK
#define UART0_D_R3T3_MASK 0x8u
#define UART0_D_R3T3_SHIFT 3
#define UART0_D_R3T3_WIDTH 1
#define UART0_D_R3T3(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R3T3_SHIFT))&UART0_D_R3T3_MASK
#define UART0_D_R4T4_MASK 0x10u
#define UART0_D_R4T4_SHIFT 4
#define UART0_D_R4T4_WIDTH 1
#define UART0_D_R4T4(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R4T4_SHIFT))&UART0_D_R4T4_MASK
#define UART0_D_R5T5_MASK 0x20u
#define UART0_D_R5T5_SHIFT 5
#define UART0_D_R5T5_WIDTH 1
#define UART0_D_R5T5(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R5T5_SHIFT))&UART0_D_R5T5_MASK
#define UART0_D_R6T6_MASK 0x40u
#define UART0_D_R6T6_SHIFT 6
#define UART0_D_R6T6_WIDTH 1
#define UART0_D_R6T6(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R6T6_SHIFT))&UART0_D_R6T6_MASK
#define UART0_D_R7T7_MASK 0x80u
#define UART0_D_R7T7_SHIFT 7
#define UART0_D_R7T7_WIDTH 1
#define UART0_D_R7T7(x) (((uint8_t)((uint8_t)(x))<<UART0_D_R7T7_SHIFT))&UART0_D_R7T7_MASK
/* MA1 Bit Fields */
#define UART0_MA1_MA_MASK 0xFFu
#define UART0_MA1_MA_SHIFT 0
#define UART0_MA1_MA_WIDTH 8
#define UART0_MA1_MA(x) (((uint8_t)((uint8_t)(x))<<UART0_MA1_MA_SHIFT))&UART0_MA1_MA_MASK
/* MA2 Bit Fields */
#define UART0_MA2_MA_MASK 0xFFu
#define UART0_MA2_MA_SHIFT 0
#define UART0_MA2_MA_WIDTH 8
#define UART0_MA2_MA(x) (((uint8_t)((uint8_t)(x))<<UART0_MA2_MA_SHIFT))&UART0_MA2_MA_MASK
/* C4 Bit Fields */
#define UART0_C4_OSR_MASK 0x1Fu
#define UART0_C4_OSR_SHIFT 0
#define UART0_C4_OSR_WIDTH 5

```

```

#define UART0_C4_OSR(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C4_OSR_SHIFT)) &UART0_C4_OSR_MASK)
#define UART0_C4_M10_MASK 0x20u
#define UART0_C4_M10_SHIFT 5
#define UART0_C4_M10_WIDTH 1
#define UART0_C4_M10(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C4_M10_SHIFT)) &UART0_C4_M10_MASK)
#define UART0_C4_MAEN2_MASK 0x40u
#define UART0_C4_MAEN2_SHIFT 6
#define UART0_C4_MAEN2_WIDTH 1
#define UART0_C4_MAEN2(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C4_MAEN2_SHIFT)) &UART0_C4_MAEN2_MASK)
#define UART0_C4_MAEN1_MASK 0x80u
#define UART0_C4_MAEN1_SHIFT 7
#define UART0_C4_MAEN1_WIDTH 1
#define UART0_C4_MAEN1(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C4_MAEN1_SHIFT)) &UART0_C4_MAEN1_MASK)
/* C5 Bit Fields */
#define UART0_C5_RESETDISC_MASK 0x1u
#define UART0_C5_RESETDISC_SHIFT 0
#define UART0_C5_RESETDISC_WIDTH 1
#define UART0_C5_RESETDISC(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C5_RESETDISC_SHIFT)) &UART0_C5_RESETDISC_MASK)
#define UART0_C5_BOTHEDGE_MASK 0x2u
#define UART0_C5_BOTHEDGE_SHIFT 1
#define UART0_C5_BOTHEDGE_WIDTH 1
#define UART0_C5_BOTHEDGE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C5_BOTHEDGE_SHIFT)) &UART0_C5_BOTHEDGE_MASK)
#define UART0_C5_RDMAE_MASK 0x20u
#define UART0_C5_RDMAE_SHIFT 5
#define UART0_C5_RDMAE_WIDTH 1
#define UART0_C5_RDMAE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C5_RDMAE_SHIFT)) &UART0_C5_RDMAE_MASK)
#define UART0_C5_TDMAE_MASK 0x80u
#define UART0_C5_TDMAE_SHIFT 7
#define UART0_C5_TDMAE_WIDTH 1
#define UART0_C5_TDMAE(x)
(((uint8_t) (((uint8_t)(x))<<UART0_C5_TDMAE_SHIFT)) &UART0_C5_TDMAE_MASK)

/* !
 * @}
 */ /* end of group UART0_Register_Masks */

/* UART0 - Peripheral instance base addresses */
/** Peripheral UART0 base address */
#define UART0_BASE (0x4006A000u)
/** Peripheral UART0 base pointer */
#define UART0 ((UART0_Type
*)UART0_BASE)
#define UART0_BASE_PTR (UART0)
/** Array initializer of UART0 peripheral base addresses */

```

```

#define UART0_BASE_ADDRS           { UART0_BASE }
/** Array initializer of UART0 peripheral base pointers */
#define UART0_BASE_PTRS            { UART0 }
/** Interrupt vectors for the UART0 peripheral type */
#define UART0_RX_TX IRQS          { UART0_IRQn }
#define UART0_ERR IRQS             { UART0_IRQn }

/* -----
-- UART0 - Register accessor macros
-----
*/
/*!
* @addtogroup UART0_Register_Accessor_Macros UART0 - Register accessor
macros
* @{
*/
/* UART0 - Register instance definitions */
/* UART0 */
#define UART0_BDH                 UART0_BDH_REG(UART0)
#define UART0_BDL                 UART0_BDL_REG(UART0)
#define UART0_C1                  UART0_C1_REG(UART0)
#define UART0_C2                  UART0_C2_REG(UART0)
#define UART0_S1                  UART0_S1_REG(UART0)
#define UART0_S2                  UART0_S2_REG(UART0)
#define UART0_C3                  UART0_C3_REG(UART0)
#define UART0_D                   UART0_D_REG(UART0)
#define UART0_MA1                 UART0_MA1_REG(UART0)
#define UART0_MA2                 UART0_MA2_REG(UART0)
#define UART0_C4                  UART0_C4_REG(UART0)
#define UART0_C5                  UART0_C5_REG(UART0)

/*!
* @}
*/
/* end of group UART0_Register_Accessor_Macros */

/*!
* @}
*/
/* end of group UART0_Peripheral_Access_Layer */

/* -----
-- USB Peripheral Access Layer
-----
*/
/*!
* @addtogroup USB_Peripheral_Access_Layer USB Peripheral Access Layer
* @{

```

```

*/
/** USB - Register Layout Typedef */
typedef struct {
    __I uint8_t PERID;                                /**< Peripheral ID
register, offset: 0x0 */
    uint8_t RESERVED_0[3];
    __I uint8_t IDCOMP;                                /**< Peripheral ID
Complement register, offset: 0x4 */
    uint8_t RESERVED_1[3];
    __I uint8_t REV;                                    /**< Peripheral
Revision register, offset: 0x8 */
    uint8_t RESERVED_2[3];
    __I uint8_t ADDINFO;                               /**< Peripheral
Additional Info register, offset: 0xC */
    uint8_t RESERVED_3[3];
    __IO uint8_t OTGISTAT;                            /**< OTG Interrupt
Status register, offset: 0x10 */
    uint8_t RESERVED_4[3];
    __IO uint8_t OTGICR;                               /**< OTG Interrupt
Control Register, offset: 0x14 */
    uint8_t RESERVED_5[3];
    __IO uint8_t OTGSTAT;                            /**< OTG Status
register, offset: 0x18 */
    uint8_t RESERVED_6[3];
    __IO uint8_t OTGCTL;                               /**< OTG Control
register, offset: 0x1C */
    uint8_t RESERVED_7[99];
    __IO uint8_t ISTAT;                               /**< Interrupt Status
register, offset: 0x80 */
    uint8_t RESERVED_8[3];
    __IO uint8_t INTEN;                               /**< Interrupt Enable
register, offset: 0x84 */
    uint8_t RESERVED_9[3];
    __IO uint8_t ERRSTAT;                            /**< Error Interrupt
Status register, offset: 0x88 */
    uint8_t RESERVED_10[3];
    __IO uint8_t ERREN;                               /**< Error Interrupt
Enable register, offset: 0x8C */
    uint8_t RESERVED_11[3];
    __I uint8_t STAT;                                /**< Status register,
offset: 0x90 */
    uint8_t RESERVED_12[3];
    __IO uint8_t CTL;                                /**< Control register,
offset: 0x94 */
    uint8_t RESERVED_13[3];
    __IO uint8_t ADDR;                               /**< Address register,
offset: 0x98 */
    uint8_t RESERVED_14[3];
    __IO uint8_t BDTPAGE1;                            /**< BDT Page Register
1, offset: 0x9C */
    uint8_t RESERVED_15[3];
    __IO uint8_t FRMNUML;                            /**< Frame Number
Register Low, offset: 0xA0 */
}

```

```

        uint8_t RESERVED_16[3];
    __IO uint8_t FRMNUMH;                                /**< Frame Number
Register High, offset: 0xA4 */
        uint8_t RESERVED_17[3];
    __IO uint8_t TOKEN;                                 /**< Token register,
offset: 0xA8 */
        uint8_t RESERVED_18[3];
    __IO uint8_t SOFTHLD;                               /**< SOF Threshold
Register, offset: 0xAC */
        uint8_t RESERVED_19[3];
    __IO uint8_t BDTPAGE2;                            /**< BDT Page Register
2, offset: 0xB0 */
        uint8_t RESERVED_20[3];
    __IO uint8_t BDTPAGE3;                            /**< BDT Page Register
3, offset: 0xB4 */
        uint8_t RESERVED_21[11];
    struct {                                         /* offset: 0xC0, array
step: 0x4 */
        __IO uint8_t ENDPT;                           /**< Endpoint
Control register, array offset: 0xC0, array step: 0x4 */
            uint8_t RESERVED_0[3];
        } ENDPOINT[16];
        __IO uint8_t USBCTRL;                         /**< USB Control
register, offset: 0x100 */
            uint8_t RESERVED_22[3];
        I uint8_t OBSERVE;                          /**< USB OTG Observe
register, offset: 0x104 */
            uint8_t RESERVED_23[3];
        __IO uint8_t CONTROL;                        /**< USB OTG Control
register, offset: 0x108 */
            uint8_t RESERVED_24[3];
        __IO uint8_t USBTRC0;                         /**< USB Transceiver
Control Register 0, offset: 0x10C */
            uint8_t RESERVED_25[7];
        __IO uint8_t USBFRMADJUST;                   /**< Frame Adjust
Register, offset: 0x114 */
    } USB_Type, *USB_MemMapPtr;

```

```

/* -----
-- USB - Register accessor macros
-----
*/

```

```

/*!
 * @addtogroup USB_Register_Accessor_Macros USB - Register accessor
macros
 * @{
 */

```

```

/* USB - Register accessors */
#define USB_PERID_REG(base)          ((base)->PERID)
#define USB_IDCOMP_REG(base)         ((base)->IDCOMP)

```

```

#define USB_REV_REG(base)                                ((base) ->REV)
#define USB_ADDINFO_REG(base)                            ((base) ->ADDINFO)
#define USB_OTGISTAT_REG(base)                          ((base) ->OTGISTAT)
#define USB_OTGICR_REG(base)                            ((base) ->OTGICR)
#define USB_OTGSTAT_REG(base)                           ((base) ->OTGSTAT)
#define USB_OTGCTL_REG(base)                            ((base) ->OTGCTL)
#define USB_ISTAT_REG(base)                             ((base) ->ISTAT)
#define USB_INTEN_REG(base)                            ((base) ->INTEN)
#define USB_ERRSTAT_REG(base)                           ((base) ->ERRSTAT)
#define USB_ERREN_REG(base)                            ((base) ->ERREN)
#define USB_STAT_REG(base)                             ((base) ->STAT)
#define USB_CTL_REG(base)                             ((base) ->CTL)
#define USB_ADDR_REG(base)                            ((base) ->ADDR)
#define USB_BDTPAGE1_REG(base)                         ((base) ->BDTPAGE1)
#define USB_FRMNUML_REG(base)                           ((base) ->FRMNUML)
#define USB_FRMNUMH_REG(base)                           ((base) ->FRMNUMH)
#define USB_TOKEN_REG(base)                            ((base) ->TOKEN)
#define USB_SOFTHLD_REG(base)                           ((base) ->SOFTHLD)
#define USB_BDTPAGE2_REG(base)                         ((base) ->BDTPAGE2)
#define USB_BDTPAGE3_REG(base)                         ((base) ->BDTPAGE3)
#define USB_ENDPT_REG(base, index)                     ((base) -
>ENDPOINT[index].ENDPT)
#define USB_ENDP_COUNT                                16
#define USB_USBCTRL_REG(base)                          ((base) ->USBCTRL)
#define USB_OBSERVE_REG(base)                          ((base) ->OBSERVE)
#define USB_CONTROL_REG(base)                          ((base) ->CONTROL)
#define USB_USBTRC0_REG(base)                          ((base) ->USBTRC0)
#define USB_USBFRMADJUST_REG(base)                    ((base) ->USBFRMADJUST)

/* !
 * @}
 */
/* /* end of group USB_Register_Accessor_Macros */


```

```

/* -----
-- USB Register Masks
-----
*/
/* !
 * @addtogroup USB_Register_Masks USB Register Masks
 * @{
 */

/* PERID Bit Fields */
#define USB_PERID_ID_MASK                           0x3Fu
#define USB_PERID_ID_SHIFT                          0
#define USB_PERID_ID_WIDTH                          6
#define USB_PERID_ID(x)                            (((uint8_t)((uint8_t)(x))<<USB_PERID_ID_SHIFT)) &USB_PERID_ID_MASK)
/* IDCMP Bit Fields */
#define USB_IDCOMP_NID_MASK                          0x3Fu
#define USB_IDCOMP_NID_SHIFT                         0


```

```

#define USB_IDCOMP_NID_WIDTH 6
#define USB_IDCOMP_NID(x) (((uint8_t)((uint8_t)(x))<<USB_IDCOMP_NID_SHIFT))&USB_IDCOMP_NID_MASK)
/* REV Bit Fields */
#define USB_REV_REV_MASK 0xFFu
#define USB_REV_REV_SHIFT 0
#define USB_REV_REV_WIDTH 8
#define USB_REV_REV(x) (((uint8_t)((uint8_t)(x))<<USB_REV_REV_SHIFT))&USB_REV_REV_MASK)
/* ADDINFO Bit Fields */
#define USB_ADDINFO_IEHOST_MASK 0x1u
#define USB_ADDINFO_IEHOST_SHIFT 0
#define USB_ADDINFO_IEHOST_WIDTH 1
#define USB_ADDINFO_IEHOST(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IEHOST_SHIFT))&USB_ADDINFO_IEHOST_MASK)
#define USB_ADDINFO_IRQNUM_MASK 0xF8u
#define USB_ADDINFO_IRQNUM_SHIFT 3
#define USB_ADDINFO_IRQNUM_WIDTH 5
#define USB_ADDINFO_IRQNUM(x) (((uint8_t)((uint8_t)(x))<<USB_ADDINFO_IRQNUM_SHIFT))&USB_ADDINFO_IRQNUM_MASK)
/* OTGISTAT Bit Fields */
#define USB_OTGISTAT_AVBUSCHG_MASK 0x1u
#define USB_OTGISTAT_AVBUSCHG_SHIFT 0
#define USB_OTGISTAT_AVBUSCHG_WIDTH 1
#define USB_OTGISTAT_AVBUSCHG(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_AVBUSCHG_SHIFT))&USB_OTGISTAT_AVBUSCHG_MASK)
#define USB_OTGISTAT_B_SESS_CHG_MASK 0x4u
#define USB_OTGISTAT_B_SESS_CHG_SHIFT 2
#define USB_OTGISTAT_B_SESS_CHG_WIDTH 1
#define USB_OTGISTAT_B_SESS_CHG(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_B_SESS_CHG_SHIFT))&USB_OTGISTAT_B_SESS_CHG_MASK)
#define USB_OTGISTAT_SESSVLDCHG_MASK 0x8u
#define USB_OTGISTAT_SESSVLDCHG_SHIFT 3
#define USB_OTGISTAT_SESSVLDCHG_WIDTH 1
#define USB_OTGISTAT_SESSVLDCHG(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_SESSVLDCHG_SHIFT))&USB_OTGISTAT_SESSVLDCHG_MASK)
#define USB_OTGISTAT_LINE_STATE_CHG_MASK 0x20u
#define USB_OTGISTAT_LINE_STATE_CHG_SHIFT 5
#define USB_OTGISTAT_LINE_STATE_CHG_WIDTH 1
#define USB_OTGISTAT_LINE_STATE_CHG(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_LINE_STATE_CHG_SHIFT))&USB_OTGISTAT_LINE_STATE_CHG_MASK)
#define USB_OTGISTAT_ONEMSEC_MASK 0x40u
#define USB_OTGISTAT_ONEMSEC_SHIFT 6
#define USB_OTGISTAT_ONEMSEC_WIDTH 1
#define USB_OTGISTAT_ONEMSEC(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_ONEMSEC_SHIFT))&USB_OTGISTAT_ONE_MSEC_MASK)
#define USB_OTGISTAT_IDCHG_MASK 0x80u

```

```

#define USB_OTGISTAT_IDCHG_SHIFT 7
#define USB_OTGISTAT_IDCHG_WIDTH 1
#define USB_OTGISTAT_IDCHG(x) (((uint8_t)((uint8_t)(x))<<USB_OTGISTAT_IDCHG_SHIFT))&USB_OTGISTAT_IDCHG_MASK
/* OTGICR Bit Fields */
#define USB_OTGICR_AVBUSEN_MASK 0x1u
#define USB_OTGICR_AVBUSEN_SHIFT 0
#define USB_OTGICR_AVBUSEN_WIDTH 1
#define USB_OTGICR_AVBUSEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_AVBUSEN_SHIFT))&USB_OTGICR_AVBUSEN_MASK
#define USB_OTGICR_BSESSEN_MASK 0x4u
#define USB_OTGICR_BSESSEN_SHIFT 2
#define USB_OTGICR_BSESSEN_WIDTH 1
#define USB_OTGICR_BSESSEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_BSESSEN_SHIFT))&USB_OTGICR_BSESSEN_MASK
#define USB_OTGICR_SESSVLDEN_MASK 0x8u
#define USB_OTGICR_SESSVLDEN_SHIFT 3
#define USB_OTGICR_SESSVLDEN_WIDTH 1
#define USB_OTGICR_SESSVLDEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_SESSVLDEN_SHIFT))&USB_OTGICR_SESSVLDEN_MASK
#define USB_OTGICR_LINESTATEEN_MASK 0x20u
#define USB_OTGICR_LINESTATEEN_SHIFT 5
#define USB_OTGICR_LINESTATEEN_WIDTH 1
#define USB_OTGICR_LINESTATEEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_LINESTATEEN_SHIFT))&USB_OTGICR_LINESTATEEN_MASK
#define USB_OTGICR_ONEMSECEN_MASK 0x40u
#define USB_OTGICR_ONEMSECEN_SHIFT 6
#define USB_OTGICR_ONEMSECEN_WIDTH 1
#define USB_OTGICR_ONEMSECEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_ONEMSECEN_SHIFT))&USB_OTGICR_ONEMSECEN_MASK
#define USB_OTGICR_IDEN_MASK 0x80u
#define USB_OTGICR_IDEN_SHIFT 7
#define USB_OTGICR_IDEN_WIDTH 1
#define USB_OTGICR_IDEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGICR_IDEN_SHIFT))&USB_OTGICR_IDEN_MASK
/* OTGSTAT Bit Fields */
#define USB_OTGSTAT_AVBUSVLD_MASK 0x1u
#define USB_OTGSTAT_AVBUSVLD_SHIFT 0
#define USB_OTGSTAT_AVBUSVLD_WIDTH 1
#define USB_OTGSTAT_AVBUSVLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_AVBUSVLD_SHIFT))&USB_OTGSTAT_AVBUSVLD_MASK
#define USB_OTGSTAT_BSESEND_MASK 0x4u
#define USB_OTGSTAT_BSESEND_SHIFT 2
#define USB_OTGSTAT_BSESEND_WIDTH 1
#define USB_OTGSTAT_BSESEND(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_BSESEND_SHIFT))&USB_OTGSTAT_BSESEND_MASK

```

```

#define USB_OTGSTAT_SESS_VLD_MASK 0x8u
#define USB_OTGSTAT_SESS_VLD_SHIFT 3
#define USB_OTGSTAT_SESS_VLD_WIDTH 1
#define USB_OTGSTAT_SESS_VLD(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_SESS_VLD_SHIFT)&USB_OTGSTAT_SESS_VLD_MASK)
#define USB_OTGSTAT_LINESTATESTABLE_MASK 0x20u
#define USB_OTGSTAT_LINESTATESTABLE_SHIFT 5
#define USB_OTGSTAT_LINESTATESTABLE_WIDTH 1
#define USB_OTGSTAT_LINESTATESTABLE(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_LINESTATESTABLE_SHIFT)&USB_OTGSTAT_LINESTATESTABLE_MASK)
#define USB_OTGSTAT_ONEMSECEN_MASK 0x40u
#define USB_OTGSTAT_ONEMSECEN_SHIFT 6
#define USB_OTGSTAT_ONEMSECEN_WIDTH 1
#define USB_OTGSTAT_ONEMSECEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ONEMSECEN_SHIFT)&USB_OTGSTAT_ONEMSECEN_MASK)
#define USB_OTGSTAT_ID_MASK 0x80u
#define USB_OTGSTAT_ID_SHIFT 7
#define USB_OTGSTAT_ID_WIDTH 1
#define USB_OTGSTAT_ID(x) (((uint8_t)((uint8_t)(x))<<USB_OTGSTAT_ID_SHIFT)&USB_OTGSTAT_ID_MASK)
/* OTGCTL Bit Fields */
#define USB_OTGCTL_OTGEN_MASK 0x4u
#define USB_OTGCTL_OTGEN_SHIFT 2
#define USB_OTGCTL_OTGEN_WIDTH 1
#define USB_OTGCTL_OTGEN(x) (((uint8_t)((uint8_t)(x))<<USB_OTGCTL_OTGEN_SHIFT)&USB_OTGCTL_OTGEN_MASK)
#define USB_OTGCTL_DMLOW_MASK 0x10u
#define USB_OTGCTL_DMLOW_SHIFT 4
#define USB_OTGCTL_DMLOW_WIDTH 1
#define USB_OTGCTL_DMLOW(x) (((uint8_t)((uint8_t)(x))<<USB_OTGCTL_DMLOW_SHIFT)&USB_OTGCTL_DMLOW_MASK)
#define USB_OTGCTL_DPLOW_MASK 0x20u
#define USB_OTGCTL_DPLOW_SHIFT 5
#define USB_OTGCTL_DPLOW_WIDTH 1
#define USB_OTGCTL_DPLOW(x) (((uint8_t)((uint8_t)(x))<<USB_OTGCTL_DPLOW_SHIFT)&USB_OTGCTL_DPLOW_MASK)
#define USB_OTGCTL_DPHIGH_MASK 0x80u
#define USB_OTGCTL_DPHIGH_SHIFT 7
#define USB_OTGCTL_DPHIGH_WIDTH 1
#define USB_OTGCTL_DPHIGH(x) (((uint8_t)((uint8_t)(x))<<USB_OTGCTL_DPHIGH_SHIFT)&USB_OTGCTL_DPHIGH_MASK)
/* ISTAT Bit Fields */
#define USB_ISTAT_USBRST_MASK 0x1u
#define USB_ISTAT_USBRST_SHIFT 0
#define USB_ISTAT_USBRST_WIDTH 1

```

```

#define USB_ISTAT_USBRST(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_USBRST_SHIFT))&USB_ISTAT_USBRST_MASK
#define USB_ISTAT_ERROR_MASK 0x2u
#define USB_ISTAT_ERROR_SHIFT 1
#define USB_ISTAT_ERROR_WIDTH 1
#define USB_ISTAT_ERROR(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_ERROR_SHIFT))&USB_ISTAT_ERROR_MASK
#define USB_ISTAT_SOFTOK_MASK 0x4u
#define USB_ISTAT_SOFTOK_SHIFT 2
#define USB_ISTAT_SOFTOK_WIDTH 1
#define USB_ISTAT_SOFTOK(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_SOFTOK_SHIFT))&USB_ISTAT_SOFTOK_MASK
#define USB_ISTAT_TOKDNE_MASK 0x8u
#define USB_ISTAT_TOKDNE_SHIFT 3
#define USB_ISTAT_TOKDNE_WIDTH 1
#define USB_ISTAT_TOKDNE(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_TOKDNE_SHIFT))&USB_ISTAT_TOKDNE_MASK
#define USB_ISTAT_SLEEP_MASK 0x10u
#define USB_ISTAT_SLEEP_SHIFT 4
#define USB_ISTAT_SLEEP_WIDTH 1
#define USB_ISTAT_SLEEP(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_SLEEP_SHIFT))&USB_ISTAT_SLEEP_MASK
#define USB_ISTAT_RESUME_MASK 0x20u
#define USB_ISTAT_RESUME_SHIFT 5
#define USB_ISTAT_RESUME_WIDTH 1
#define USB_ISTAT_RESUME(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_RESUME_SHIFT))&USB_ISTAT_RESUME_MASK
#define USB_ISTAT_ATTACH_MASK 0x40u
#define USB_ISTAT_ATTACH_SHIFT 6
#define USB_ISTAT_ATTACH_WIDTH 1
#define USB_ISTAT_ATTACH(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_ATTACH_SHIFT))&USB_ISTAT_ATTACH_MASK
#define USB_ISTAT_STALL_MASK 0x80u
#define USB_ISTAT_STALL_SHIFT 7
#define USB_ISTAT_STALL_WIDTH 1
#define USB_ISTAT_STALL(x)
(((uint8_t)((uint8_t)(x))<<USB_ISTAT_STALL_SHIFT))&USB_ISTAT_STALL_MASK
/* INTEN Bit Fields */
#define USB_INTEN_USBRSTEN_MASK 0x1u
#define USB_INTEN_USBRSTEN_SHIFT 0
#define USB_INTEN_USBRSTEN_WIDTH 1
#define USB_INTEN_USBRSTEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_USBRSTEN_SHIFT))&USB_INTEN_USBRSTEN_MASK
#define USB_INTEN_ERROREN_MASK 0x2u
#define USB_INTEN_ERROREN_SHIFT 1
#define USB_INTEN_ERROREN_WIDTH 1

```

```

#define USB_INTEN_ERROREN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_ERROREN_SHIFT))&USB_INTEN_ERROREN_M
ASK)

#define USB_INTEN_SOFTOKEN_MASK 0x4u
#define USB_INTEN_SOFTOKEN_SHIFT 2
#define USB_INTEN_SOFTOKEN_WIDTH 1
#define USB_INTEN_SOFTOKEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_SOFTOKEN_SHIFT))&USB_INTEN_SOFTOKEN_
MASK)

#define USB_INTEN_TOKDNEEN_MASK 0x8u
#define USB_INTEN_TOKDNEEN_SHIFT 3
#define USB_INTEN_TOKDNEEN_WIDTH 1
#define USB_INTEN_TOKDNEEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_TOKDNEEN_SHIFT))&USB_INTEN_TOKDNEEN_
MASK)

#define USB_INTEN_SLEEPEN_MASK 0x10u
#define USB_INTEN_SLEEPEN_SHIFT 4
#define USB_INTEN_SLEEPEN_WIDTH 1
#define USB_INTEN_SLEEPEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_SLEEPEN_SHIFT))&USB_INTEN_SLEEPEN_M
ASK)

#define USB_INTEN_RESUMEEN_MASK 0x20u
#define USB_INTEN_RESUMEEN_SHIFT 5
#define USB_INTEN_RESUMEEN_WIDTH 1
#define USB_INTEN_RESUMEEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_RESUMEEN_SHIFT))&USB_INTEN_RESUMEEN_
MASK)

#define USB_INTEN_ATTACHEN_MASK 0x40u
#define USB_INTEN_ATTACHEN_SHIFT 6
#define USB_INTEN_ATTACHEN_WIDTH 1
#define USB_INTEN_ATTACHEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_ATTACHEN_SHIFT))&USB_INTEN_ATTACHEN_
MASK)

#define USB_INTEN_STALLEN_MASK 0x80u
#define USB_INTEN_STALLEN_SHIFT 7
#define USB_INTEN_STALLEN_WIDTH 1
#define USB_INTEN_STALLEN(x)
(((uint8_t)((uint8_t)(x))<<USB_INTEN_STALLEN_SHIFT))&USB_INTEN_STALLEN_M
ASK)

/* ERRSTAT Bit Fields */
#define USB_ERRSTAT_PIDERR_MASK 0x1u
#define USB_ERRSTAT_PIDERR_SHIFT 0
#define USB_ERRSTAT_PIDERR_WIDTH 1
#define USB_ERRSTAT_PIDERR(x)
(((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_PIDERR_SHIFT))&USB_ERRSTAT_PIDERR_
MASK)

#define USB_ERRSTAT_CRC5EOF_MASK 0x2u
#define USB_ERRSTAT_CRC5EOF_SHIFT 1
#define USB_ERRSTAT_CRC5EOF_WIDTH 1
#define USB_ERRSTAT_CRC5EOF(x)
(((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_CRC5EOF_SHIFT))&USB_ERRSTAT_CRC5E
OF_MASK)

#define USB_ERRSTAT_CRC16_MASK 0x4u
#define USB_ERRSTAT_CRC16_SHIFT 2

```

```

#define USB_ERRSTAT_CRC16_WIDTH 1
#define USB_ERRSTAT_CRC16(x) (((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_CRC16_SHIFT))&USB_ERRSTAT_CRC16_M
ASK)
#define USB_ERRSTAT_DFN8_MASK 0x8u
#define USB_ERRSTAT_DFN8_SHIFT 3
#define USB_ERRSTAT_DFN8_WIDTH 1
#define USB_ERRSTAT_DFN8(x) (((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_DFN8_SHIFT))&USB_ERRSTAT_DFN8_M
ASK)
#define USB_ERRSTAT_BTOERR_MASK 0x10u
#define USB_ERRSTAT_BTOERR_SHIFT 4
#define USB_ERRSTAT_BTOERR_WIDTH 1
#define USB_ERRSTAT_BTOERR(x) (((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_BTOERR_SHIFT))&USB_ERRSTAT_BTOERR_M
ASK)
#define USB_ERRSTAT_DMAERR_MASK 0x20u
#define USB_ERRSTAT_DMAERR_SHIFT 5
#define USB_ERRSTAT_DMAERR_WIDTH 1
#define USB_ERRSTAT_DMAERR(x) (((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_DMAERR_SHIFT))&USB_ERRSTAT_DMAERR_M
ASK)
#define USB_ERRSTAT_BTSERR_MASK 0x80u
#define USB_ERRSTAT_BTSERR_SHIFT 7
#define USB_ERRSTAT_BTSERR_WIDTH 1
#define USB_ERRSTAT_BTSERR(x) (((uint8_t)((uint8_t)(x))<<USB_ERRSTAT_BTSERR_SHIFT))&USB_ERRSTAT_BTSERR_M
ASK)
/* ERREN Bit Fields */
#define USB_ERREN_PIDERREN_MASK 0x1u
#define USB_ERREN_PIDERREN_SHIFT 0
#define USB_ERREN_PIDERREN_WIDTH 1
#define USB_ERREN_PIDERREN(x) (((uint8_t)((uint8_t)(x))<<USB_ERREN_PIDERREN_SHIFT))&USB_ERREN_PIDERREN_M
ASK)
#define USB_ERREN_CRC5EOFEN_MASK 0x2u
#define USB_ERREN_CRC5EOFEN_SHIFT 1
#define USB_ERREN_CRC5EOFEN_WIDTH 1
#define USB_ERREN_CRC5EOFEN(x) (((uint8_t)((uint8_t)(x))<<USB_ERREN_CRC5EOFEN_SHIFT))&USB_ERREN_CRC5EOF
EN_M
ASK)
#define USB_ERREN_CRC16EN_MASK 0x4u
#define USB_ERREN_CRC16EN_SHIFT 2
#define USB_ERREN_CRC16EN_WIDTH 1
#define USB_ERREN_CRC16EN(x) (((uint8_t)((uint8_t)(x))<<USB_ERREN_CRC16EN_SHIFT))&USB_ERREN_CRC16EN_M
ASK)
#define USB_ERREN_DFN8EN_MASK 0x8u
#define USB_ERREN_DFN8EN_SHIFT 3
#define USB_ERREN_DFN8EN_WIDTH 1
#define USB_ERREN_DFN8EN(x) (((uint8_t)((uint8_t)(x))<<USB_ERREN_DFN8EN_SHIFT))&USB_ERREN_DFN8EN_M
ASK)
#define USB_ERREN_BTOERREN_MASK 0x10u

```

```

#define USB_ERREN_BTOERREN_SHIFT 4
#define USB_ERREN_BTOERREN_WIDTH 1
#define USB_ERREN_BTOERREN(x) (((uint8_t)((uint8_t)(x)<<USB_ERREN_BTOERREN_SHIFT))&USB_ERREN_BTOERREN_MASK)
#define USB_ERREN_DMAERREN_MASK 0x20u
#define USB_ERREN_DMAERREN_SHIFT 5
#define USB_ERREN_DMAERREN_WIDTH 1
#define USB_ERREN_DMAERREN(x) (((uint8_t)((uint8_t)(x)<<USB_ERREN_DMAERREN_SHIFT))&USB_ERREN_DMAERREN_MASK)
#define USB_ERREN_BTSERREN_MASK 0x80u
#define USB_ERREN_BTSERREN_SHIFT 7
#define USB_ERREN_BTSERREN_WIDTH 1
#define USB_ERREN_BTSERREN(x) (((uint8_t)((uint8_t)(x)<<USB_ERREN_BTSERREN_SHIFT))&USB_ERREN_BTSERREN_MASK)
/* STAT Bit Fields */
#define USB_STAT_ODD_MASK 0x4u
#define USB_STAT_ODD_SHIFT 2
#define USB_STAT_ODD_WIDTH 1
#define USB_STAT_ODD(x) (((uint8_t)((uint8_t)(x)<<USB_STAT_ODD_SHIFT))&USB_STAT_ODD_MASK)
#define USB_STAT_TX_MASK 0x8u
#define USB_STAT_TX_SHIFT 3
#define USB_STAT_TX_WIDTH 1
#define USB_STAT_TX(x) (((uint8_t)((uint8_t)(x)<<USB_STAT_TX_SHIFT))&USB_STAT_TX_MASK)
#define USB_STAT_ENDP_MASK 0xF0u
#define USB_STAT_ENDP_SHIFT 4
#define USB_STAT_ENDP_WIDTH 4
#define USB_STAT_ENDP(x) (((uint8_t)((uint8_t)(x)<<USB_STAT_ENDP_SHIFT))&USB_STAT_ENDP_MASK)
/* CTL Bit Fields */
#define USB_CTL_USBENSOSEN_MASK 0x1u
#define USB_CTL_USBENSOSEN_SHIFT 0
#define USB_CTL_USBENSOSEN_WIDTH 1
#define USB_CTL_USBENSOSEN(x) (((uint8_t)((uint8_t)(x)<<USB_CTL_USBENSOSEN_SHIFT))&USB_CTL_USBENSOSEN_MASK)
#define USB_CTL_ODDRST_MASK 0x2u
#define USB_CTL_ODDRST_SHIFT 1
#define USB_CTL_ODDRST_WIDTH 1
#define USB_CTL_ODDRST(x) (((uint8_t)((uint8_t)(x)<<USB_CTL_ODDRST_SHIFT))&USB_CTL_ODDRST_MASK)
#define USB_CTL_RESUME_MASK 0x4u
#define USB_CTL_RESUME_SHIFT 2
#define USB_CTL_RESUME_WIDTH 1
#define USB_CTL_RESUME(x) (((uint8_t)((uint8_t)(x)<<USB_CTL_RESUME_SHIFT))&USB_CTL_RESUME_MASK)
#define USB_CTL_HOSTMODEEN_MASK 0x8u
#define USB_CTL_HOSTMODEEN_SHIFT 3
#define USB_CTL_HOSTMODEEN_WIDTH 1

```

```

#define USB_CTL_HOSTMODEEN(x)
(((uint8_t) (((uint8_t)(x))<<USB_CTL_HOSTMODEEN_SHIFT)) &USB_CTL_HOSTMODEEN
_MASK)
#define USB_CTL_RESET_MASK 0x10u
#define USB_CTL_RESET_SHIFT 4
#define USB_CTL_RESET_WIDTH 1
#define USB_CTL_RESET(x)
(((uint8_t) (((uint8_t)(x))<<USB_CTL_RESET_SHIFT)) &USB_CTL_RESET_MASK)
#define USB_CTL_TXSUSPENDTOKENBUSY_MASK 0x20u
#define USB_CTL_TXSUSPENDTOKENBUSY_SHIFT 5
#define USB_CTL_TXSUSPENDTOKENBUSY_WIDTH 1
#define USB_CTL_TXSUSPENDTOKENBUSY(x)
(((uint8_t) (((uint8_t)(x))<<USB_CTL_TXSUSPENDTOKENBUSY_SHIFT)) &USB_CTL_TX
SUSPENDTOKENBUSY_MASK)
#define USB_CTL_SE0_MASK 0x40u
#define USB_CTL_SE0_SHIFT 6
#define USB_CTL_SE0_WIDTH 1
#define USB_CTL_SE0(x)
(((uint8_t) (((uint8_t)(x))<<USB_CTL_SE0_SHIFT)) &USB_CTL_SE0_MASK)
#define USB_CTL_JSTATE_MASK 0x80u
#define USB_CTL_JSTATE_SHIFT 7
#define USB_CTL_JSTATE_WIDTH 1
#define USB_CTL_JSTATE(x)
(((uint8_t) (((uint8_t)(x))<<USB_CTL_JSTATE_SHIFT)) &USB_CTL_JSTATE_MASK)
/* ADDR Bit Fields */
#define USB_ADDR_ADDR_MASK 0x7Fu
#define USB_ADDR_ADDR_SHIFT 0
#define USB_ADDR_ADDR_WIDTH 7
#define USB_ADDR_ADDR(x)
(((uint8_t) (((uint8_t)(x))<<USB_ADDR_ADDR_SHIFT)) &USB_ADDR_ADDR_MASK)
#define USB_ADDR_LSEN_MASK 0x80u
#define USB_ADDR_LSEN_SHIFT 7
#define USB_ADDR_LSEN_WIDTH 1
#define USB_ADDR_LSEN(x)
(((uint8_t) (((uint8_t)(x))<<USB_ADDR_LSEN_SHIFT)) &USB_ADDR_LSEN_MASK)
/* BDTPAGE1 Bit Fields */
#define USB_BDTPAGE1_BDTBA_MASK 0xFEu
#define USB_BDTPAGE1_BDTBA_SHIFT 1
#define USB_BDTPAGE1_BDTBA_WIDTH 7
#define USB_BDTPAGE1_BDTBA(x)
(((uint8_t) (((uint8_t)(x))<<USB_BDTPAGE1_BDTBA_SHIFT)) &USB_BDTPAGE1_BDTBA
_MASK)
/* FRMNUML Bit Fields */
#define USB_FRMNUML_FRM_MASK 0xFFu
#define USB_FRMNUML_FRM_SHIFT 0
#define USB_FRMNUML_FRM_WIDTH 8
#define USB_FRMNUML_FRM(x)
(((uint8_t) (((uint8_t)(x))<<USB_FRMNUML_FRM_SHIFT)) &USB_FRMNUML_FRM_MASK)
/* FRMNUMH Bit Fields */
#define USB_FRMNUMH_FRM_MASK 0x7u
#define USB_FRMNUMH_FRM_SHIFT 0
#define USB_FRMNUMH_FRM_WIDTH 3
#define USB_FRMNUMH_FRM(x)
(((uint8_t) (((uint8_t)(x))<<USB_FRMNUMH_FRM_SHIFT)) &USB_FRMNUMH_FRM_MASK)

```

```

/* TOKEN Bit Fields */
#define USB_TOKEN_TOKENENDPT_MASK 0xFFu
#define USB_TOKEN_TOKENENDPT_SHIFT 0
#define USB_TOKEN_TOKENENDPT_WIDTH 4
#define USB_TOKEN_TOKENENDPT(x) (((uint8_t)((uint8_t)(x)) << USB_TOKEN_TOKENENDPT_SHIFT) & USB_TOKEN_TOKENENDPT_MASK)
#define USB_TOKEN_TOKENPID_MASK 0xF0u
#define USB_TOKEN_TOKENPID_SHIFT 4
#define USB_TOKEN_TOKENPID_WIDTH 4
#define USB_TOKEN_TOKENPID(x) (((uint8_t)((uint8_t)(x)) << USB_TOKEN_TOKENPID_SHIFT) & USB_TOKEN_TOKENPID_MASK)
/* SOFTHLD Bit Fields */
#define USB_SOFTHLD_CNT_MASK 0xFFFFu
#define USB_SOFTHLD_CNT_SHIFT 0
#define USB_SOFTHLD_CNT_WIDTH 8
#define USB_SOFTHLD_CNT(x) (((uint8_t)((uint8_t)(x)) << USB_SOFTHLD_CNT_SHIFT) & USB_SOFTHLD_CNT_MASK)
/* BDTPAGE2 Bit Fields */
#define USB_BDTPAGE2_BDTBA_MASK 0xFFFFu
#define USB_BDTPAGE2_BDTBA_SHIFT 0
#define USB_BDTPAGE2_BDTBA_WIDTH 8
#define USB_BDTPAGE2_BDTBA(x) (((uint8_t)((uint8_t)(x)) << USB_BDTPAGE2_BDTBA_SHIFT) & USB_BDTPAGE2_BDTBA_MASK)
/* BDTPAGE3 Bit Fields */
#define USB_BDTPAGE3_BDTBA_MASK 0xFFFFu
#define USB_BDTPAGE3_BDTBA_SHIFT 0
#define USB_BDTPAGE3_BDTBA_WIDTH 8
#define USB_BDTPAGE3_BDTBA(x) (((uint8_t)((uint8_t)(x)) << USB_BDTPAGE3_BDTBA_SHIFT) & USB_BDTPAGE3_BDTBA_MASK)
/* ENDPT Bit Fields */
#define USB_ENDPT_EPHSHK_MASK 0x1u
#define USB_ENDPT_EPHSHK_SHIFT 0
#define USB_ENDPT_EPHSHK_WIDTH 1
#define USB_ENDPT_EPHSHK(x) (((uint8_t)((uint8_t)(x)) << USB_ENDPT_EPHSHK_SHIFT) & USB_ENDPT_EPHSHK_MASK)
#define USB_ENDPT_EPSTALL_MASK 0x2u
#define USB_ENDPT_EPSTALL_SHIFT 1
#define USB_ENDPT_EPSTALL_WIDTH 1
#define USB_ENDPT_EPSTALL(x) (((uint8_t)((uint8_t)(x)) << USB_ENDPT_EPSTALL_SHIFT) & USB_ENDPT_EPSTALL_MASK)
#define USB_ENDPT_EPTXEN_MASK 0x4u
#define USB_ENDPT_EPTXEN_SHIFT 2
#define USB_ENDPT_EPTXEN_WIDTH 1
#define USB_ENDPT_EPTXEN(x) (((uint8_t)((uint8_t)(x)) << USB_ENDPT_EPTXEN_SHIFT) & USB_ENDPT_EPTXEN_MASK)
#define USB_ENDPT_EPRXEN_MASK 0x8u
#define USB_ENDPT_EPRXEN_SHIFT 3

```

```

#define USB_ENDPT_EPRXEN_WIDTH 1
#define USB_ENDPT_EPRXEN(x) (((uint8_t)((uint8_t)(x))<<USB_ENDPT_EPRXEN_SHIFT))&USB_ENDPT_EPRXEN_MASK
#define USB_ENDPT_EPCTLDIS_MASK 0x10u
#define USB_ENDPT_EPCTLDIS_SHIFT 4
#define USB_ENDPT_EPCTLDIS_WIDTH 1
#define USB_ENDPT_EPCTLDIS(x) (((uint8_t)((uint8_t)(x))<<USB_ENDPT_EPCTLDIS_SHIFT))&USB_ENDPT_EPCTLDIS_MASK
#define USB_ENDPT_RETRYDIS_MASK 0x40u
#define USB_ENDPT_RETRYDIS_SHIFT 6
#define USB_ENDPT_RETRYDIS_WIDTH 1
#define USB_ENDPT_RETRYDIS(x) (((uint8_t)((uint8_t)(x))<<USB_ENDPT_RETRYDIS_SHIFT))&USB_ENDPT_RETRYDIS_MASK
#define USB_ENDPT_HOSTWOHUB_MASK 0x80u
#define USB_ENDPT_HOSTWOHUB_SHIFT 7
#define USB_ENDPT_HOSTWOHUB_WIDTH 1
#define USB_ENDPT_HOSTWOHUB(x) (((uint8_t)((uint8_t)(x))<<USB_ENDPT_HOSTWOHUB_SHIFT))&USB_ENDPT_HOSTWOHUB_MASK
/* USBCTRL Bit Fields */
#define USB_USBCTRL_PDE_MASK 0x40u
#define USB_USBCTRL_PDE_SHIFT 6
#define USB_USBCTRL_PDE_WIDTH 1
#define USB_USBCTRL_PDE(x) (((uint8_t)((uint8_t)(x))<<USB_USBCTRL_PDE_SHIFT))&USB_USBCTRL_PDE_MASK
#define USB_USBCTRL_SUSP_MASK 0x80u
#define USB_USBCTRL_SUSP_SHIFT 7
#define USB_USBCTRL_SUSP_WIDTH 1
#define USB_USBCTRL_SUSP(x) (((uint8_t)((uint8_t)(x))<<USB_USBCTRL_SUSP_SHIFT))&USB_USBCTRL_SUSP_MASK
/* OBSERVE Bit Fields */
#define USB_OBSERVE_DMPD_MASK 0x10u
#define USB_OBSERVE_DMPD_SHIFT 4
#define USB_OBSERVE_DMPD_WIDTH 1
#define USB_OBSERVE_DMPD(x) (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DMPD_SHIFT))&USB_OBSERVE_DMPD_MASK
#define USB_OBSERVE_DPPD_MASK 0x40u
#define USB_OBSERVE_DPPD_SHIFT 6
#define USB_OBSERVE_DPPD_WIDTH 1
#define USB_OBSERVE_DPPD(x) (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPD_SHIFT))&USB_OBSERVE_DPPD_MASK
#define USB_OBSERVE_DPPU_MASK 0x80u
#define USB_OBSERVE_DPPU_SHIFT 7
#define USB_OBSERVE_DPPU_WIDTH 1
#define USB_OBSERVE_DPPU(x) (((uint8_t)((uint8_t)(x))<<USB_OBSERVE_DPPU_SHIFT))&USB_OBSERVE_DPPU_MASK
/* CONTROL Bit Fields */

```

```

#define USB_CONTROL_DPPULLUPNONOTG_MASK          0x10u
#define USB_CONTROL_DPPULLUPNONOTG_SHIFT         4
#define USB_CONTROL_DPPULLUPNONOTG_WIDTH          1
#define USB_CONTROL_DPPULLUPNONOTG(x)             (((uint8_t)((uint8_t)(x))<<USB_CONTROL_DPPULLUPNONOTG_SHIFT))&USB_CONTROL_DPPULLUPNONOTG_MASK
/* USBTRC0 Bit Fields */
#define USB_USBTRC0_USB_RESUME_INT_MASK          0x1u
#define USB_USBTRC0_USB_RESUME_INT_SHIFT         0
#define USB_USBTRC0_USB_RESUME_INT_WIDTH          1
#define USB_USBTRC0_USB_RESUME_INT(x)             (((uint8_t)((uint8_t)(x))<<USB_USBTRC0_USB_RESUME_INT_SHIFT))&USB_USBTRC0_USB_RESUME_INT_MASK
#define USB_USBTRC0_SYNC_DET_MASK                 0x2u
#define USB_USBTRC0_SYNC_DET_SHIFT                1
#define USB_USBTRC0_SYNC_DET_WIDTH               1
#define USB_USBTRC0_SYNC_DET(x)                  (((uint8_t)((uint8_t)(x))<<USB_USBTRC0_SYNC_DET_SHIFT))&USB_USBTRC0_SYNC_DET_MASK
#define USB_USBTRC0_USBRESMEN_MASK                0x20u
#define USB_USBTRC0_USBRESMEN_SHIFT               5
#define USB_USBTRC0_USBRESMEN_WIDTH              1
#define USB_USBTRC0_USBRESMEN(x)                 (((uint8_t)((uint8_t)(x))<<USB_USBTRC0_USBRESMEN_SHIFT))&USB_USBTRC0_USBRESMEN_MASK
#define USB_USBTRC0_USBRESET_MASK                 0x80u
#define USB_USBTRC0_USBRESET_SHIFT                7
#define USB_USBTRC0_USBRESET_WIDTH               1
#define USB_USBTRC0_USBRESET(x)                  (((uint8_t)((uint8_t)(x))<<USB_USBTRC0_USBRESET_SHIFT))&USB_USBTRC0_USBRESET_MASK
/* USBFRMADJUST Bit Fields */
#define USB_USBFRMADJUST_ADJ_MASK                0xFFu
#define USB_USBFRMADJUST_ADJ_SHIFT               0
#define USB_USBFRMADJUST_ADJ_WIDTH              8
#define USB_USBFRMADJUST_ADJ(x)                  (((uint8_t)((uint8_t)(x))<<USB_USBFRMADJUST_ADJ_SHIFT))&USB_USBFRMADJUST_ADJ_MASK
/* !
 * @}
 */
/* /* end of group USB_Register_Masks */


```

```

/* USB - Peripheral instance base addresses */
/** Peripheral USB0 base address */
#define USB0_BASE                                (0x40072000u)
/** Peripheral USB0 base pointer */
#define USB0                                     ((USB_Type *)USB0_BASE)
#define USB0_BASE_PTR                           (USB0)
/** Array initializer of USB peripheral base addresses */
#define USB_BASE_ADDRS                          { USB0_BASE }
/** Array initializer of USB peripheral base pointers */
#define USB_BASE_PTRS                           { USB0 }


```

```

/** Interrupt vectors for the USB peripheral type */
#define USB_IRQS { USB0_IRQn }

/*
-----  

-- USB - Register accessor macros  

-----  

----- */

/*!  

 * @addtogroup USB_Register_Accessor_Macros USB - Register accessor  

macros  

 * @{
 * /  

  

/* USB - Register instance definitions */  

/* USB0 */  

#define USB0_PERID USB_PERID_REG(USB0)  

#define USB0_IDCOMP USB_IDCOMP_REG(USB0)  

#define USB0_REV USB_REV_REG(USB0)  

#define USB0_ADDINFO USB_ADDINFO_REG(USB0)  

#define USB0_OTGISTAT USB_OTGISTAT_REG(USB0)  

#define USB0_OTGICR USB_OTGICR_REG(USB0)  

#define USB0_OTGSTAT USB_OTGSTAT_REG(USB0)  

#define USB0_OTGCTL USB_OTGCTL_REG(USB0)  

#define USB0_ISTAT USB_ISTAT_REG(USB0)  

#define USB0_INTEN USB_INTEN_REG(USB0)  

#define USB0_ERRSTAT USB_ERRSTAT_REG(USB0)  

#define USB0_ERREN USB_ERREN_REG(USB0)  

#define USB0_STAT USB_STAT_REG(USB0)  

#define USB0_CTL USB_CTL_REG(USB0)  

#define USB0_ADDR USB_ADDR_REG(USB0)  

#define USB0_BDTPAGE1 USB_BDTPAGE1_REG(USB0)  

#define USB0_FRMNUML USB_FRMNUML_REG(USB0)  

#define USB0_FRMNUMH USB_FRMNUMH_REG(USB0)  

#define USB0_TOKEN USB_TOKEN_REG(USB0)  

#define USB0_SOFTHLD USB_SOFTHLD_REG(USB0)  

#define USB0_BDTPAGE2 USB_BDTPAGE2_REG(USB0)  

#define USB0_BDTPAGE3 USB_BDTPAGE3_REG(USB0)  

#define USB0_ENDPT0 USB_ENDPT_REG(USB0, 0)  

#define USB0_ENDPT1 USB_ENDPT_REG(USB0, 1)  

#define USB0_ENDPT2 USB_ENDPT_REG(USB0, 2)  

#define USB0_ENDPT3 USB_ENDPT_REG(USB0, 3)  

#define USB0_ENDPT4 USB_ENDPT_REG(USB0, 4)  

#define USB0_ENDPT5 USB_ENDPT_REG(USB0, 5)  

#define USB0_ENDPT6 USB_ENDPT_REG(USB0, 6)  

#define USB0_ENDPT7 USB_ENDPT_REG(USB0, 7)  

#define USB0_ENDPT8 USB_ENDPT_REG(USB0, 8)  

#define USB0_ENDPT9 USB_ENDPT_REG(USB0, 9)  

#define USB0_ENDPT10 USB_ENDPT_REG(USB0, 10)  

#define USB0_ENDPT11 USB_ENDPT_REG(USB0, 11)  

#define USB0_ENDPT12 USB_ENDPT_REG(USB0, 12)  

#define USB0_ENDPT13 USB_ENDPT_REG(USB0, 13)

```

```

#define USB0_ENDPT14          USB_ENDPT_REG(USB0,14)
#define USB0_ENDPT15          USB_ENDPT_REG(USB0,15)
#define USB0_USBCTRL           USB_USBCTRL_REG(USB0)
#define USB0_OBSERVE           USB_OBSERVE_REG(USB0)
#define USB0_CONTROL            USB_CONTROL_REG(USB0)
#define USB0_USBTRC0           USB_USBTRC0_REG(USB0)
#define USB0_USBFRMADJUST      USB_USBFRMADJUST_REG(USB0)

/* USB - Register array accessors */
#define USB0_ENDPT(index)      USB_ENDPT_REG(USB0,index)

/*!
 * @}
 */ /* end of group USB_Register_Accessor_Macros */

/*!
 * @}
 */ /* end of group USB_Peripheral_Access_Layer */

/*
** End of section using anonymous unions
*/
#if defined(__ARMCC_VERSION)
    #pragma pop
#elif defined(__CWCC__)
    #pragma pop
#elif defined(__GNUC__)
    /* leave anonymous unions enabled */
#elif defined(__IAR_SYSTEMS_ICC__)
    #pragma language=default
#else
    #error Not supported compiler type
#endif

/*!
 * @}
 */ /* end of group Peripheral_access_layer */

/*
-----  

-- Backward Compatibility  

----- */
/*!  

 * @addtogroup Backward_Compatibility_Symbols Backward Compatibility  

 * @{
 */

```

```

#define DMA_REQC_ARR_DMAC_MASK
This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC_SHIFT
This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMAC(x)
This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_MASK
This_symbol_has_been_deprecated
#define DMA_REQC_ARR_CFSM_SHIFT
This_symbol_has_been_deprecated
#define DMA_REQC0
This_symbol_has_been_deprecated
#define DMA_REQC1
This_symbol_has_been_deprecated
#define DMA_REQC2
This_symbol_has_been_deprecated
#define DMA_REQC3
This_symbol_has_been_deprecated
#define MCG_S_LOLS_MASK
#define MCG_S_LOLS_SHIFT
#define SIM_FCFG2_MAXADDR_MASK
#define SIM_FCFG2_MAXADDR_SHIFT
#define SIM_FCFG2_MAXADDR
#define SPI_C2_SPLPIE_MASK
This_symbol_has_been_deprecated
#define SPI_C2_SPLPIE_SHIFT
This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_MASK
This_symbol_has_been_deprecated
#define UART_C4_LBKDDMAS_SHIFT
This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_MASK
This_symbol_has_been_deprecated
#define UART_C4_ILDMAS_SHIFT
This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_MASK
This_symbol_has_been_deprecated
#define UART_C4_TCDMAS_SHIFT
This_symbol_has_been_deprecated
#define UARTLP_Type
#define UARTLP_BDH_REG
#define UARTLP_BDL_REG
#define UARTLP_C1_REG
#define UARTLP_C2_REG
#define UARTLP_S1_REG
#define UARTLP_S2_REG
#define UARTLP_C3_REG
#define UARTLP_D_REG
#define UARTLP_MA1_REG
#define UARTLP_MA2_REG
#define UARTLP_C4_REG
#define UARTLP_C5_REG
#define UARTLP_BDH_SBR_MASK
MCG_S_LOLS0_MASK
MCG_S_LOLS0_SHIFT
SIM_FCFG2_MAXADDR0_MASK
SIM_FCFG2_MAXADDR0_SHIFT
SIM_FCFG2_MAXADDR0
UART0_Type
UART0_BDH_REG
UART0_BDL_REG
UART0_C1_REG
UART0_C2_REG
UART0_S1_REG
UART0_S2_REG
UART0_C3_REG
UART0_D_REG
UART0_MA1_REG
UART0_MA2_REG
UART0_C4_REG
UART0_C5_REG
UART0_BDH_SBR_MASK

```

```

#define UARTLP_BDH_SBR_SHIFT
#define UARTLP_BDH_SBR(x)
#define UARTLP_BDH_SBNS_MASK
#define UARTLP_BDH_SBNS_SHIFT
#define UARTLP_BDH_RXEDGIE_MASK
#define UARTLP_BDH_RXEDGIE_SHIFT
#define UARTLP_BDH_LBKDIE_MASK
#define UARTLP_BDH_LBKDIE_SHIFT
#define UARTLP_BDL_SBR_MASK
#define UARTLP_BDL_SBR_SHIFT
#define UARTLP_BDL_SBR(x)
#define UARTLP_C1_PT_MASK
#define UARTLP_C1_PT_SHIFT
#define UARTLP_C1_PE_MASK
#define UARTLP_C1_PE_SHIFT
#define UARTLP_C1_ILT_MASK
#define UARTLP_C1_ILT_SHIFT
#define UARTLP_C1_WAKE_MASK
#define UARTLP_C1_WAKE_SHIFT
#define UARTLP_C1_M_MASK
#define UARTLP_C1_M_SHIFT
#define UARTLP_C1_RSRC_MASK
#define UARTLP_C1_RSRC_SHIFT
#define UARTLP_C1_DOZEEN_MASK
#define UARTLP_C1_DOZEEN_SHIFT
#define UARTLP_C1_LOOPS_MASK
#define UARTLP_C1_LOOPS_SHIFT
#define UARTLP_C2_SBK_MASK
#define UARTLP_C2_SBK_SHIFT
#define UARTLP_C2_RWU_MASK
#define UARTLP_C2_RWU_SHIFT
#define UARTLP_C2_RE_MASK
#define UARTLP_C2_RE_SHIFT
#define UARTLP_C2_TE_MASK
#define UARTLP_C2_TE_SHIFT
#define UARTLP_C2_ILIE_MASK
#define UARTLP_C2_ILIE_SHIFT
#define UARTLP_C2_RIE_MASK
#define UARTLP_C2_RIE_SHIFT
#define UARTLP_C2_TCIE_MASK
#define UARTLP_C2_TCIE_SHIFT
#define UARTLP_C2_TIE_MASK
#define UARTLP_C2_TIE_SHIFT
#define UARTLP_S1_PF_MASK
#define UARTLP_S1_PF_SHIFT
#define UARTLP_S1_FE_MASK
#define UARTLP_S1_FE_SHIFT
#define UARTLP_S1_NF_MASK
#define UARTLP_S1_NF_SHIFT
#define UARTLP_S1_OR_MASK
#define UARTLP_S1_OR_SHIFT
#define UARTLP_S1_IDLE_MASK
#define UARTLP_S1_IDLE_SHIFT
#define UARTLP_S1_RDRF_MASK
UART0_BDH_SBR_SHIFT
UART0_BDH_SBR(x)
UART0_BDH_SBNS_MASK
UART0_BDH_SBNS_SHIFT
UART0_BDH_RXEDGIE_MASK
UART0_BDH_RXEDGIE_SHIFT
UART0_BDH_LBKDIE_MASK
UART0_BDH_LBKDIE_SHIFT
UART0_BDL_SBR_MASK
UART0_BDL_SBR_SHIFT
UART0_BDL_SBR(x)
UART0_C1_PT_MASK
UART0_C1_PT_SHIFT
UART0_C1_PE_MASK
UART0_C1_PE_SHIFT
UART0_C1_ILT_MASK
UART0_C1_ILT_SHIFT
UART0_C1_WAKE_MASK
UART0_C1_WAKE_SHIFT
UART0_C1_M_MASK
UART0_C1_M_SHIFT
UART0_C1_RSRC_MASK
UART0_C1_RSRC_SHIFT
UART0_C1_DOZEEN_MASK
UART0_C1_DOZEEN_SHIFT
UART0_C1_LOOPS_MASK
UART0_C1_LOOPS_SHIFT
UART0_C2_SBK_MASK
UART0_C2_SBK_SHIFT
UART0_C2_RWU_MASK
UART0_C2_RWU_SHIFT
UART0_C2_RE_MASK
UART0_C2_RE_SHIFT
UART0_C2_TE_MASK
UART0_C2_TE_SHIFT
UART0_C2_ILIE_MASK
UART0_C2_ILIE_SHIFT
UART0_C2_RIE_MASK
UART0_C2_RIE_SHIFT
UART0_C2_TCIE_MASK
UART0_C2_TCIE_SHIFT
UART0_C2_TIE_MASK
UART0_C2_TIE_SHIFT
UART0_S1_PF_MASK
UART0_S1_PF_SHIFT
UART0_S1_FE_MASK
UART0_S1_FE_SHIFT
UART0_S1_NF_MASK
UART0_S1_NF_SHIFT
UART0_S1_OR_MASK
UART0_S1_OR_SHIFT
UART0_S1_IDLE_MASK
UART0_S1_IDLE_SHIFT
UART0_S1_RDRF_MASK

```

```

#define UARTLP_S1_RDRF_SHIFT          UART0_S1_RDRF_SHIFT
#define UARTLP_S1_TC_MASK             UART0_S1_TC_MASK
#define UARTLP_S1_TC_SHIFT            UART0_S1_TC_SHIFT
#define UARTLP_S1_TDRE_MASK           UART0_S1_TDRE_MASK
#define UARTLP_S1_TDRE_SHIFT          UART0_S1_TDRE_SHIFT
#define UARTLP_S2_RAF_MASK            UART0_S2_RAF_MASK
#define UARTLP_S2_RAF_SHIFT           UART0_S2_RAF_SHIFT
#define UARTLP_S2_LBKDE_MASK          UART0_S2_LBKDE_MASK
#define UARTLP_S2_LBKDE_SHIFT         UART0_S2_LBKDE_SHIFT
#define UARTLP_S2_BRK13_MASK          UART0_S2_BRK13_MASK
#define UARTLP_S2_BRK13_SHIFT         UART0_S2_BRK13_SHIFT
#define UARTLP_S2_RWUID_MASK          UART0_S2_RWUID_MASK
#define UARTLP_S2_RWUID_SHIFT         UART0_S2_RWUID_SHIFT
#define UARTLP_S2_RXINV_MASK          UART0_S2_RXINV_MASK
#define UARTLP_S2_RXINV_SHIFT         UART0_S2_RXINV_SHIFT
#define UARTLP_S2_MSBF_MASK           UART0_S2_MSBF_MASK
#define UARTLP_S2_MSBF_SHIFT          UART0_S2_MSBF_SHIFT
#define UARTLP_S2_RXEDGIF_MASK        UART0_S2_RXEDGIF_MASK
#define UARTLP_S2_RXEDGIF_SHIFT       UART0_S2_RXEDGIF_SHIFT
#define UARTLP_S2_LBKDIF_MASK         UART0_S2_LBKDIF_MASK
#define UARTLP_S2_LBKDIF_SHIFT        UART0_S2_LBKDIF_SHIFT
#define UARTLP_C3_PEIE_MASK           UART0_C3_PEIE_MASK
#define UARTLP_C3_PEIE_SHIFT          UART0_C3_PEIE_SHIFT
#define UARTLP_C3_FEIE_MASK           UART0_C3_FEIE_MASK
#define UARTLP_C3_FEIE_SHIFT          UART0_C3_FEIE_SHIFT
#define UARTLP_C3_NEIE_MASK           UART0_C3_NEIE_MASK
#define UARTLP_C3_NEIE_SHIFT          UART0_C3_NEIE_SHIFT
#define UARTLP_C3_ORIE_MASK           UART0_C3_ORIE_MASK
#define UARTLP_C3_ORIE_SHIFT          UART0_C3_ORIE_SHIFT
#define UARTLP_C3_TXINV_MASK          UART0_C3_TXINV_MASK
#define UARTLP_C3_TXINV_SHIFT         UART0_C3_TXINV_SHIFT
#define UARTLP_C3_TXDIR_MASK          UART0_C3_TXDIR_MASK
#define UARTLP_C3_TXDIR_SHIFT         UART0_C3_TXDIR_SHIFT
#define UARTLP_C3_R9T8_MASK           UART0_C3_R9T8_MASK
#define UARTLP_C3_R9T8_SHIFT          UART0_C3_R9T8_SHIFT
#define UARTLP_C3_R8T9_MASK           UART0_C3_R8T9_MASK
#define UARTLP_C3_R8T9_SHIFT          UART0_C3_R8T9_SHIFT
#define UARTLP_D_R0T0_MASK            UART0_D_R0T0_MASK
#define UARTLP_D_R0T0_SHIFT           UART0_D_R0T0_SHIFT
#define UARTLP_D_R1T1_MASK            UART0_D_R1T1_MASK
#define UARTLP_D_R1T1_SHIFT           UART0_D_R1T1_SHIFT
#define UARTLP_D_R2T2_MASK            UART0_D_R2T2_MASK
#define UARTLP_D_R2T2_SHIFT           UART0_D_R2T2_SHIFT
#define UARTLP_D_R3T3_MASK            UART0_D_R3T3_MASK
#define UARTLP_D_R3T3_SHIFT           UART0_D_R3T3_SHIFT
#define UARTLP_D_R4T4_MASK            UART0_D_R4T4_MASK
#define UARTLP_D_R4T4_SHIFT           UART0_D_R4T4_SHIFT
#define UARTLP_D_R5T5_MASK            UART0_D_R5T5_MASK
#define UARTLP_D_R5T5_SHIFT           UART0_D_R5T5_SHIFT
#define UARTLP_D_R6T6_MASK            UART0_D_R6T6_MASK
#define UARTLP_D_R6T6_SHIFT           UART0_D_R6T6_SHIFT
#define UARTLP_D_R7T7_MASK            UART0_D_R7T7_MASK
#define UARTLP_D_R7T7_SHIFT           UART0_D_R7T7_SHIFT
#define UARTLP_MA1_MA_MASK            UART0_MA1_MA_MASK

```

```

#define UARTLP_MA1_MA_SHIFT          UART0_MA1_MA_SHIFT
#define UARTLP_MA1_MA(x)             UART0_MA1_MA(x)
#define UARTLP_MA2_MA_MASK           UART0_MA2_MA_MASK
#define UARTLP_MA2_MA_SHIFT          UART0_MA2_MA_SHIFT
#define UARTLP_MA2_MA(x)             UART0_MA2_MA(x)
#define UARTLP_C4_OSR_MASK           UART0_C4_OSR_MASK
#define UARTLP_C4_OSR_SHIFT          UART0_C4_OSR_SHIFT
#define UARTLP_C4_OSR(x)             UART0_C4_OSR(x)
#define UARTLP_C4_M10_MASK           UART0_C4_M10_MASK
#define UARTLP_C4_M10_SHIFT          UART0_C4_M10_SHIFT
#define UARTLP_C4_MAEN2_MASK          UART0_C4_MAEN2_MASK
#define UARTLP_C4_MAEN2_SHIFT         UART0_C4_MAEN2_SHIFT
#define UARTLP_C4_MAEN1_MASK          UART0_C4_MAEN1_MASK
#define UARTLP_C4_MAEN1_SHIFT         UART0_C4_MAEN1_SHIFT
#define UARTLP_C5_RESYNCDIS_MASK     UART0_C5_RESYNCDIS_MASK
#define UARTLP_C5_RESYNCDIS_SHIFT    UART0_C5_RESYNCDIS_SHIFT
#define UARTLP_C5_BOTHEDGE_MASK       UART0_C5_BOTHEDGE_MASK
#define UARTLP_C5_BOTHEDGE_SHIFT      UART0_C5_BOTHEDGE_SHIFT
#define UARTLP_C5_RDMAE_MASK          UART0_C5_RDMAE_MASK
#define UARTLP_C5_RDMAE_SHIFT         UART0_C5_RDMAE_SHIFT
#define UARTLP_C5_TDMAE_MASK          UART0_C5_TDMAE_MASK
#define UARTLP_C5_TDMAE_SHIFT         UART0_C5_TDMAE_SHIFT
#define UARTLP_BASES                  UARTLP_BASES

#define NV_FOPT_EZPORT_DIS_MASK      ADC_BASE_PTRS
This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT     CMP_BASE_PTRS
This_symbol_has_been_deprecated
#define ADC_BASES                    DAC_BASE_PTRS
#define CMP_BASES                   DMA_BASE_PTRS
#define DAC_BASES                   DMAMUX_BASE_PTRS
#define DMA_BASES                   FGPIOA_BASE_PTR
#define DMAMUX_BASES                FGPIOA_BASE
#define FGPTA_BASE_PTR               FGPIOA
#define FGPTA_BASE                   FGPIOB_BASE_PTR
#define FGPTA                         FGPIOB
#define FPTB                          FGPIOB
#define FPTB_BASE_PTR                FGPIOC_BASE_PTR
#define FPTB_BASE                     FGPIOC_BASE
#define FPTC                          FGPIOC
#define FPTC_BASE_PTR                FGPIOD_BASE_PTR
#define FPTC_BASE                     FGPIOD_BASE
#define FPTD                          FGPIOD
#define FPTD_BASE_PTR                FGPIOE_BASE_PTR
#define FPTD_BASE                     FGPIOE_BASE
#define FPTE                          FGPIOE
#define FPTE_BASE_PTR                GPIOA_BASE_PTRS
#define FPTE_BASE                     GPIOA_BASE
#define FGPIO                         GPIOA
#define FGPIO_BASE_PTR                GPIOB_BASE_PTR
#define FGPIO_BASE                     GPIOB_BASE

```

```

#define PTB                                GPIOB
#define PTC_BASE_PTR                         GPIOC_BASE_PTR
#define PTC_BASE                             GPIOC_BASE
#define PTC                                 GPIOC
#define PTD_BASE_PTR                         GPIOD_BASE_PTR
#define PTD_BASE                            GPIOD_BASE
#define PTD                                 GPIOD
#define PTE_BASE_PTR                         GPIOE_BASE_PTR
#define PTE_BASE                            GPIOE_BASE
#define PTE                                 GPIOE
#define GPIO_BASES                           GPIO_BASE_PTRS
#define I2C_BASES                            I2C_BASE_PTRS
#define LLWU_BASES                           LLWU_BASE_PTRS
#define LPTMR_BASES                          LPTMR_BASE_PTRS
#define MCG_BASES                            MCG_BASE_PTRS
#define MCM_BASES                            MCM_BASE_PTRS
#define MTB_BASES                            MTB_BASE_PTRS
#define MTBDWT_BASES                         MTBDWT_BASE_PTRS
#define NV_BASES                             NV_BASES
#define OSC_BASES                            OSC_BASE_PTRS
#define PIT_BASES                            PIT_BASE_PTRS
#define PMC_BASES                            PMC_BASE_PTRS
#define PORT_BASES                           PORT_BASE_PTRS
#define RCM_BASES                            RCM_BASE_PTRS
#define ROM_BASES                            ROM_BASE_PTRS
#define RTC_BASES                            RTC_BASE_PTRS
#define SIM_BASES                            SIM_BASE_PTRS
#define SMC_BASES                            SMC_BASE_PTRS
#define SPI_BASES                            SPI_BASE_PTRS
#define TPM_BASES                            TPM_BASE_PTRS
#define TSI_BASES                            TSI_BASE_PTRS
#define UART_BASES                           UART_BASE_PTRS
#define UART0_BASES                          UART0_BASE_PTRS
#define USB_BASES                            USB_BASE_PTRS
#define LPTimer_IRQn                         LPTMR0_IRQn
#define LPTimer_IRQHandler                   LPTMR0_IRQHandler
#define LLW_IRQn                            LLWU_IRQn
#define LLW_IRQHandler                      LLWU_IRQHandler

/*
 * @}
 */ /* end of group Backward_Compatibility_Symbols */

#else /* #if !defined(MKL25Z4_H_) */
/* There is already included the same memory map. Check if it is
compatible (has the same major version) */
#if (MCU_MEM_MAP_VERSION != 0x0200u)
#if (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING))
#warning There are included two not compatible versions of memory
maps. Please check possible differences.
#endif /* (!defined(MCU_MEM_MAP_SUPPRESS_VERSION_WARNING)) */
#endif /* (MCU_MEM_MAP_VERSION != 0x0200u) */
#endif /* #if !defined(MKL25Z4_H_) */

```

```

/* MKL25Z4.h, eof. */
/* dma.h
 * brief : DMA configuration function
 */

#ifndef __DMA_H__
#define __DMA_H__


/***
* @brief - Enables the clock for DMA
* @return void
***/
void dma_clockenable();

/***
* @brief - configuration for dma
* @return void
***/
void DMA_config();

void DMA2_IRQHandler();


#endif
/***
* @file port.h
* @brief Library for GPIO and LED management on KL25Z
* @author Shreya Chakraborty
* @author Miles Fain
* @version 1
* @date 2018-02-21
*/



#include <stdint.h>
#include <MKL25Z4.h>
#ifndef __PORT_H__
#define __PORT_H__


#define RGB_RED_PIN (18)
#define RGB_GREEN_PIN (19)
#define RGB_BLUE_PIN (1)

#define RGB_GREEN_ON() (PORTB_Clear( RGB_GREEN_PIN ))
#define RGB_GREEN_OFF() (PORTB_Set( RGB_GREEN_PIN ))
#define RGB_GREEN_TOGGLE() (PORTB_Toggle( RGB_GREEN_PIN ))


#define RGB_RED_ON() (PORTB_Clear( RGB_RED_PIN ))
#define RGB_RED_OFF() (PORTB_Set( RGB_RED_PIN ))
#define RGB_RED_TOGGLE() (PORTB_Toggle( RGB_RED_PIN ))


#define RGB_BLUE_ON() (PORTD_Clear( RGB_BLUE_PIN ))

```

```

#define RGB_BLUE_OFF() (PORTD_Set( RGB_BLUE_PIN ))
#define RGB_BLUE_TOGGLE() (PORTD_Toggle( RGB_BLUE_PIN ))

void GPIO_Configure();

static inline void Toggle_Red_LED()
{
    PTB->PTOR |= (1 << 18);
}

//These two functions should use the appropriate GPIO function to toggle
the output (PSOR)
static inline void PORTB_Set(uint8_t bit_num)
{
    PTB->PSOR |= (1 << bit_num); //port set output register for portB
the given pin number
}
static inline void PORTD_Set(uint8_t bit_num)
{
    PTD->PSOR |= (1 << bit_num); //port set output register for portD
for the given pin number
}
//These two functions should use the appropriate GPIO function to toggle
the output (PCOR)
static inline void PORTB_Clear(uint8_t bit_num)
{
    PTB->PCOR |= (1 << bit_num); //pin out clear for corresponding pin
in port b
}
static inline void PORTD_Clear(uint8_t bit_num)
{
    PTD->PCOR |= (1 << bit_num); //pin out clear for corresponding pin
in port d
}

//These two functions should use the appropriate GPIO function to toggle
the output (PTOR)
static inline void PORTB_Toggle(uint8_t bit_num)
{
    PTB->PTOR |= (1 << bit_num); //toggle port b pin bit_num
}
static inline void PORTD_Toggle(uint8_t bit_num)
{
    PTD->PTOR |= (1 << bit_num); //toggle port b pin bit_num
}

#endif // __PORT_H__
/*
** ##### Keil ARM C/C++ Compiler
** Compilers: Keil ARM C/C++ Compiler
**           Freescale C/C++ for Embedded ARM
**           GNU C Compiler
**           GNU C Compiler - CodeSourcery Sourcery G++
**           IAR ANSI C/C++ Compiler for ARM

```

```
**  
** Reference manual: KL25P80M48SF0RM, Rev.3, Sep 2012  
** Version: rev. 2.5, 2015-02-19  
** Build: b150612  
**  
** Abstract:  
** Extension to the CMSIS register access layer header.  
**  
** Copyright (c) 2015 Freescale Semiconductor, Inc.  
** All rights reserved.  
**  
** Redistribution and use in source and binary forms, with or without  
modification,  
** are permitted provided that the following conditions are met:  
**  
** o Redistributions of source code must retain the above copyright  
notice, this list  
** of conditions and the following disclaimer.  
**  
** o Redistributions in binary form must reproduce the above  
copyright notice, this  
** list of conditions and the following disclaimer in the  
documentation and/or  
** other materials provided with the distribution.  
**  
** o Neither the name of Freescale Semiconductor, Inc. nor the names  
of its  
** contributors may be used to endorse or promote products derived  
from this  
** software without specific prior written permission.  
**  
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND  
CONTRIBUTORS "AS IS" AND  
** ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
THE IMPLIED  
** WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
ARE  
** DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS  
BE LIABLE FOR  
** ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
CONSEQUENTIAL DAMAGES  
** (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
** LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON  
** ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR  
TORT  
** (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  
USE OF THIS  
** SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
**  
** http: www.freescale.com  
** mail: support@freescale.com  
**
```

```

**      Revisions:
**      - rev. 1.0 (2012-06-13)
**          Initial version.
**      - rev. 1.1 (2012-06-21)
**          Update according to reference manual rev. 1.
**      - rev. 1.2 (2012-08-01)
**          Device type UARTLP changed to UART0.
**      - rev. 1.3 (2012-10-04)
**          Update according to reference manual rev. 3.
**      - rev. 1.4 (2012-11-22)
**          MCG module - bit LOLS in MCG_S register renamed to LOLS0.
**          NV registers - bit EZPORT_DIS in NV_FOPT register removed.
**      - rev. 2.0 (2013-10-29)
**          Register accessor macros added to the memory map.
**          Symbols for Processor Expert memory map compatibility added to
the memory map.
**          Startup file for gcc has been updated according to CMSIS 3.2.
**          System initialization updated.
**      - rev. 2.1 (2014-07-16)
**          Module access macro module_BASES replaced by module_BASE_PTRS.
**          System initialization and startup updated.
**      - rev. 2.2 (2014-08-22)
**          System initialization updated - default clock config changed.
**      - rev. 2.3 (2014-08-28)
**          Update of startup files - possibility to override DefaultISR
added.
**      - rev. 2.4 (2014-10-14)
**          Interrupt INT_LPTimer renamed to INT_LPTMR0.
**      - rev. 2.5 (2015-02-19)
**          Renamed interrupt vector LLW to LLWU.
**
** ##########
*/
/*
 * WARNING! DO NOT EDIT THIS FILE DIRECTLY!
 *
 * This file was generated automatically and any changes may be lost.
 */
#ifndef __MKL25Z4_EXTENSION_H__
#define __MKL25Z4_EXTENSION_H__

#include "MKL25Z4.h"
#include "fsl_bitaccess.h"

#if defined(__IAR_SYSTEMS_ICC__)
/*
 * Suppress "Error[Pm008]: sections of code should not be 'commented
out' (MISRA C 2004 rule 2.4)"
 * as some register descriptions contain code examples
 */
#pragma diag_suppress=pm008
#endif

```

```

/*
 * MKL25Z4 ADC
 *
 * Analog-to-Digital Converter
 *
 * Registers defined in this header file:
 * - ADC_SC1 - ADC Status and Control Registers 1
 * - ADC_CFG1 - ADC Configuration Register 1
 * - ADC_CFG2 - ADC Configuration Register 2
 * - ADC_R - ADC Data Result Register
 * - ADC_CV1 - Compare Value Registers
 * - ADC_CV2 - Compare Value Registers
 * - ADC_SC2 - Status and Control Register 2
 * - ADC_SC3 - Status and Control Register 3
 * - ADC_OFS - ADC Offset Correction Register
 * - ADC_PG - ADC Plus-Side Gain Register
 * - ADC_MG - ADC Minus-Side Gain Register
 * - ADC_CLPD - ADC Plus-Side General Calibration Value Register
 * - ADC_CLPS - ADC Plus-Side General Calibration Value Register
 * - ADC_CLP4 - ADC Plus-Side General Calibration Value Register
 * - ADC_CLP3 - ADC Plus-Side General Calibration Value Register
 * - ADC_CLP2 - ADC Plus-Side General Calibration Value Register
 * - ADC_CLP1 - ADC Plus-Side General Calibration Value Register
 * - ADC_CLP0 - ADC Plus-Side General Calibration Value Register
 * - ADC_CLMD - ADC Minus-Side General Calibration Value Register
 * - ADC_CLMS - ADC Minus-Side General Calibration Value Register
 * - ADC_CLM4 - ADC Minus-Side General Calibration Value Register
 * - ADC_CLM3 - ADC Minus-Side General Calibration Value Register
 * - ADC_CLM2 - ADC Minus-Side General Calibration Value Register
 * - ADC_CLM1 - ADC Minus-Side General Calibration Value Register
 * - ADC_CLM0 - ADC Minus-Side General Calibration Value Register
*/
#define ADC_INSTANCE_COUNT (1U) /*!< Number of instances of the ADC module. */
#define ADC0_IDX (0U) /*!< Instance number for ADC0. */

/*****
 * ADC_SC1 - ADC Status and Control Registers 1
 ****/
/*!
 * @brief ADC_SC1 - ADC Status and Control Registers 1 (RW)
 *
 * Reset value: 0x0000001FU
 *
 * SC1A is used for both software and hardware trigger modes of
 * operation. To
 * allow sequential conversions of the ADC to be triggered by internal
 * peripherals,

```

```

 * the ADC can have more than one status and control register: one for
each
 * conversion. The SC1B-SC1n registers indicate potentially multiple SC1
registers
 * for use only in hardware trigger mode. See the chip configuration
information
 * about the number of SC1n registers specific to this device. The SC1n
registers
 * have identical fields, and are used in a "ping-pong" approach to
control ADC
 * operation. At any one point in time, only one of the SC1n registers is
actively
 * controlling ADC conversions. Updating SC1A while SC1n is actively
controlling
 * a conversion is allowed, and vice-versa for any of the SC1n registers
specific
 * to this MCU. Writing SC1A while SC1A is actively controlling a
conversion
 * aborts the current conversion. In Software Trigger mode, when
SC2[ADTRG]=0,
 * writes to SC1A subsequently initiate a new conversion, if SC1[ADCH]
contains a
 * value other than all 1s. Writing any of the SC1n registers while that
specific
 * SC1n register is actively controlling a conversion aborts the current
conversion.
 * None of the SC1B-SC1n registers are used for software trigger
operation and
 * therefore writes to the SC1B-SC1n registers do not initiate a new
conversion.
 */
/*!
 * @name Constants and macros for entire ADC_SC1 register
 */
/*@{*/
#define ADC_RD_SC1(base, index)  (ADC_SC1_REG(base, index))
#define ADC_WR_SC1(base, index, value) (ADC_SC1_REG(base, index) = (value))
#define ADC_RMW_SC1(base, index, mask, value) (ADC_WR_SC1(base, index,
(ADC_RD_SC1(base, index) & ~mask) | (value)))
#define ADC_SET_SC1(base, index, value) (BME_OR32(&ADC_SC1_REG(base,
index), (uint32_t)(value)))
#define ADC_CLR_SC1(base, index, value) (BME_AND32(&ADC_SC1_REG(base,
index), (uint32_t)(~(value))))
#define ADC_TOG_SC1(base, index, value) (BME_XOR32(&ADC_SC1_REG(base,
index), (uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual ADC_SC1 bitfields
*/
/*!
 * @name Register ADC_SC1, field ADCH[4:0] (RW)

```

```
  
*  
* Selects one of the input channels. The input channel decode depends on  
the  
* value of DIFF. DAD0-DAD3 are associated with the input pin pairs DADPx  
and  
* DADMx. Some of the input channel options in the bitfield-setting  
descriptions might  
* not be available for your device. For the actual ADC channel  
assignments for  
* your device, see the Chip Configuration details. The successive  
approximation  
* converter subsystem is turned off when the channel select bits are all  
set,  
* that is, ADCH = 11111. This feature allows explicit disabling of the  
ADC and  
* isolation of the input channel from all sources. Terminating  
continuous  
* conversions this way prevents an additional single conversion from  
being performed. It  
* is not necessary to set ADCH to all 1s to place the ADC in a low-power  
state  
* when continuous conversions are not enabled because the module  
automatically  
* enters a low-power state when a conversion completes.  
*  
* Values:  
* - 0b00000 - When DIFF=0, DADP0 is selected as input; when DIFF=1, DAD0  
is  
*     selected as input.  
* - 0b00001 - When DIFF=0, DADP1 is selected as input; when DIFF=1, DAD1  
is  
*     selected as input.  
* - 0b00010 - When DIFF=0, DADP2 is selected as input; when DIFF=1, DAD2  
is  
*     selected as input.  
* - 0b00011 - When DIFF=0, DADP3 is selected as input; when DIFF=1, DAD3  
is  
*     selected as input.  
* - 0b00100 - When DIFF=0, AD4 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b00101 - When DIFF=0, AD5 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b00110 - When DIFF=0, AD6 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b00111 - When DIFF=0, AD7 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b01000 - When DIFF=0, AD8 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b01001 - When DIFF=0, AD9 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b01010 - When DIFF=0, AD10 is selected as input; when DIFF=1, it is  
*     reserved.  
* - 0b01011 - When DIFF=0, AD11 is selected as input; when DIFF=1, it is  
*     reserved.
```

```

* - 0b01100 - When DIFF=0, AD12 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b01101 - When DIFF=0, AD13 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b01110 - When DIFF=0, AD14 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b01111 - When DIFF=0, AD15 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10000 - When DIFF=0, AD16 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10001 - When DIFF=0, AD17 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10010 - When DIFF=0, AD18 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10011 - When DIFF=0, AD19 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10100 - When DIFF=0, AD20 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10101 - When DIFF=0, AD21 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10110 - When DIFF=0, AD22 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b10111 - When DIFF=0, AD23 is selected as input; when DIFF=1, it is
*   reserved.
* - 0b11000 - Reserved.
* - 0b11001 - Reserved.
* - 0b11010 - When DIFF=0, Temp Sensor (single-ended) is selected as
input;
*   when DIFF=1, Temp Sensor (differential) is selected as input.
* - 0b11011 - When DIFF=0, Bandgap (single-ended) is selected as input;
when
*   DIFF=1, Bandgap (differential) is selected as input.
* - 0b11100 - Reserved.
* - 0b11101 - When DIFF=0, VREFSH is selected as input; when DIFF=1, -
VREFSH
*   (differential) is selected as input. Voltage reference selected is
determined
*   by SC2[REFSEL].
* - 0b11110 - When DIFF=0, VREFSL is selected as input; when DIFF=1, it
is
*   reserved. Voltage reference selected is determined by SC2[REFSEL].
* - 0b11111 - Module is disabled.
*/
/*@{ */
/*! @brief Read current value of the ADC_SC1_ADCH field. */
#define ADC_RD_SC1_ADCH(base, index) ((ADC_SC1_REG(base, index) &
ADC_SC1_ADCH_MASK) >> ADC_SC1_ADCH_SHIFT)
#define ADC_BRD_SC1_ADCH(base, index) (BME_UBFX32(&ADC_SC1_REG(base,
index), ADC_SC1_ADCH_SHIFT, ADC_SC1_ADCH_WIDTH))

/*! @brief Set the ADCH field to a new value. */
#define ADC_WR_SC1_ADCH(base, index, value) (ADC_RMW_SC1(base, index,
ADC_SC1_ADCH_MASK, ADC_SC1_ADCH(value)))

```

```

#define ADC_BWR_SC1_ADCH(base, index, value)
(BME_BFI32(&ADC_SC1_REG(base, index), ((uint32_t)(value) <<
ADC_SC1_ADCH_SHIFT), ADC_SC1_ADCH_SHIFT, ADC_SC1_ADCH_WIDTH))
/*@}*/

/*!
 * @name Register ADC_SC1, field DIFF[5] (RW)
 *
 * Configures the ADC to operate in differential mode. When enabled, this
mode
 * automatically selects from the differential channels, and changes the
 * conversion algorithm and the number of cycles to complete a
conversion.
 *
 * Values:
 * - 0b0 - Single-ended conversions and input channels are selected.
 * - 0b1 - Differential conversions and input channels are selected.
 */
/*@{*/
/*! @brief Read current value of the ADC_SC1_DIFF field. */
#define ADC_RD_SC1_DIFF(base, index) ((ADC_SC1_REG(base, index) &
ADC_SC1_DIFF_MASK) >> ADC_SC1_DIFF_SHIFT)
#define ADC_BRD_SC1_DIFF(base, index) (BME_UBFX32(&ADC_SC1_REG(base,
index), ADC_SC1_DIFF_SHIFT, ADC_SC1_DIFF_WIDTH))

/*! @brief Set the DIFF field to a new value. */
#define ADC_WR_SC1_DIFF(base, index, value) (ADC_RMW_SC1(base, index,
ADC_SC1_DIFF_MASK, ADC_SC1_DIFF(value)))
#define ADC_BWR_SC1_DIFF(base, index, value)
(BME_BFI32(&ADC_SC1_REG(base, index), ((uint32_t)(value) <<
ADC_SC1_DIFF_SHIFT), ADC_SC1_DIFF_SHIFT, ADC_SC1_DIFF_WIDTH))
/*@}*/

/*!
 * @name Register ADC_SC1, field AIEN[6] (RW)
 *
 * Enables conversion complete interrupts. When COCO becomes set while
the
 * respective AIEN is high, an interrupt is asserted.
 *
 * Values:
 * - 0b0 - Conversion complete interrupt is disabled.
 * - 0b1 - Conversion complete interrupt is enabled.
 */
/*@{*/
/*! @brief Read current value of the ADC_SC1_AIEN field. */
#define ADC_RD_SC1_AIEN(base, index) ((ADC_SC1_REG(base, index) &
ADC_SC1_AIEN_MASK) >> ADC_SC1_AIEN_SHIFT)
#define ADC_BRD_SC1_AIEN(base, index) (BME_UBFX32(&ADC_SC1_REG(base,
index), ADC_SC1_AIEN_SHIFT, ADC_SC1_AIEN_WIDTH))

/*! @brief Set the AIEN field to a new value. */
#define ADC_WR_SC1_AIEN(base, index, value) (ADC_RMW_SC1(base, index,
ADC_SC1_AIEN_MASK, ADC_SC1_AIEN(value)))

```

```

#define ADC_BWR_SC1_AIEN(base, index, value)
(BME_BFI32(&ADC_SC1_REG(base, index), ((uint32_t)(value) <<
ADC_SC1_AIEN_SHIFT), ADC_SC1_AIEN_SHIFT, ADC_SC1_AIEN_WIDTH))
/*@}*/

/*!!
 * @name Register ADC_SC1, field COCO[7] (RO)
 *
 * This is a read-only field that is set each time a conversion is
completed
 * when the compare function is disabled, or SC2[ACFE]=0 and the hardware
average
 * function is disabled, or SC3[AVGE]=0. When the compare function is
enabled, or
 * SC2[ACFE]=1, COCO is set upon completion of a conversion only if the
compare
 * result is true. When the hardware average function is enabled, or
SC3[AVGE]=1,
 * COCO is set upon completion of the selected number of conversions
(determined
 * by AVGS). COCO in SC1A is also set at the completion of a calibration
sequence.
 * COCO is cleared when the respective SC1n register is written or when
the
 * respective Rn register is read.
 *
 * Values:
 * - 0b0 - Conversion is not completed.
 * - 0b1 - Conversion is completed.
 */
/*@{*/
/*! @brief Read current value of the ADC_SC1_COCO field. */
#define ADC_RD_SC1_COCO(base, index) ((ADC_SC1_REG(base, index) &
ADC_SC1_COCO_MASK) >> ADC_SC1_COCO_SHIFT)
#define ADC_BRD_SC1_COCO(base, index) (BME_UBFX32(&ADC_SC1_REG(base,
index), ADC_SC1_COCO_SHIFT, ADC_SC1_COCO_WIDTH))
/*@}*/

*****  

* ADC_CFG1 - ADC Configuration Register 1  

*****  

* Reset value: 0x00000000U
*
* The configuration Register 1 (CFG1) selects the mode of operation,
clock
 * source, clock divide, and configuration for low power or long sample
time.

```

```

/*
/*!
 * @name Constants and macros for entire ADC_CFG1 register
 */
/*@{*/
#define ADC_RD_CFG1(base)          (ADC_CFG1_REG(base))
#define ADC_WR_CFG1(base, value)   (ADC_CFG1_REG(base) = (value))
#define ADC_RMW_CFG1(base, mask, value) (ADC_WR_CFG1(base,
(ADC_RD_CFG1(base) & ~mask) | (value)))
#define ADC_SET_CFG1(base, value)  (BME_OR32(&ADC_CFG1_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CFG1(base, value)  (BME_AND32(&ADC_CFG1_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CFG1(base, value)  (BME_XOR32(&ADC_CFG1_REG(base),
(uint32_t)(value)))
/*}@*/
/*
 * Constants & macros for individual ADC_CFG1 bitfields
 */
/*!
 * @name Register ADC_CFG1, field ADICLK[1:0] (RW)
 *
 * Selects the input clock source to generate the internal clock, ADCK.
Note
 * that when the ADACK clock source is selected, it is not required to be
active
 * prior to conversion start. When it is selected and it is not active
prior to a
 * conversion start, when CFG2[ADACKEN]=0, the asynchronous clock is
activated at
 * the start of a conversion and deactivated when conversions are
terminated. In
 * this case, there is an associated clock startup delay each time the
clock
 * source is re-activated.
*
 * Values:
 * - 0b00 - Bus clock
 * - 0b01 - (Bus clock)/2
 * - 0b10 - Alternate clock (ALTCLK)
 * - 0b11 - Asynchronous clock (ADACK)
*/
/*@*/
/*! @brief Read current value of the ADC_CFG1_ADICLK field. */
#define ADC_RD_CFG1_ADICLK(base) ((ADC_CFG1_REG(base) &
ADC_CFG1_ADICLK_MASK) >> ADC_CFG1_ADICLK_SHIFT)
#define ADC_BRD_CFG1_ADICLK(base) (BME_UBFX32(&ADC_CFG1_REG(base),
ADC_CFG1_ADICLK_SHIFT, ADC_CFG1_ADICLK_WIDTH))

/*! @brief Set the ADICLK field to a new value. */
#define ADC_WR_CFG1_ADICLK(base, value) (ADC_RMW_CFG1(base,
ADC_CFG1_ADICLK_MASK, ADC_CFG1_ADICLK(value)))

```

```

#define ADC_BWR_CFG1_ADICLK(base, value) (BME_BFI32(&ADC_CFG1_REG(base),  

((uint32_t)(value) << ADC_CFG1_ADICLK_SHIFT), ADC_CFG1_ADICLK_SHIFT,  

ADC_CFG1_ADICLK_WIDTH))  

/*@}*/

/*!  

 * @name Register ADC_CFG1, field MODE[3:2] (RW)  

 *  

 * Selects the ADC resolution mode.  

 *  

 * Values:  

 * - 0b00 - When DIFF=0:It is single-ended 8-bit conversion; when DIFF=1,  

it is  

 *      differential 9-bit conversion with 2's complement output.  

 * - 0b01 - When DIFF=0:It is single-ended 12-bit conversion ; when  

DIFF=1, it  

 *      is differential 13-bit conversion with 2's complement output.  

 * - 0b10 - When DIFF=0:It is single-ended 10-bit conversion ; when  

DIFF=1, it  

 *      is differential 11-bit conversion with 2's complement output.  

 * - 0b11 - When DIFF=0:It is single-ended 16-bit conversion; when  

DIFF=1, it is  

 *      differential 16-bit conversion with 2's complement output.  

*/  

/*@{*/  

/*! @brief Read current value of the ADC_CFG1_MODE field. */  

#define ADC_RD_CFG1_MODE(base) ((ADC_CFG1_REG(base) & ADC_CFG1_MODE_MASK)  

>> ADC_CFG1_MODE_SHIFT)  

#define ADC_BRD_CFG1_MODE(base) (BME_UBFX32(&ADC_CFG1_REG(base),  

ADC_CFG1_MODE_SHIFT, ADC_CFG1_MODE_WIDTH))

/*! @brief Set the MODE field to a new value. */  

#define ADC_WR_CFG1_MODE(base, value) (ADC_RMW_CFG1(base,  

ADC_CFG1_MODE_MASK, ADC_CFG1_MODE(value)))  

#define ADC_BWR_CFG1_MODE(base, value) (BME_BFI32(&ADC_CFG1_REG(base),  

((uint32_t)(value) << ADC_CFG1_MODE_SHIFT), ADC_CFG1_MODE_SHIFT,  

ADC_CFG1_MODE_WIDTH))  

/*@}*/

/*!  

 * @name Register ADC_CFG1, field ADLSMP[4] (RW)  

 *  

 * ADLSMP selects between different sample times based on the conversion  

mode  

 * selected. This bit adjusts the sample period to allow higher impedance  

inputs to  

 * be accurately sampled or to maximize conversion speed for lower  

impedance  

 * inputs. Longer sample times can also be used to lower overall power  

consumption  

 * if continuous conversions are enabled and high conversion rates are  

not  

 * required. When ADLSMP=1, the long sample time select bits,  

(ADLSTS[1:0]), can select

```

```

* the extent of the long sample time.
*
* Values:
* - 0b0 - Short sample time.
* - 0b1 - Long sample time.
*/
/*@{ */
/*! @brief Read current value of the ADC_CFG1_ADLSMP field. */
#define ADC_RD_CFG1_ADLSMP(base) ((ADC_CFG1_REG(base) &
ADC_CFG1_ADLSMP_MASK) >> ADC_CFG1_ADLSMP_SHIFT)
#define ADC_BRD_CFG1_ADLSMP(base) (BME_UBFX32(&ADC_CFG1_REG(base),
ADC_CFG1_ADLSMP_SHIFT, ADC_CFG1_ADLSMP_WIDTH))

/*! @brief Set the ADLSMP field to a new value. */
#define ADC_WR_CFG1_ADLSMP(base, value) (ADC_RMW_CFG1(base,
ADC_CFG1_ADLSMP_MASK, ADC_CFG1_ADLSMP(value)))
#define ADC_BWR_CFG1_ADLSMP(base, value) (BME_BFI32(&ADC_CFG1_REG(base),
((uint32_t)(value) << ADC_CFG1_ADLSMP_SHIFT), ADC_CFG1_ADLSMP_SHIFT,
ADC_CFG1_ADLSMP_WIDTH))
/*@} */

/*!
* @name Register ADC_CFG1, field ADIV[6:5] (RW)
*
* ADIV selects the divide ratio used by the ADC to generate the internal
clock
* ADCK.
*
* Values:
* - 0b00 - The divide ratio is 1 and the clock rate is input clock.
* - 0b01 - The divide ratio is 2 and the clock rate is (input clock)/2.
* - 0b10 - The divide ratio is 4 and the clock rate is (input clock)/4.
* - 0b11 - The divide ratio is 8 and the clock rate is (input clock)/8.
*/
/*@{ */
/*! @brief Read current value of the ADC_CFG1_ADIV field. */
#define ADC_RD_CFG1_ADIV(base) ((ADC_CFG1_REG(base) & ADC_CFG1_ADIV_MASK)
>> ADC_CFG1_ADIV_SHIFT)
#define ADC_BRD_CFG1_ADIV(base) (BME_UBFX32(&ADC_CFG1_REG(base),
ADC_CFG1_ADIV_SHIFT, ADC_CFG1_ADIV_WIDTH))

/*! @brief Set the ADIV field to a new value. */
#define ADC_WR_CFG1_ADIV(base, value) (ADC_RMW_CFG1(base,
ADC_CFG1_ADIV_MASK, ADC_CFG1_ADIV(value)))
#define ADC_BWR_CFG1_ADIV(base, value) (BME_BFI32(&ADC_CFG1_REG(base),
((uint32_t)(value) << ADC_CFG1_ADIV_SHIFT), ADC_CFG1_ADIV_SHIFT,
ADC_CFG1_ADIV_WIDTH))
/*@} */

/*!
* @name Register ADC_CFG1, field ADLPC[7] (RW)
*
* Controls the power configuration of the successive approximation
converter.

```

```

 * This optimizes power consumption when higher sample rates are not
required.
 *
 * Values:
 * - 0b0 - Normal power configuration.
 * - 0b1 - Low-power configuration. The power is reduced at the expense
of
 *      maximum clock speed.
 */
/*@{*/
/*! @brief Read current value of the ADC_CFG1_ADLPC field. */
#define ADC_RD_CFG1_ADLPC(base) ((ADC_CFG1_REG(base) &
ADC_CFG1_ADLPC_MASK) >> ADC_CFG1_ADLPC_SHIFT)
#define ADC_BRD_CFG1_ADLPC(base) (BME_UBFX32(&ADC_CFG1_REG(base),
ADC_CFG1_ADLPC_SHIFT, ADC_CFG1_ADLPC_WIDTH))

/*! @brief Set the ADLPC field to a new value. */
#define ADC_WR_CFG1_ADLPC(base, value) (ADC_RMW_CFG1(base,
ADC_CFG1_ADLPC_MASK, ADC_CFG1_ADLPC(value)))
#define ADC_BWR_CFG1_ADLPC(base, value) (BME_BFI32(&ADC_CFG1_REG(base),
((uint32_t)(value) << ADC_CFG1_ADLPC_SHIFT), ADC_CFG1_ADLPC_SHIFT,
ADC_CFG1_ADLPC_WIDTH))
/*}@*/ */

/********************* ADC_CFG2 - ADC Configuration Register 2 *****************/
***** */
 * ADC_CFG2 - ADC Configuration Register 2
***** */

/*!
 * @brief ADC_CFG2 - ADC Configuration Register 2 (RW)
 *
 * Reset value: 0x00000000U
 *
 * Configuration Register 2 (CFG2) selects the special high-speed
configuration
 * for very high speed conversions and selects the long sample time
duration
 * during long sample mode.
 */
/*!
 * @name Constants and macros for entire ADC_CFG2 register
 */
/*@{*/
#define ADC_RD_CFG2(base) (ADC_CFG2_REG(base))
#define ADC_WR_CFG2(base, value) (ADC_CFG2_REG(base) = (value))
#define ADC_RMW_CFG2(base, mask, value) (ADC_WR_CFG2(base,
(ADC_RD_CFG2(base) & ~mask) | (value)))
#define ADC_SET_CFG2(base, value) (BME_OR32(&ADC_CFG2_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CFG2(base, value) (BME_AND32(&ADC_CFG2_REG(base),
(uint32_t)(~(value))))

```

```

#define ADC_TOG_CFG2(base, value)  (BME_XOR32(&ADC_CFG2_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual ADC_CFG2 bitfields
 */

/*!  

 * @name Register ADC_CFG2, field ADLSTS[1:0] (RW)  

 *  

 * Selects between the extended sample times when long sample time is  

selected,  

 * that is, when CFG1[ADLSMP]=1. This allows higher impedance inputs to  

be  

 * accurately sampled or to maximize conversion speed for lower impedance  

inputs.  

 * Longer sample times can also be used to lower overall power  

consumption when  

 * continuous conversions are enabled if high conversion rates are not  

required.  

 *  

 * Values:  

 * - 0b00 - Default longest sample time; 20 extra ADCK cycles; 24 ADCK  

cycles  

 * total.  

 * - 0b01 - 12 extra ADCK cycles; 16 ADCK cycles total sample time.  

 * - 0b10 - 6 extra ADCK cycles; 10 ADCK cycles total sample time.  

 * - 0b11 - 2 extra ADCK cycles; 6 ADCK cycles total sample time.  

*/
/*@{*/  

/*! @brief Read current value of the ADC_CFG2_ADLSTS field. */  

#define ADC_RD_CFG2_ADLSTS(base) ((ADC_CFG2_REG(base) &  

ADC_CFG2_ADLSTS_MASK) >> ADC_CFG2_ADLSTS_SHIFT)  

#define ADC_BRD_CFG2_ADLSTS(base) (BME_UBFX32(&ADC_CFG2_REG(base),  

ADC_CFG2_ADLSTS_SHIFT, ADC_CFG2_ADLSTS_WIDTH))  

  

/*! @brief Set the ADLSTS field to a new value. */  

#define ADC_WR_CFG2_ADLSTS(base, value) (ADC_RMW_CFG2(base,  

ADC_CFG2_ADLSTS_MASK, ADC_CFG2_ADLSTS(value)))  

#define ADC_BWR_CFG2_ADLSTS(base, value) (BME_BFI32(&ADC_CFG2_REG(base),  

((uint32_t)(value) << ADC_CFG2_ADLSTS_SHIFT), ADC_CFG2_ADLSTS_SHIFT,  

ADC_CFG2_ADLSTS_WIDTH))  

/*@}*/  

  

/*!  

 * @name Register ADC_CFG2, field ADHSC[2] (RW)  

 *  

 * Configures the ADC for very high-speed operation. The conversion  

sequence is  

 * altered with 2 ADCK cycles added to the conversion time to allow  

higher speed  

 * conversion clocks.  

 *

```

```

* Values:
* - 0b0 - Normal conversion sequence selected.
* - 0b1 - High-speed conversion sequence selected with 2 additional ADCK
cycles
*      to total conversion time.
*/
/*@{*/
/*! @brief Read current value of the ADC_CFG2_ADHSC field. */
#define ADC_RD_CFG2_ADHSC(base) ((ADC_CFG2_REG(base) &
ADC_CFG2_ADHSC_MASK) >> ADC_CFG2_ADHSC_SHIFT)
#define ADC_BRD_CFG2_ADHSC(base) (BME_UBFX32(&ADC_CFG2_REG(base),
ADC_CFG2_ADHSC_SHIFT, ADC_CFG2_ADHSC_WIDTH))

/*! @brief Set the ADHSC field to a new value. */
#define ADC_WR_CFG2_ADHSC(base, value) (ADC_RMW_CFG2(base,
ADC_CFG2_ADHSC_MASK, ADC_CFG2_ADHSC(value)))
#define ADC_BWR_CFG2_ADHSC(base, value) (BME_BFI32(&ADC_CFG2_REG(base),
((uint32_t)(value) << ADC_CFG2_ADHSC_SHIFT), ADC_CFG2_ADHSC_SHIFT,
ADC_CFG2_ADHSC_WIDTH))
/*@}*/
/*!
* @name Register ADC_CFG2, field ADACKEN[3] (RW)
*
* Enables the asynchronous clock source and the clock source output
regardless
* of the conversion and status of CFG1[ADICLK]. Based on MCU
configuration, the
* asynchronous clock may be used by other modules. See chip
configuration
* information. Setting this field allows the clock to be used even while
the ADC is
* idle or operating from a different clock source. Also, latency of
initiating a
* single or first-continuous conversion with the asynchronous clock
selected is
* reduced because the ADACK clock is already operational.
*
* Values:
* - 0b0 - Asynchronous clock output disabled; Asynchronous clock is
enabled
*      only if selected by ADICLK and a conversion is active.
* - 0b1 - Asynchronous clock and clock output is enabled regardless of
the
*      state of the ADC.
*/
/*@{*/
/*! @brief Read current value of the ADC_CFG2_ADACKEN field. */
#define ADC_RD_CFG2_ADACKEN(base) ((ADC_CFG2_REG(base) &
ADC_CFG2_ADACKEN_MASK) >> ADC_CFG2_ADACKEN_SHIFT)
#define ADC_BRD_CFG2_ADACKEN(base) (BME_UBFX32(&ADC_CFG2_REG(base),
ADC_CFG2_ADACKEN_SHIFT, ADC_CFG2_ADACKEN_WIDTH))

/*! @brief Set the ADACKEN field to a new value. */

```

```

#define ADC_WR_CFG2_ADACKEN(base, value) (ADC_RMW_CFG2(base,
ADC_CFG2_ADACKEN_MASK, ADC_CFG2_ADACKEN(value)))
#define ADC_BWR_CFG2_ADACKEN(base, value) (BME_BFI32(&ADC_CFG2_REG(base),
((uint32_t)(value) << ADC_CFG2_ADACKEN_SHIFT), ADC_CFG2_ADACKEN_SHIFT,
ADC_CFG2_ADACKEN_WIDTH))
/*@}*/

/*
 * @name Register ADC_CFG2, field MUXSEL[4] (RW)
 *
 * Changes the ADC mux setting to select between alternate sets of ADC
channels.
 *
 * Values:
 * - 0b0 - ADxxa channels are selected.
 * - 0b1 - ADxxb channels are selected.
 */
/*@*/
/*! @brief Read current value of the ADC_CFG2_MUXSEL field. */
#define ADC_RD_CFG2_MUXSEL(base) ((ADC_CFG2_REG(base) &
ADC_CFG2_MUXSEL_MASK) >> ADC_CFG2_MUXSEL_SHIFT)
#define ADC_BRD_CFG2_MUXSEL(base) (BME_UBFX32(&ADC_CFG2_REG(base),
ADC_CFG2_MUXSEL_SHIFT, ADC_CFG2_MUXSEL_WIDTH))

/*! @brief Set the MUXSEL field to a new value. */
#define ADC_WR_CFG2_MUXSEL(base, value) (ADC_RMW_CFG2(base,
ADC_CFG2_MUXSEL_MASK, ADC_CFG2_MUXSEL(value)))
#define ADC_BWR_CFG2_MUXSEL(base, value) (BME_BFI32(&ADC_CFG2_REG(base),
((uint32_t)(value) << ADC_CFG2_MUXSEL_SHIFT), ADC_CFG2_MUXSEL_SHIFT,
ADC_CFG2_MUXSEL_WIDTH))
/*@*/

*****
* ADC_R - ADC Data Result Register
*****
 */

/*!
 * @brief ADC_R - ADC Data Result Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * The data result registers (Rn) contain the result of an ADC conversion
of the
 * channel selected by the corresponding status and channel control
register
 * (SC1A:SC1n). For every status and channel control register, there is a
 * corresponding data result register. Unused bits in R n are cleared in
unsigned
 * right-justified modes and carry the sign bit (MSB) in sign-extended
2's complement

```

```

 * modes. For example, when configured for 10-bit single-ended mode,
D[15:10] are
 * cleared. When configured for 11-bit differential mode, D[15:10] carry
the sign
 * bit, that is, bit 10 extended through bit 15. The following table
describes the
 * behavior of the data result registers in the different modes of
operation.
 * Data result register description Conversion mode D15 D14 D13 D12 D11
D10 D9 D8 D7
 * D6 D5 D4 D3 D2 D1 D0 Format 16-bit differential S D D D D D D D D D D D
D D D D
 * D Signed 2's complement 16-bit single-ended D D D D D D D D D D D D D D D
D D D
 * Unsigned right justified 13-bit differential S S S S D D D D D D D D D D D
D D D
 * Sign-extended 2's complement 12-bit single-ended 0 0 0 0 D D D D D D D D D
D D D D
 * Unsigned right-justified 11-bit differential S S S S S S S D D D D D D D D
D D D
 * Sign-extended 2's complement 10-bit single-ended 0 0 0 0 0 0 D D D D D D D
D D D D
 * D Unsigned right-justified 9-bit differential S S S S S S S S D D D D D D D
D D D D
 * Sign-extended 2's complement 8-bit single-ended 0 0 0 0 0 0 0 0 D D D
D D D
 * D Unsigned right-justified S: Sign bit or sign bit extension; D: Data,
which
 * is 2's complement data if indicated
 */
/*!
 * @name Constants and macros for entire ADC_R register
 */
/*@{*/
#define ADC_RD_R(base, index)      (ADC_R_REG(base, index))
/*}@*/



/*
 * Constants & macros for individual ADC_R bitfields
 */

/*!
 * @name Register ADC_R, field D[15:0] (RO)
 */
/*@{*/
/*! @brief Read current value of the ADC_R_D field. */
#define ADC_RD_R_D(base, index) ((ADC_R_REG(base, index) & ADC_R_D_MASK)
>> ADC_R_D_SHIFT)
#define ADC_BRD_R_D(base, index) (BME_UBFX32(&ADC_R_REG(base, index),
ADC_R_D_SHIFT, ADC_R_D_WIDTH))
/*}@*/



/*********************
```

```

 * ADC_CV1 - Compare Value Registers
 ****
 **** */

/*!
 * @brief ADC_CV1 - Compare Value Registers (RW)
 *
 * Reset value: 0x00000000U
 *
 * The compare value registers (CV1 and CV2) contain a compare value used
 * to
 * compare the conversion result when the compare function is enabled,
 * that is,
 * SC2[ACFE]=1. This register is formatted in the same way as the Rn
 * registers in
 * different modes of operation for both bit position definition and
 * value format
 * using unsigned or sign-extended 2's complement. Therefore, the compare
 * function
 * uses only the CVn fields that are related to the ADC mode of
 * operation. The
 * compare value 2 register (CV2) is used only when the compare range
 * function is
 * enabled, that is, SC2[ACREN]=1.
 */
/*!
 * @name Constants and macros for entire ADC_CV1 register
 */
/*@{*/
#define ADC_RD_CV1(base)          (ADC_CV1_REG(base))
#define ADC_WR_CV1(base, value)    (ADC_CV1_REG(base) = (value))
#define ADC_RMW_CV1(base, mask, value) (ADC_WR_CV1(base,
(ADC_RD_CV1(base) & ~mask) | (value)))
#define ADC_SET_CV1(base, value)   (BME_OR32(&ADC_CV1_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CV1(base, value)   (BME_AND32(&ADC_CV1_REG(base),
(uint32_t)(~value)))
#define ADC_TOG_CV1(base, value)   (BME_XOR32(&ADC_CV1_REG(base),
(uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual ADC_CV1 bitfields
 */

/*!
 * @name Register ADC_CV1, field CV[15:0] (RW)
 */
/*@{*/
/*! @brief Read current value of the ADC_CV1_CV field. */
#define ADC_RD_CV1_CV(base) ((ADC_CV1_REG(base) & ADC_CV1_CV_MASK) >>
ADC_CV1_CV_SHIFT)

```

```

#define ADC_BRD_CV1_CV(base)  (BME_UBFX32(&ADC_CV1_REG(base),  

ADC_CV1_CV_SHIFT, ADC_CV1_CV_WIDTH))

/*! @brief Set the CV field to a new value. */  

#define ADC_WR_CV1_CV(base, value)  (ADC_RMW_CV1(base, ADC_CV1_CV_MASK,  

ADC_CV1_CV(value)))  

#define ADC_BWR_CV1_CV(base, value)  (BME_BFI32(&ADC_CV1_REG(base),  

(uint32_t)(value) << ADC_CV1_CV_SHIFT), ADC_CV1_CV_SHIFT,  

ADC_CV1_CV_WIDTH)  

/*@}*/

/*********************  

*****  

 * ADC_CV2 - Compare Value Registers  

*****/  

/*********************  

*****/  

  

/*!  

 * @brief ADC_CV2 - Compare Value Registers (RW)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The compare value registers (CV1 and CV2) contain a compare value used  

to  

 * compare the conversion result when the compare function is enabled,  

that is,  

 * SC2[ACFE]=1. This register is formatted in the same way as the Rn  

registers in  

 * different modes of operation for both bit position definition and  

value format  

 * using unsigned or sign-extended 2's complement. Therefore, the compare  

function  

 * uses only the CVn fields that are related to the ADC mode of  

operation. The  

 * compare value 2 register (CV2) is used only when the compare range  

function is  

 * enabled, that is, SC2[ACREN]=1.  

 */  

/*!  

 * @name Constants and macros for entire ADC_CV2 register  

 */  

/*@{*/  

#define ADC_RD_CV2(base)          (ADC_CV2_REG(base))  

#define ADC_WR_CV2(base, value)    (ADC_CV2_REG(base) = (value))  

#define ADC_RMW_CV2(base, mask, value)  (ADC_WR_CV2(base,  

(ADC_RD_CV2(base) & ~mask) | (value)))  

#define ADC_SET_CV2(base, value)   (BME_OR32(&ADC_CV2_REG(base),  

(uint32_t)(value)))  

#define ADC_CLR_CV2(base, value)   (BME_AND32(&ADC_CV2_REG(base),  

(uint32_t)(~value)))  

#define ADC_TOG_CV2(base, value)   (BME_XOR32(&ADC_CV2_REG(base),  

(uint32_t)(value)))  

/*@}*/
```

```

/*
 * Constants & macros for individual ADC_CV2 bitfields
 */

/*!
 * @name Register ADC_CV2, field CV[15:0] (RW)
 */
/*@{*/
/*! @brief Read current value of the ADC_CV2_CV field. */
#define ADC_RD_CV2_CV(base) ((ADC_CV2_REG(base) & ADC_CV2_CV_MASK) >>
ADC_CV2_CV_SHIFT)
#define ADC_BRD_CV2_CV(base) (BME_UBXFX32(&ADC_CV2_REG(base),
ADC_CV2_CV_SHIFT, ADC_CV2_CV_WIDTH))

/*! @brief Set the CV field to a new value. */
#define ADC_WR_CV2_CV(base, value) (ADC_RMW_CV2(base, ADC_CV2_CV_MASK,
ADC_CV2_CV(value)))
#define ADC_BWR_CV2_CV(base, value) (BME_BFI32(&ADC_CV2_REG(base),
((uint32_t)(value) << ADC_CV2_CV_SHIFT), ADC_CV2_CV_SHIFT,
ADC_CV2_CV_WIDTH))
/*@}*/
*****  

* ADC_SC2 - Status and Control Register 2  

*****  

/*!
 * @brief ADC_SC2 - Status and Control Register 2 (RW)
 *
 * Reset value: 0x00000000U
 *
 * The status and control register 2 (SC2) contains the conversion
active,
 * hardware/software trigger select, compare function, and voltage
reference select of
 * the ADC module.
 */
/*!
 * @name Constants and macros for entire ADC_SC2 register
 */
/*@{*/
#define ADC_RD_SC2(base) (ADC_SC2_REG(base))
#define ADC_WR_SC2(base, value) (ADC_SC2_REG(base) = (value))
#define ADC_RMW_SC2(base, mask, value) (ADC_WR_SC2(base,
(ADC_RD_SC2(base) & ~mask) | (value)))
#define ADC_SET_SC2(base, value) (BME_OR32(&ADC_SC2_REG(base),
(uint32_t)(value)))
#define ADC_CLR_SC2(base, value) (BME_AND32(&ADC_SC2_REG(base),
(uint32_t)(~(value))))

```

```

#define ADC_TOG_SC2(base, value) (BME_XOR32(&ADC_SC2_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual ADC_SC2 bitfields
 */

/*!  

 * @name Register ADC_SC2, field REFSEL[1:0] (RW)  

 *  

 * Selects the voltage reference source used for conversions.  

 *  

 * Values:  

 * - 0b00 - Default voltage reference pin pair, that is, external pins  

VREFH and  

*      VREFL  

* - 0b01 - Alternate reference pair, that is, VALTH and VALTL . This  

pair may  

*      be additional external pins or internal sources depending on the  

MCU  

*      configuration. See the chip configuration information for details  

specific to  

*      this MCU  

* - 0b10 - Reserved  

* - 0b11 - Reserved  

*/
/*@{*/  

/*! @brief Read current value of the ADC_SC2_REFSEL field. */  

#define ADC_RD_SC2_REFSEL(base) ((ADC_SC2_REG(base) &  

ADC_SC2_REFSEL_MASK) >> ADC_SC2_REFSEL_SHIFT)  

#define ADC_BRD_SC2_REFSEL(base) (BME_UBFX32(&ADC_SC2_REG(base),  

ADC_SC2_REFSEL_SHIFT, ADC_SC2_REFSEL_WIDTH))  

  

/*! @brief Set the REFSEL field to a new value. */  

#define ADC_WR_SC2_REFSEL(base, value) (ADC_RMW_SC2(base,  

ADC_SC2_REFSEL_MASK, ADC_SC2_REFSEL(value)))  

#define ADC_BWR_SC2_REFSEL(base, value) (BME_BFI32(&ADC_SC2_REG(base),  

((uint32_t)(value) << ADC_SC2_REFSEL_SHIFT), ADC_SC2_REFSEL_SHIFT,  

ADC_SC2_REFSEL_WIDTH))  

/*@}*/

/*
 * @name Register ADC_SC2, field DMAEN[2] (RW)  

 *  

 * Values:  

 * - 0b0 - DMA is disabled.  

 * - 0b1 - DMA is enabled and will assert the ADC DMA request during an  

ADC  

*      conversion complete event noted when any of the SC1n[COCO] flags  

is asserted.  

*/
/*@{*/  

/*! @brief Read current value of the ADC_SC2_DMAEN field. */

```

```

#define ADC_RD_SC2_DMAEN(base) ((ADC_SC2_REG(base) & ADC_SC2_DMAEN_MASK)
>> ADC_SC2_DMAEN_SHIFT)
#define ADC_BRD_SC2_DMAEN(base) (BME_UBFX32(&ADC_SC2_REG(base),
ADC_SC2_DMAEN_SHIFT, ADC_SC2_DMAEN_WIDTH))

/*! @brief Set the DMAEN field to a new value. */
#define ADC_WR_SC2_DMAEN(base, value) (ADC_RMW_SC2(base,
ADC_SC2_DMAEN_MASK, ADC_SC2_DMAEN(value)))
#define ADC_BWR_SC2_DMAEN(base, value) (BME_BFI32(&ADC_SC2_REG(base),
((uint32_t)(value) << ADC_SC2_DMAEN_SHIFT), ADC_SC2_DMAEN_SHIFT,
ADC_SC2_DMAEN_WIDTH))
/*@}*/

/*!
 * @name Register ADC_SC2, field ACREN[3] (RW)
 *
 * Configures the compare function to check if the conversion result of
the
 * input being monitored is either between or outside the range formed by
CV1 and CV2
 * determined by the value of ACFGTE. ACFE must be set for ACFGTE to have
any
 * effect.
 *
 * Values:
 * - 0b0 - Range function disabled. Only CV1 is compared.
 * - 0b1 - Range function enabled. Both CV1 and CV2 are compared.
 */
/*@*/
/*! @brief Read current value of the ADC_SC2_ACREN field. */
#define ADC_RD_SC2_ACREN(base) ((ADC_SC2_REG(base) & ADC_SC2_ACREN_MASK)
>> ADC_SC2_ACREN_SHIFT)
#define ADC_BRD_SC2_ACREN(base) (BME_UBFX32(&ADC_SC2_REG(base),
ADC_SC2_ACREN_SHIFT, ADC_SC2_ACREN_WIDTH))

/*! @brief Set the ACREN field to a new value. */
#define ADC_WR_SC2_ACREN(base, value) (ADC_RMW_SC2(base,
ADC_SC2_ACREN_MASK, ADC_SC2_ACREN(value)))
#define ADC_BWR_SC2_ACREN(base, value) (BME_BFI32(&ADC_SC2_REG(base),
((uint32_t)(value) << ADC_SC2_ACREN_SHIFT), ADC_SC2_ACREN_SHIFT,
ADC_SC2_ACREN_WIDTH))
/*@*/

/*!
 * @name Register ADC_SC2, field ACFGTE[4] (RW)
 *
 * Configures the compare function to check the conversion result
relative to
 * the CV1 and CV2 based upon the value of ACREN. ACFE must be set for
ACFGTE to
 * have any effect.
 *
 * Values:

```

```

* - 0b0 - Configures less than threshold, outside range not inclusive
and
*      inside range not inclusive; functionality based on the values
placed in CV1 and
*      CV2.
* - 0b1 - Configures greater than or equal to threshold, outside and
inside
*      ranges inclusive; functionality based on the values placed in CV1
and CV2.
*/
/*@{*/
/*! @brief Read current value of the ADC_SC2_ACFGT field. */
#define ADC_RD_SC2_ACFGT(base) ((ADC_SC2_REG(base) & ADC_SC2_ACFGT_MASK)
>> ADC_SC2_ACFGT_SHIFT)
#define ADC_BRD_SC2_ACFGT(base) (BME_UBXF32(&ADC_SC2_REG(base),
ADC_SC2_ACFGT_SHIFT, ADC_SC2_ACFGT_WIDTH))

/*! @brief Set the ACFGT field to a new value. */
#define ADC_WR_SC2_ACFGT(base, value) (ADC_RMW_SC2(base,
ADC_SC2_ACFGT_MASK, ADC_SC2_ACFGT(value)))
#define ADC_BWR_SC2_ACFGT(base, value) (BME_BFI32(&ADC_SC2_REG(base),
((uint32_t)(value) << ADC_SC2_ACFGT_SHIFT), ADC_SC2_ACFGT_SHIFT,
ADC_SC2_ACFGT_WIDTH))
/*}@*/ */

/*!
* @name Register ADC_SC2, field ACFE[5] (RW)
*
* Enables the compare function.
*
* Values:
* - 0b0 - Compare function disabled.
* - 0b1 - Compare function enabled.
*/
/*@{*/
/*! @brief Read current value of the ADC_SC2_ACFE field. */
#define ADC_RD_SC2_ACFE(base) ((ADC_SC2_REG(base) & ADC_SC2_ACFE_MASK) >>
ADC_SC2_ACFE_SHIFT)
#define ADC_BRD_SC2_ACFE(base) (BME_UBXF32(&ADC_SC2_REG(base),
ADC_SC2_ACFE_SHIFT, ADC_SC2_ACFE_WIDTH))

/*! @brief Set the ACFE field to a new value. */
#define ADC_WR_SC2_ACFE(base, value) (ADC_RMW_SC2(base,
ADC_SC2_ACFE_MASK, ADC_SC2_ACFE(value)))
#define ADC_BWR_SC2_ACFE(base, value) (BME_BFI32(&ADC_SC2_REG(base),
((uint32_t)(value) << ADC_SC2_ACFE_SHIFT), ADC_SC2_ACFE_SHIFT,
ADC_SC2_ACFE_WIDTH))
/*}@*/ */

/*!
* @name Register ADC_SC2, field ADTRG[6] (RW)
*
* Selects the type of trigger used for initiating a conversion. Two
types of

```

```

    * trigger are selectable: Software trigger: When software trigger is
selected, a
    * conversion is initiated following a write to SC1A. Hardware trigger:
When
    * hardware trigger is selected, a conversion is initiated following the
assertion of
    * the ADHWT input after a pulse of the ADHWTn input.
    *
    * Values:
    * - 0b0 - Software trigger selected.
    * - 0b1 - Hardware trigger selected.
    */
/*@{*/
/*! @brief Read current value of the ADC_SC2_ADTRG field. */
#define ADC_RD_SC2_ADTRG(base) ((ADC_SC2_REG(base) & ADC_SC2_ADTRG_MASK)
>> ADC_SC2_ADTRG_SHIFT)
#define ADC_BRD_SC2_ADTRG(base) (BME_UBFX32(&ADC_SC2_REG(base),
ADC_SC2_ADTRG_SHIFT, ADC_SC2_ADTRG_WIDTH))

/*! @brief Set the ADTRG field to a new value. */
#define ADC_WR_SC2_ADTRG(base, value) (ADC_RMW_SC2(base,
ADC_SC2_ADTRG_MASK, ADC_SC2_ADTRG(value)))
#define ADC_BWR_SC2_ADTRG(base, value) (BME_BFI32(&ADC_SC2_REG(base),
((uint32_t)(value) << ADC_SC2_ADTRG_SHIFT), ADC_SC2_ADTRG_SHIFT,
ADC_SC2_ADTRG_WIDTH))
/*@}*/

/*
    * @name Register ADC_SC2, field ADACT[7] (RO)
    *
    * Indicates that a conversion or hardware averaging is in progress.
ADACT is
    * set when a conversion is initiated and cleared when a conversion is
completed or
    * aborted.
    *
    * Values:
    * - 0b0 - Conversion not in progress.
    * - 0b1 - Conversion in progress.
    */
/*@{*/
/*! @brief Read current value of the ADC_SC2_ADACT field. */
#define ADC_RD_SC2_ADACT(base) ((ADC_SC2_REG(base) & ADC_SC2_ADACT_MASK)
>> ADC_SC2_ADACT_SHIFT)
#define ADC_BRD_SC2_ADACT(base) (BME_UBFX32(&ADC_SC2_REG(base),
ADC_SC2_ADACT_SHIFT, ADC_SC2_ADACT_WIDTH))
/*@}*/

*****
* ADC_SC3 - Status and Control Register 3
*****
*****/

```

```

/*!
 * @brief ADC_SC3 - Status and Control Register 3 (RW)
 *
 * Reset value: 0x00000000U
 *
 * The Status and Control Register 3 (SC3) controls the calibration,
 * continuous
 * convert, and hardware averaging functions of the ADC module.
 */
/*!
 * @name Constants and macros for entire ADC_SC3 register
 */
/*@{*/
#define ADC_RD_SC3(base)          (ADC_SC3_REG(base))
#define ADC_WR_SC3(base, value)   (ADC_SC3_REG(base) = (value))
#define ADC_RMW_SC3(base, mask, value) (ADC_WR_SC3(base,
(ADC_RD_SC3(base) & ~mask) | (value)))
#define ADC_SET_SC3(base, value)  (BME_OR32(&ADC_SC3_REG(base),
(uint32_t)(value)))
#define ADC_CLR_SC3(base, value)  (BME_AND32(&ADC_SC3_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_SC3(base, value)  (BME_XOR32(&ADC_SC3_REG(base),
(uint32_t)(value)))
/*}@*/
/*
 * Constants & macros for individual ADC_SC3 bitfields
 */

/*!
 * @name Register ADC_SC3, field AVGS[1:0] (RW)
 *
 * Determines how many ADC conversions will be averaged to create the ADC
 * average result.
 *
 * Values:
 * - 0b00 - 4 samples averaged.
 * - 0b01 - 8 samples averaged.
 * - 0b10 - 16 samples averaged.
 * - 0b11 - 32 samples averaged.
 */
/*@{*/
/*! @brief Read current value of the ADC_SC3_AVGS field. */
#define ADC_RD_SC3_AVGS(base) ((ADC_SC3_REG(base) & ADC_SC3_AVGS_MASK) >>
ADC_SC3_AVGS_SHIFT)
#define ADC_BRD_SC3_AVGS(base) (BME_UBFX32(&ADC_SC3_REG(base),
ADC_SC3_AVGS_SHIFT, ADC_SC3_AVGS_WIDTH))

/*! @brief Set the AVGS field to a new value. */
#define ADC_WR_SC3_AVGS(base, value) (ADC_RMW_SC3(base,
(ADC_SC3_AVGS_MASK | ADC_SC3_CALF_MASK), ADC_SC3_AVGS(value)))

```

```

#define ADC_BWR_SC3_AVGS(base, value) (BME_BFI32(&ADC_SC3_REG(base),  

((uint32_t)(value) << ADC_SC3_AVGS_SHIFT), ADC_SC3_AVGS_SHIFT,  

ADC_SC3_AVGS_WIDTH))  

/*@}*/

/*!  

 * @name Register ADC_SC3, field AVGE[2] (RW)  

 *  

 * Enables the hardware average function of the ADC.  

 *  

 * Values:  

 * - 0b0 - Hardware average function disabled.  

 * - 0b1 - Hardware average function enabled.  

 */  

/*@{*/  

/*! @brief Read current value of the ADC_SC3_AVGE field. */  

#define ADC_RD_SC3_AVGE(base) ((ADC_SC3_REG(base) & ADC_SC3_AVGE_MASK) >>  

ADC_SC3_AVGE_SHIFT)  

#define ADC_BRD_SC3_AVGE(base) (BME_UBFX32(&ADC_SC3_REG(base),  

ADC_SC3_AVGE_SHIFT, ADC_SC3_AVGE_WIDTH))

/*! @brief Set the AVGE field to a new value. */  

#define ADC_WR_SC3_AVGE(base, value) (ADC_RMW_SC3(base,  

(ADC_SC3_AVGE_MASK | ADC_SC3_CALF_MASK), ADC_SC3_AVGE(value)))  

#define ADC_BWR_SC3_AVGE(base, value) (BME_BFI32(&ADC_SC3_REG(base),  

((uint32_t)(value) << ADC_SC3_AVGE_SHIFT), ADC_SC3_AVGE_SHIFT,  

ADC_SC3_AVGE_WIDTH))  

/*@}*/

/*!  

 * @name Register ADC_SC3, field ADCO[3] (RW)  

 *  

 * Enables continuous conversions.  

 *  

 * Values:  

 * - 0b0 - One conversion or one set of conversions if the hardware  

average  

*      function is enabled, that is, AVGE=1, after initiating a  

conversion.  

* - 0b1 - Continuous conversions or sets of conversions if the hardware  

average  

*      function is enabled, that is, AVGE=1, after initiating a  

conversion.  

 */  

/*@{*/  

/*! @brief Read current value of the ADC_SC3_ADCO field. */  

#define ADC_RD_SC3_ADCO(base) ((ADC_SC3_REG(base) & ADC_SC3_ADCO_MASK) >>  

ADC_SC3_ADCO_SHIFT)  

#define ADC_BRD_SC3_ADCO(base) (BME_UBFX32(&ADC_SC3_REG(base),  

ADC_SC3_ADCO_SHIFT, ADC_SC3_ADCO_WIDTH))

/*! @brief Set the ADCO field to a new value. */  

#define ADC_WR_SC3_ADCO(base, value) (ADC_RMW_SC3(base,  

(ADC_SC3_ADCO_MASK | ADC_SC3_CALF_MASK), ADC_SC3_ADCO(value)))

```

```

#define ADC_BWR_SC3_ADCO(base, value) (BME_BFI32(&ADC_SC3_REG(base),  

((uint32_t)(value) << ADC_SC3_ADCO_SHIFT), ADC_SC3_ADCO_SHIFT,  

ADC_SC3_ADCO_WIDTH))  

/*@}*/

/*!  

 * @name Register ADC_SC3, field CALF[6] (W1C)  

 *  

 * Displays the result of the calibration sequence. The calibration  

sequence  

 * will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop  

mode is  

 * entered before the calibration sequence completes. Writing 1 to CALF  

clears it.  

 *  

 * Values:  

 * - 0b0 - Calibration completed normally.  

 * - 0b1 - Calibration failed. ADC accuracy specifications are not  

guaranteed.  

 */  

/*@*/  

/*! @brief Read current value of the ADC_SC3_CALF field. */  

#define ADC_RD_SC3_CALF(base) ((ADC_SC3_REG(base) & ADC_SC3_CALF_MASK) >>  

ADC_SC3_CALF_SHIFT)  

#define ADC_BRD_SC3_CALF(base) (BME_UBFX32(&ADC_SC3_REG(base),  

ADC_SC3_CALF_SHIFT, ADC_SC3_CALF_WIDTH))

/*! @brief Set the CALF field to a new value. */  

#define ADC_WR_SC3_CALF(base, value) (ADC_RMW_SC3(base,  

ADC_SC3_CALF_MASK, ADC_SC3_CALF(value)))  

#define ADC_BWR_SC3_CALF(base, value) (BME_BFI32(&ADC_SC3_REG(base),  

((uint32_t)(value) << ADC_SC3_CALF_SHIFT), ADC_SC3_CALF_SHIFT,  

ADC_SC3_CALF_WIDTH))  

/*@*/
```

```

/*!  

 * @name Register ADC_SC3, field CAL[7] (RW)  

 *  

 * Begins the calibration sequence when set. This field stays set while  

the  

 * calibration is in progress and is cleared when the calibration  

sequence is  

 * completed. CALF must be checked to determine the result of the  

calibration sequence.  

 * Once started, the calibration routine cannot be interrupted by writes  

to the  

 * ADC registers or the results will be invalid and CALF will set.  

Setting CAL  

 * will abort any current conversion.  

 */  

/*@*/  

/*! @brief Read current value of the ADC_SC3_CAL field. */  

#define ADC_RD_SC3_CAL(base) ((ADC_SC3_REG(base) & ADC_SC3_CAL_MASK) >>  

ADC_SC3_CAL_SHIFT)
```

```

#define ADC_BRD_SC3_CAL(base)  (BME_UBFX32(&ADC_SC3_REG(base),  

ADC_SC3_CAL_SHIFT, ADC_SC3_CAL_WIDTH))

/*! @brief Set the CAL field to a new value. */  

#define ADC_WR_SC3_CAL(base, value)  (ADC_RMW_SC3(base, (ADC_SC3_CAL_MASK  

| ADC_SC3_CALF_MASK), ADC_SC3_CAL(value)))  

#define ADC_BWR_SC3_CAL(base, value)  (BME_BFI32(&ADC_SC3_REG(base),  

(uint32_t)(value) << ADC_SC3_CAL_SHIFT), ADC_SC3_CAL_SHIFT,  

ADC_SC3_CAL_WIDTH))  

/*@}*/

/*********************  

*****  

 * ADC_OFS - ADC Offset Correction Register  

*****  

/*********************  

****/  

  

/*!  

 * @brief ADC_OFS - ADC Offset Correction Register (RW)  

 *  

 * Reset value: 0x00000004U  

 *  

 * The ADC Offset Correction Register (OFS) contains the user-selected or  

 * calibration-generated offset error correction value. This register is  

a 2's  

 * complement, left-justified, 16-bit value . The value in OFS is  

subtracted from the  

 * conversion and the result is transferred into the result registers,  

Rn. If the  

 * result is greater than the maximum or less than the minimum result  

value, it is  

 * forced to the appropriate limit for the current mode of operation.  

 */  

/*!  

 * @name Constants and macros for entire ADC_OFS register  

 */  

/*@*/  

#define ADC_RD_OFS(base)          (ADC_OFS_REG(base))  

#define ADC_WR_OFS(base, value)   (ADC_OFS_REG(base) = (value))  

#define ADC_RMW_OFS(base, mask, value) (ADC_WR_OFS(base,  

(ADC_RD_OFS(base) & ~mask) | (value)))  

#define ADC_SET_OFS(base, value)  (BME_OR32(&ADC_OFS_REG(base),  

(uint32_t)(value)))  

#define ADC_CLR_OFS(base, value)  (BME_AND32(&ADC_OFS_REG(base),  

(uint32_t)(~(value))))  

#define ADC_TOG_OFS(base, value)  (BME_XOR32(&ADC_OFS_REG(base),  

(uint32_t)(value)))  

/*@*/  

  

/*
 * Constants & macros for individual ADC_OFS bitfields
 */

```

```

/*!
 * @name Register ADC_OFS, field OFS[15:0] (RW)
 */
/*@{*/
/*! @brief Read current value of the ADC_OFS_OFS field. */
#define ADC_RD_OFS_OFS(base) ((ADC_OFS_REG(base) & ADC_OFS_OFS_MASK) >>
ADC_OFS_OFS_SHIFT)
#define ADC_BRD_OFS_OFS(base) (BME_UBFX32(&ADC_OFS_REG(base),
ADC_OFS_OFS_SHIFT, ADC_OFS_OFS_WIDTH))

/*! @brief Set the OFS field to a new value. */
#define ADC_WR_OFS_OFS(base, value) (ADC_RMW_OFS(base, ADC_OFS_OFS_MASK,
ADC_OFS_OFS(value)))
#define ADC_BWR_OFS_OFS(base, value) (BME_BFI32(&ADC_OFS_REG(base),
(uint32_t)(value) << ADC_OFS_OFS_SHIFT), ADC_OFS_OFS_SHIFT,
ADC_OFS_OFS_WIDTH))
/*@}*/

/********************* ADC_PG - ADC Plus-Side Gain Register *****/
****

 * ADC_PG - ADC Plus-Side Gain Register

***** */

/*!
 * @brief ADC_PG - ADC Plus-Side Gain Register (RW)
 *
 * Reset value: 0x00008200U
 *
 * The Plus-Side Gain Register (PG) contains the gain error correction
for the
 * plus-side input in differential mode or the overall conversion in
single-ended
 * mode. PG, a 16-bit real number in binary format, is the gain
adjustment
 * factor, with the radix point fixed between ADPG15 and ADPG14. This
register must be
 * written by the user with the value described in the calibration
procedure.
 * Otherwise, the gain error specifications may not be met.
 */
/*!
 * @name Constants and macros for entire ADC_PG register
 */
/*@{*/
#define ADC_RD_PG(base) (ADC_PG_REG(base))
#define ADC_WR_PG(base, value) (ADC_PG_REG(base) = (value))
#define ADC_RMW_PG(base, mask, value) (ADC_WR_PG(base, (ADC_RD_PG(base) &
~(mask)) | (value)))
#define ADC_SET_PG(base, value) (BME_OR32(&ADC_PG_REG(base),
(uint32_t)(value)))
#define ADC_CLR_PG(base, value) (BME_AND32(&ADC_PG_REG(base),
(uint32_t)(~(value))))

```

```

#define ADC_TOG_PG(base, value)  (BME_XOR32(&ADC_PG_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual ADC_PG bitfields
 */

/*!!  

 * @name Register ADC_PG, field PG[15:0] (RW)  

 */  

/*@{ */  

/*! @brief Read current value of the ADC_PG_PG field. */  

#define ADC_RD_PG_PG(base) ((ADC_PG_REG(base) & ADC_PG_PG_MASK) >>  

ADC_PG_PG_SHIFT)  

#define ADC_BRD_PG_PG(base) (BME_UBFX32(&ADC_PG_REG(base),  

ADC_PG_PG_SHIFT, ADC_PG_PG_WIDTH))  

  

/*! @brief Set the PG field to a new value. */  

#define ADC_WR_PG_PG(base, value) (ADC_RMW_PG(base, ADC_PG_PG_MASK,  

ADC_PG_PG(value)))  

#define ADC_BWR_PG_PG(base, value) (BME_BFI32(&ADC_PG_REG(base),  

(uint32_t)(value) << ADC_PG_PG_SHIFT), ADC_PG_PG_SHIFT,  

ADC_PG_PG_WIDTH))  

/*@}/
```

\*\*\*\*\*

\* ADC\_MG - ADC Minus-Side Gain Register

\*\*\*\*\*

\*\*\*\*\*

```

/*!!  

 * @brief ADC_MG - ADC Minus-Side Gain Register (RW)  

 *  

 * Reset value: 0x00008200U  

 *  

 * The Minus-Side Gain Register (MG) contains the gain error correction  

for the  

* minus-side input in differential mode. This register is ignored in  

* single-ended mode. MG, a 16-bit real number in binary format, is the  

gain adjustment  

* factor, with the radix point fixed between ADMG15 and ADMG14. This  

register must  

* be written by the user with the value described in the calibration  

procedure.  

* Otherwise, the gain error specifications may not be met.  

*/  

/*!!  

 * @name Constants and macros for entire ADC_MG register  

 */  

/*@{ */  

#define ADC_RD_MG(base) (ADC_MG_REG(base))
```

```

#define ADC_WR_MG(base, value)      (ADC_MG_REG(base) = (value))
#define ADC_RMW_MG(base, mask, value) (ADC_WR_MG(base, (ADC_RD_MG(base) &
~(mask)) | (value)))
#define ADC_SET_MG(base, value)     (BME_OR32(&ADC_MG_REG(base),
(uint32_t)(value)))
#define ADC_CLR_MG(base, value)     (BME_AND32(&ADC_MG_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_MG(base, value)     (BME_XOR32(&ADC_MG_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual ADC_MG bitfields
 */

/*!!
 * @name Register ADC_MG, field MG[15:0] (RW)
 */
/*@{*/
/*! @brief Read current value of the ADC_MG_MG field. */
#define ADC_RD_MG_MG(base) ((ADC_MG_REG(base) & ADC_MG_MG_MASK) >>
ADC_MG_MG_SHIFT)
#define ADC_BRD_MG_MG(base) (BME_UBFX32(&ADC_MG_REG(base),
ADC_MG_MG_SHIFT, ADC_MG_MG_WIDTH))

/*! @brief Set the MG field to a new value. */
#define ADC_WR_MG_MG(base, value) (ADC_RMW_MG(base, ADC_MG_MG_MASK,
ADC_MG_MG(value)))
#define ADC_BWR_MG_MG(base, value) (BME_BFI32(&ADC_MG_REG(base),
(uint32_t)(value) << ADC_MG_MG_SHIFT), ADC_MG_MG_SHIFT,
ADC_MG_MG_WIDTH)
/*@}*/

/********************* ****
 * ADC_CLPD - ADC Plus-Side General Calibration Value Register
 ****
 */

/*!!
 * @brief ADC_CLPD - ADC Plus-Side General Calibration Value Register
 * (RW)
 *
 * Reset value: 0x0000000AU
 *
 * The Plus-Side General Calibration Value Registers (CLPx) contain
 * calibration
 * information that is generated by the calibration function. These
 * registers
 * contain seven calibration values of varying widths: CLP0[5:0],
 * CLP1[6:0],
 * CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. CLPx are
 * automatically set

```

```

 * when the self-calibration sequence is done, that is, CAL is cleared.
If these
 * registers are written by the user after calibration, the linearity
error
 * specifications may not be met.
*/
/*!
 * @name Constants and macros for entire ADC_CLPD register
 */
/*@{*/
#define ADC_RD_CLPD(base)          (ADC_CLPD_REG(base))
#define ADC_WR_CLPD(base, value)   (ADC_CLPD_REG(base) = (value))
#define ADC_RMW_CLPD(base, mask, value) (ADC_WR_CLPD(base,
(ADC_RD_CLPD(base) & ~mask) | (value)))
#define ADC_SET_CLPD(base, value)  (BME_OR32(&ADC_CLPD_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLPD(base, value)   (BME_AND32(&ADC_CLPD_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLPD(base, value)   (BME_XOR32(&ADC_CLPD_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual ADC_CLPD bitfields
*/
/*!
 * @name Register ADC_CLPD, field CLPD[5:0] (RW)
 *
 * Calibration Value
 */
/*@{*/
/*! @brief Read current value of the ADC_CLPD_CLPD field. */
#define ADC_RD_CLPD_CLPD(base) ((ADC_CLPD_REG(base) & ADC_CLPD_CLPD_MASK)
>> ADC_CLPD_CLPD_SHIFT)
#define ADC_BRD_CLPD_CLPD(base) (BME_UBFX32(&ADC_CLPD_REG(base),
ADC_CLPD_CLPD_SHIFT, ADC_CLPD_CLPD_WIDTH))

/*! @brief Set the CLPD field to a new value. */
#define ADC_WR_CLPD_CLPD(base, value) (ADC_RMW_CLPD(base,
ADC_CLPD_CLPD_MASK, ADC_CLPD_CLPD(value)))
#define ADC_BWR_CLPD_CLPD(base, value) (BME_BFI32(&ADC_CLPD_REG(base),
((uint32_t)(value) << ADC_CLPD_CLPD_SHIFT), ADC_CLPD_CLPD_SHIFT,
ADC_CLPD_CLPD_WIDTH))
/*@}*/
*****  

*****  

 * ADC_CLPS - ADC Plus-Side General Calibration Value Register  

*****  

*****/
/*!

```

```

 * @brief ADC_CLPS - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000020U
 *
 * For more information, see CLPD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLPS register
 */
/*@{ */

#define ADC_RD_CLPS(base)          (ADC_CLPS_REG(base))
#define ADC_WR_CLPS(base, value)   (ADC_CLPS_REG(base) = (value))
#define ADC_RMW_CLPS(base, mask, value) (ADC_WR_CLPS(base,
(ADC_RD_CLPS(base) & ~mask) | (value)))
#define ADC_SET_CLPS(base, value)  (BME_OR32(&ADC_CLPS_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLPS(base, value)  (BME_AND32(&ADC_CLPS_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLPS(base, value)  (BME_XOR32(&ADC_CLPS_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual ADC_CLPS bitfields
 */

/*!
 * @name Register ADC_CLPS, field CLPS[5:0] (RW)
 *
 * Calibration Value
 */
/*@{ */

/*! @brief Read current value of the ADC_CLPS_CLPS field. */
#define ADC_RD_CLPS_CLPS(base) ((ADC_CLPS_REG(base) & ADC_CLPS_CLPS_MASK)
>> ADC_CLPS_CLPS_SHIFT)
#define ADC_BRD_CLPS_CLPS(base) (BME_UBFX32(&ADC_CLPS_REG(base),
ADC_CLPS_CLPS_SHIFT, ADC_CLPS_CLPS_WIDTH))

/*! @brief Set the CLPS field to a new value. */
#define ADC_WR_CLPS_CLPS(base, value) (ADC_RMW_CLPS(base,
ADC_CLPS_CLPS_MASK, ADC_CLPS_CLPS(value)))
#define ADC_BWR_CLPS_CLPS(base, value) (BME_BFI32(&ADC_CLPS_REG(base),
((uint32_t)(value) << ADC_CLPS_CLPS_SHIFT), ADC_CLPS_CLPS_SHIFT,
ADC_CLPS_CLPS_WIDTH))
/*@} */

*****  

*****  

 * ADC_CLP4 - ADC Plus-Side General Calibration Value Register  

*****  

*****/

```

```

/*!
 * @brief ADC_CLP4 - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000200U
 *
 * For more information, see CLPD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLP4 register
 */
/*@{ */

#define ADC_RD_CLP4(base)          (ADC_CLP4_REG(base))
#define ADC_WR_CLP4(base, value)   (ADC_CLP4_REG(base) = (value))
#define ADC_RMW_CLP4(base, mask, value) (ADC_WR_CLP4(base,
(ADC_RD_CLP4(base) & ~mask) | (value)))
#define ADC_SET_CLP4(base, value)  (BME_OR32(&ADC_CLP4_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLP4(base, value)  (BME_AND32(&ADC_CLP4_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLP4(base, value)  (BME_XOR32(&ADC_CLP4_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual ADC_CLP4 bitfields
 */
/*!
 * @name Register ADC_CLP4, field CLP4[9:0] (RW)
 *
 * Calibration Value
 */
/*@{ */

/*! @brief Read current value of the ADC_CLP4_CLP4 field. */
#define ADC_RD_CLP4_CLP4(base) ((ADC_CLP4_REG(base) & ADC_CLP4_CLP4_MASK)
>> ADC_CLP4_CLP4_SHIFT)
#define ADC_BRD_CLP4_CLP4(base) (BME_UBFX32(&ADC_CLP4_REG(base),
ADC_CLP4_CLP4_SHIFT, ADC_CLP4_CLP4_WIDTH))

/*! @brief Set the CLP4 field to a new value. */
#define ADC_WR_CLP4_CLP4(base, value) (ADC_RMW_CLP4(base,
ADC_CLP4_CLP4_MASK, ADC_CLP4_CLP4(value)))
#define ADC_BWR_CLP4_CLP4(base, value) (BME_BFI32(&ADC_CLP4_REG(base),
((uint32_t)(value) << ADC_CLP4_CLP4_SHIFT), ADC_CLP4_CLP4_SHIFT,
ADC_CLP4_CLP4_WIDTH))
/*@} */

*****  

* ADC_CLP3 - ADC Plus-Side General Calibration Value Register  

*****  

*****/

```

```

/*!
 * @brief ADC_CLP3 - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000100U
 *
 * For more information, see CLPD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLP3 register
 */
/*@{ */

#define ADC_RD_CLP3(base)          (ADC_CLP3_REG(base))
#define ADC_WR_CLP3(base, value)   (ADC_CLP3_REG(base) = (value))
#define ADC_RMW_CLP3(base, mask, value) (ADC_WR_CLP3(base,
(ADC_RD_CLP3(base) & ~mask) | (value)))
#define ADC_SET_CLP3(base, value)  (BME_OR32(&ADC_CLP3_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLP3(base, value)  (BME_AND32(&ADC_CLP3_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLP3(base, value)  (BME_XOR32(&ADC_CLP3_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual ADC_CLP3 bitfields
 */

/*!
 * @name Register ADC_CLP3, field CLP3[8:0] (RW)
 *
 * Calibration Value
 */
/*@{ */

/*! @brief Read current value of the ADC_CLP3_CLP3 field. */
#define ADC_RD_CLP3_CLP3(base) ((ADC_CLP3_REG(base) & ADC_CLP3_CLP3_MASK)
>> ADC_CLP3_CLP3_SHIFT)
#define ADC_BRD_CLP3_CLP3(base) (BME_UBFX32(&ADC_CLP3_REG(base),
ADC_CLP3_CLP3_SHIFT, ADC_CLP3_CLP3_WIDTH))

/*! @brief Set the CLP3 field to a new value. */
#define ADC_WR_CLP3_CLP3(base, value) (ADC_RMW_CLP3(base,
ADC_CLP3_CLP3_MASK, ADC_CLP3_CLP3(value)))
#define ADC_BWR_CLP3_CLP3(base, value) (BME_BFI32(&ADC_CLP3_REG(base),
((uint32_t)(value) << ADC_CLP3_CLP3_SHIFT), ADC_CLP3_CLP3_SHIFT,
ADC_CLP3_CLP3_WIDTH))
/*@} */

*****  

* ADC_CLP2 - ADC Plus-Side General Calibration Value Register

```

```
*****
**** */

/*!
 * @brief ADC_CLP2 - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000080U
 *
 * For more information, see CLPD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLP2 register
 */
/*@{*/
#define ADC_RD_CLP2(base)          (ADC_CLP2_REG(base))
#define ADC_WR_CLP2(base, value)   (ADC_CLP2_REG(base) = (value))
#define ADC_RMW_CLP2(base, mask, value) (ADC_WR_CLP2(base,
(ADC_RD_CLP2(base) & ~mask) | (value)))
#define ADC_SET_CLP2(base, value)  (BME_OR32(&ADC_CLP2_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLP2(base, value)  (BME_AND32(&ADC_CLP2_REG(base),
(uint32_t)(~value)))
#define ADC_TOG_CLP2(base, value)  (BME_XOR32(&ADC_CLP2_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual ADC_CLP2 bitfields
 */

/*!
 * @name Register ADC_CLP2, field CLP2[7:0] (RW)
 *
 * Calibration Value
 */
/*@{*/
/*! @brief Read current value of the ADC_CLP2_CLP2 field. */
#define ADC_RD_CLP2_CLP2(base) ((ADC_CLP2_REG(base) & ADC_CLP2_CLP2_MASK)
>> ADC_CLP2_CLP2_SHIFT)
#define ADC_BRD_CLP2_CLP2(base) (BME_UBFX32(&ADC_CLP2_REG(base),
ADC_CLP2_CLP2_SHIFT, ADC_CLP2_CLP2_WIDTH))

/*! @brief Set the CLP2 field to a new value. */
#define ADC_WR_CLP2_CLP2(base, value) (ADC_RMW_CLP2(base,
ADC_CLP2_CLP2_MASK, ADC_CLP2_CLP2(value)))
#define ADC_BWR_CLP2_CLP2(base, value) (BME_BFI32(&ADC_CLP2_REG(base),
((uint32_t)(value) << ADC_CLP2_CLP2_SHIFT), ADC_CLP2_CLP2_SHIFT,
ADC_CLP2_CLP2_WIDTH))
/*@}*/

*****
*****
```

```

 * ADC_CLP1 - ADC Plus-Side General Calibration Value Register
 ****
 */

/*!
 * @brief ADC_CLP1 - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000040U
 *
 * For more information, see CLPD register description.
 */
/**!
 * @name Constants and macros for entire ADC_CLP1 register
 */
/*@{*/
#define ADC_RD_CLP1(base)          (ADC_CLP1_REG(base))
#define ADC_WR_CLP1(base, value)   (ADC_CLP1_REG(base) = (value))
#define ADC_RMW_CLP1(base, mask, value) (ADC_WR_CLP1(base,
(ADC_RD_CLP1(base) & ~mask) | (value)))
#define ADC_SET_CLP1(base, value)  (BME_OR32(&ADC_CLP1_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLP1(base, value)   (BME_AND32(&ADC_CLP1_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLP1(base, value)  (BME_XOR32(&ADC_CLP1_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual ADC_CLP1 bitfields
 */
/**!
 * @name Register ADC_CLP1, field CLP1[6:0] (RW)
 *
 * Calibration Value
 */
/*@{*/
/*! @brief Read current value of the ADC_CLP1_CLP1 field. */
#define ADC_RD_CLP1_CLP1(base) ((ADC_CLP1_REG(base) & ADC_CLP1_CLP1_MASK)
>> ADC_CLP1_CLP1_SHIFT)
#define ADC_BRD_CLP1_CLP1(base) (BME_UBFX32(&ADC_CLP1_REG(base),
ADC_CLP1_CLP1_SHIFT, ADC_CLP1_CLP1_WIDTH))

/*! @brief Set the CLP1 field to a new value. */
#define ADC_WR_CLP1_CLP1(base, value) (ADC_RMW_CLP1(base,
ADC_CLP1_CLP1_MASK, ADC_CLP1_CLP1(value)))
#define ADC_BWR_CLP1_CLP1(base, value) (BME_BFI32(&ADC_CLP1_REG(base),
((uint32_t)(value) << ADC_CLP1_CLP1_SHIFT), ADC_CLP1_CLP1_SHIFT,
ADC_CLP1_CLP1_WIDTH))
/*@}*/

```

```

*****
 * ADC_CLP0 - ADC Plus-Side General Calibration Value Register
*****
**** */

/*!
 * @brief ADC_CLP0 - ADC Plus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000020U
 *
 * For more information, see CLPD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLP0 register
 */
/*@{ */
#define ADC_RD_CLP0(base)          (ADC_CLP0_REG(base))
#define ADC_WR_CLP0(base, value)   (ADC_CLP0_REG(base) = (value))
#define ADC_RMW_CLP0(base, mask, value) (ADC_WR_CLP0(base,
(ADC_RD_CLP0(base) & ~mask)) | (value)))
#define ADC_SET_CLP0(base, value)  (BME_OR32(&ADC_CLP0_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLP0(base, value)  (BME_AND32(&ADC_CLP0_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLP0(base, value)  (BME_XOR32(&ADC_CLP0_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual ADC_CLP0 bitfields
 */

/*!
 * @name Register ADC_CLP0, field CLP0[5:0] (RW)
 *
 * Calibration Value
 */
/*@{ */
/*! @brief Read current value of the ADC_CLP0_CLP0 field. */
#define ADC_RD_CLP0_CLP0(base) ((ADC_CLP0_REG(base) & ADC_CLP0_CLP0_MASK)
>> ADC_CLP0_CLP0_SHIFT)
#define ADC_BRD_CLP0_CLP0(base) (BME_UBFX32(&ADC_CLP0_REG(base),
ADC_CLP0_CLP0_SHIFT, ADC_CLP0_CLP0_WIDTH))

/*! @brief Set the CLP0 field to a new value. */
#define ADC_WR_CLP0_CLP0(base, value) (ADC_RMW_CLP0(base,
ADC_CLP0_CLP0_MASK, ADC_CLP0_CLP0(value)))
#define ADC_BWR_CLP0_CLP0(base, value) (BME_BFI32(&ADC_CLP0_REG(base),
((uint32_t)(value) << ADC_CLP0_CLP0_SHIFT), ADC_CLP0_CLP0_SHIFT,
ADC_CLP0_CLP0_WIDTH))
/*@} */

```

```

/*****
 * ADC_CLMD - ADC Minus-Side General Calibration Value Register
 *****/
/*!
 * @brief ADC_CLMD - ADC Minus-Side General Calibration Value Register
 (RW)
 *
 * Reset value: 0x0000000AU
 *
 * The Minus-Side General Calibration Value (CLMx) registers contain
 calibration
 * information that is generated by the calibration function. These
 registers
 * contain seven calibration values of varying widths: CLM0[5:0],
 CLM1[6:0],
 * CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. CLMx are
 automatically
 * set when the self-calibration sequence is done, that is, CAL is
 cleared. If
 * these registers are written by the user after calibration, the
 linearity error
 * specifications may not be met.
 */
/*!
 * @name Constants and macros for entire ADC_CLMD register
 */
/*@{*/
#define ADC_RD_CLMD(base)          (ADC_CLMD_REG(base))
#define ADC_WR_CLMD(base, value)   (ADC_CLMD_REG(base) = (value))
#define ADC_RMW_CLMD(base, mask, value) (ADC_WR_CLMD(base,
(ADC_RD_CLMD(base) & ~mask) | (value)))
#define ADC_SET_CLMD(base, value)  (BME_OR32(&ADC_CLMD_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLMD(base, value)  (BME_AND32(&ADC_CLMD_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLMD(base, value)  (BME_XOR32(&ADC_CLMD_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual ADC_CLMD bitfields
 */
/*!
 * @name Register ADC_CLMD, field CLMD[5:0] (RW)
 *
 * Calibration Value
 */
/*@{*/

```

```

/*! @brief Read current value of the ADC_CLMD_CLMD field. */
#define ADC_RD_CLMD_CLMD(base) ((ADC_CLMD_REG(base)) & ADC_CLMD_CLMD_MASK)
>> ADC_CLMD_CLMD_SHIFT
#define ADC_BRD_CLMD_CLMD(base) (BME_UBFX32(&ADC_CLMD_REG(base),
ADC_CLMD_CLMD_SHIFT, ADC_CLMD_CLMD_WIDTH))

/*! @brief Set the CLMD field to a new value. */
#define ADC_WR_CLMD_CLMD(base, value) (ADC_RMW_CLMD(base,
ADC_CLMD_CLMD_MASK, ADC_CLMD_CLMD(value)))
#define ADC_BWR_CLMD_CLMD(base, value) (BME_BFI32(&ADC_CLMD_REG(base),
((uint32_t)(value) << ADC_CLMD_CLMD_SHIFT), ADC_CLMD_CLMD_SHIFT,
ADC_CLMD_CLMD_WIDTH))
/*@}*/

/********************* ADC_CLMS - ADC Minus-Side General Calibration Value Register *****/
**** */

/*!
* @brief ADC_CLMS - ADC Minus-Side General Calibration Value Register
(RW)
*
* Reset value: 0x00000020U
*
* For more information, see CLMD register description.
*/
/**!
* @name Constants and macros for entire ADC_CLMS register
*/
/*@*/
#define ADC_RD_CLMS(base) (ADC_CLMS_REG(base))
#define ADC_WR_CLMS(base, value) (ADC_CLMS_REG(base) = (value))
#define ADC_RMW_CLMS(base, mask, value) (ADC_WR_CLMS(base,
(ADC_RD_CLMS(base) & ~mask) | value))
#define ADC_SET_CLMS(base, value) (BME_OR32(&ADC_CLMS_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLMS(base, value) (BME_AND32(&ADC_CLMS_REG(base),
(uint32_t)(~value)))
#define ADC_TOG_CLMS(base, value) (BME_XOR32(&ADC_CLMS_REG(base),
(uint32_t)(value)))
/*@*/

/*
* Constants & macros for individual ADC_CLMS bitfields
*/
/*!
* @name Register ADC_CLMS, field CLMS[5:0] (RW)
*
* Calibration Value
*/

```

```

/*@{*/
/*! @brief Read current value of the ADC_CLMS_CLMS field. */
#define ADC_RD_CLMS_CLMS(base) ((ADC_CLMS_REG(base) & ADC_CLMS_CLMS_MASK)
>> ADC_CLMS_CLMS_SHIFT)
#define ADC_BRD_CLMS_CLMS(base) (BME_UBFX32(&ADC_CLMS_REG(base),
ADC_CLMS_CLMS_SHIFT, ADC_CLMS_CLMS_WIDTH))

/*! @brief Set the CLMS field to a new value. */
#define ADC_WR_CLMS_CLMS(base, value) (ADC_RMW_CLMS(base,
ADC_CLMS_CLMS_MASK, ADC_CLMS_CLMS(value)))
#define ADC_BWR_CLMS_CLMS(base, value) (BME_BFI32(&ADC_CLMS_REG(base),
(uint32_t)(value) << ADC_CLMS_CLMS_SHIFT), ADC_CLMS_CLMS_SHIFT,
ADC_CLMS_CLMS_WIDTH))
/*}@*/



/***** ADC_CLM4 - ADC Minus-Side General Calibration Value Register *****/
*****/


/*!
 * @brief ADC_CLM4 - ADC Minus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000200U
 *
 * For more information, see CLMD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLM4 register
 */
/*@{*/
#define ADC_RD_CLM4(base) (ADC_CLM4_REG(base))
#define ADC_WR_CLM4(base, value) (ADC_CLM4_REG(base) = (value))
#define ADC_RMW_CLM4(base, mask, value) (ADC_WR_CLM4(base,
(ADC_RD_CLM4(base) & ~mask) | (value)))
#define ADC_SET_CLM4(base, value) (BME_OR32(&ADC_CLM4_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLM4(base, value) (BME_AND32(&ADC_CLM4_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLM4(base, value) (BME_XOR32(&ADC_CLM4_REG(base),
(uint32_t)(value)))
/*}@*/



/*
 * Constants & macros for individual ADC_CLM4 bitfields
 */
/*!
 * @name Register ADC_CLM4, field CLM4[9:0] (RW)
 *
 * Calibration Value
 */

```

```

/*
/*@{ */
/*! @brief Read current value of the ADC_CLM4_CLM4 field. */
#define ADC_RD_CLM4_CLM4(base) ((ADC_CLM4_REG(base) & ADC_CLM4_CLM4_MASK)
>> ADC_CLM4_CLM4_SHIFT)
#define ADC_BRD_CLM4_CLM4(base) (BME_UBFX32(&ADC_CLM4_REG(base),
ADC_CLM4_CLM4_SHIFT, ADC_CLM4_CLM4_WIDTH))

/*! @brief Set the CLM4 field to a new value. */
#define ADC_WR_CLM4_CLM4(base, value) (ADC_RMW_CLM4(base,
ADC_CLM4_CLM4_MASK, ADC_CLM4_CLM4(value)))
#define ADC_BWR_CLM4_CLM4(base, value) (BME_BFI32(&ADC_CLM4_REG(base),
((uint32_t)(value) << ADC_CLM4_CLM4_SHIFT), ADC_CLM4_CLM4_SHIFT,
ADC_CLM4_CLM4_WIDTH))
/*}@*/ */

*****  

* ADC_CLM3 - ADC Minus-Side General Calibration Value Register  

*****  

*  

/*!  

 * @brief ADC_CLM3 - ADC Minus-Side General Calibration Value Register  

(RW)  

*  

* Reset value: 0x00000100U  

*  

* For more information, see CLMD register description.  

*/  

/*!  

 * @name Constants and macros for entire ADC_CLM3 register  

*/  

/*@{ */
#define ADC_RD_CLM3(base) (ADC_CLM3_REG(base) )
#define ADC_WR_CLM3(base, value) (ADC_CLM3_REG(base) = (value))
#define ADC_RMW_CLM3(base, mask, value) (ADC_WR_CLM3(base,
(ADC_RD_CLM3(base) & ~mask) | (value)))
#define ADC_SET_CLM3(base, value) (BME_OR32(&ADC_CLM3_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLM3(base, value) (BME_AND32(&ADC_CLM3_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLM3(base, value) (BME_XOR32(&ADC_CLM3_REG(base),
(uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual ADC_CLM3 bitfields
*/
/*!  

 * @name Register ADC_CLM3, field CLM3[8:0] (RW)  

*

```

```

 * Calibration Value
 */
/*@{ */
/*! @brief Read current value of the ADC_CLM3_CLM3 field. */
#define ADC_RD_CLM3_CLM3(base) ((ADC_CLM3_REG(base) & ADC_CLM3_CLM3_MASK)
>> ADC_CLM3_CLM3_SHIFT)
#define ADC_BRD_CLM3_CLM3(base) (BME_UBFX32(&ADC_CLM3_REG(base),
ADC_CLM3_CLM3_SHIFT, ADC_CLM3_CLM3_WIDTH))

/*! @brief Set the CLM3 field to a new value. */
#define ADC_WR_CLM3_CLM3(base, value) (ADC_RMW_CLM3(base,
ADC_CLM3_CLM3_MASK, ADC_CLM3_CLM3(value)))
#define ADC_BWR_CLM3_CLM3(base, value) (BME_BFI32(&ADC_CLM3_REG(base),
(uint32_t)(value) << ADC_CLM3_CLM3_SHIFT), ADC_CLM3_CLM3_SHIFT,
ADC_CLM3_CLM3_WIDTH))
/*@} */

/********************* ADC CLM2 - ADC Minus-Side General Calibration Value Register ****
****

 * ADC_CLM2 - ADC Minus-Side General Calibration Value Register

***** ****
****

/*!
 * @brief ADC_CLM2 - ADC Minus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000080U
 *
 * For more information, see CLMD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLM2 register
 */
/*@{ */
#define ADC_RD_CLM2(base) (ADC_CLM2_REG(base))
#define ADC_WR_CLM2(base, value) (ADC_CLM2_REG(base) = (value))
#define ADC_RMW_CLM2(base, mask, value) (ADC_WR_CLM2(base,
(ADC_RD_CLM2(base) & ~mask) | (value)))
#define ADC_SET_CLM2(base, value) (BME_OR32(&ADC_CLM2_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLM2(base, value) (BME_AND32(&ADC_CLM2_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLM2(base, value) (BME_XOR32(&ADC_CLM2_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual ADC_CLM2 bitfields
 */
/*!
 * @name Register ADC_CLM2, field CLM2[7:0] (RW)

```

```

/*
 * Calibration Value
 */
/*@{*/
/*! @brief Read current value of the ADC_CLM2_CLM2 field. */
#define ADC_RD_CLM2_CLM2(base) ((ADC_CLM2_REG(base) & ADC_CLM2_CLM2_MASK)
>> ADC_CLM2_CLM2_SHIFT)
#define ADC_BRD_CLM2_CLM2(base) (BME_UBFX32(&ADC_CLM2_REG(base),
ADC_CLM2_CLM2_SHIFT, ADC_CLM2_CLM2_WIDTH))

/*! @brief Set the CLM2 field to a new value. */
#define ADC_WR_CLM2_CLM2(base, value) (ADC_RMW_CLM2(base,
ADC_CLM2_CLM2_MASK, ADC_CLM2_CLM2(value)))
#define ADC_BWR_CLM2_CLM2(base, value) (BME_BFI32(&ADC_CLM2_REG(base),
(uint32_t)(value) << ADC_CLM2_CLM2_SHIFT), ADC_CLM2_CLM2_SHIFT,
ADC_CLM2_CLM2_WIDTH))
/*}@*/



/********************* ****
***** *
 * ADC_CLM1 - ADC Minus-Side General Calibration Value Register
***** /
*****



/*!
 * @brief ADC_CLM1 - ADC Minus-Side General Calibration Value Register
(RW)
 *
 * Reset value: 0x00000040U
 *
 * For more information, see CLMD register description.
 */
/*!
 * @name Constants and macros for entire ADC_CLM1 register
 */
/*@{*/
#define ADC_RD_CLM1(base) (ADC_CLM1_REG(base))
#define ADC_WR_CLM1(base, value) (ADC_CLM1_REG(base) = (value))
#define ADC_RMW_CLM1(base, mask, value) (ADC_WR_CLM1(base,
(ADC_RD_CLM1(base) & ~mask) | (value)))
#define ADC_SET_CLM1(base, value) (BME_OR32(&ADC_CLM1_REG(base),
(uint32_t)(value)))
#define ADC_CLR_CLM1(base, value) (BME_AND32(&ADC_CLM1_REG(base),
(uint32_t)(~(value))))
#define ADC_TOG_CLM1(base, value) (BME_XOR32(&ADC_CLM1_REG(base),
(uint32_t)(value)))
/*}@*/



/*
 * Constants & macros for individual ADC_CLM1 bitfields
 */
/*!

```

```

* @name Register ADC_CLM1, field CLM1[6:0] (RW)
*
* Calibration Value
*/
/*@{ */
/*! @brief Read current value of the ADC_CLM1_CLM1 field. */
#define ADC_RD_CLM1_CLM1(base) ((ADC_CLM1_REG(base) & ADC_CLM1_CLM1_MASK) >> ADC_CLM1_CLM1_SHIFT)
#define ADC_BRD_CLM1_CLM1(base) (BME_UBFX32(&ADC_CLM1_REG(base), ADC_CLM1_CLM1_SHIFT, ADC_CLM1_CLM1_WIDTH))

/*! @brief Set the CLM1 field to a new value. */
#define ADC_WR_CLM1_CLM1(base, value) (ADC_RMW_CLM1(base, ADC_CLM1_CLM1_MASK, ADC_CLM1_CLM1(value)))
#define ADC_BWR_CLM1_CLM1(base, value) (BME_BFI32(&ADC_CLM1_REG(base), (uint32_t)(value) << ADC_CLM1_CLM1_SHIFT, ADC_CLM1_CLM1_SHIFT, ADC_CLM1_CLM1_WIDTH))
/*}@*/ */

/********************* ADC_CLM0 - ADC Minus-Side General Calibration Value Register *****/
**** */

* ADC_CLM0 - ADC Minus-Side General Calibration Value Register
***** */

/*!
* @brief ADC_CLM0 - ADC Minus-Side General Calibration Value Register (RW)
*
* Reset value: 0x00000020U
*
* For more information, see CLMD register description.
*/
/*!
* @name Constants and macros for entire ADC_CLM0 register
*/
/*@{ */
#define ADC_RD_CLM0(base) (ADC_CLM0_REG(base))
#define ADC_WR_CLM0(base, value) (ADC_CLM0_REG(base) = (value))
#define ADC_RMW_CLM0(base, mask, value) (ADC_WR_CLM0(base, (ADC_RD_CLM0(base) & ~mask) | value))
#define ADC_SET_CLM0(base, value) (BME_OR32(&ADC_CLM0_REG(base), (uint32_t)(value)))
#define ADC_CLR_CLM0(base, value) (BME_AND32(&ADC_CLM0_REG(base), (uint32_t)(~(value))))
#define ADC_TOG_CLM0(base, value) (BME_XOR32(&ADC_CLM0_REG(base), (uint32_t)(value)))
/*}@*/ */

/*
* Constants & macros for individual ADC_CLM0 bitfields
*/

```

```

/*!
 * @name Register ADC_CLM0, field CLM0[5:0] (RW)
 *
 * Calibration Value
 */
/*@{ */

/*! @brief Read current value of the ADC_CLM0_CLM0 field. */
#define ADC_RD_CLM0_CLM0(base) ((ADC_CLM0_REG(base) & ADC_CLM0_CLM0_MASK) \
>> ADC_CLM0_CLM0_SHIFT)
#define ADC_BRD_CLM0_CLM0(base) (BME_UBFX32(&ADC_CLM0_REG(base), \
ADC_CLM0_CLM0_SHIFT, ADC_CLM0_CLM0_WIDTH))

/*! @brief Set the CLM0 field to a new value. */
#define ADC_WR_CLM0_CLM0(base, value) (ADC_RMW_CLM0(base, \
ADC_CLM0_CLM0_MASK, ADC_CLM0_CLM0(value)))
#define ADC_BWR_CLM0_CLM0(base, value) (BME_BFI32(&ADC_CLM0_REG(base), \
(uint32_t)(value) << ADC_CLM0_CLM0_SHIFT), ADC_CLM0_CLM0_SHIFT, \
ADC_CLM0_CLM0_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 CMP
 *
 * High-Speed Comparator (CMP), Voltage Reference (VREF) Digital-to-
Analog Converter (DAC), and Analog Mux (ANMUX)
 *
 * Registers defined in this header file:
 * - CMP_CR0 - CMP Control Register 0
 * - CMP_CR1 - CMP Control Register 1
 * - CMP_FPR - CMP Filter Period Register
 * - CMP_SCR - CMP Status and Control Register
 * - CMP_DACCR - DAC Control Register
 * - CMP_MUXCR - MUX Control Register
 */
#define CMP_INSTANCE_COUNT (1U) /*!< Number of instances of the CMP
module. */
#define CMP0_IDX (0U) /*!< Instance number for CMP0. */

*****  

*****  

 * CMP_CR0 - CMP Control Register 0  

*****  

*****  

/*!  

 * @brief CMP_CR0 - CMP Control Register 0 (RW)  

 * Reset value: 0x00U  

*/
/*!  

 * @name Constants and macros for entire CMP_CR0 register  

*/

```

```

/*@{ */
#define CMP_RD_CR0(base)          (CMP_CR0_REG(base))
#define CMP_WR_CR0(base, value)   (CMP_CR0_REG(base) = (value))
#define CMP_RMW_CR0(base, mask, value) (CMP_WR_CR0(base,
(CMP_RD_CR0(base) & ~mask) | (value)))
#define CMP_SET_CR0(base, value)  (BME_OR8(&CMP_CR0_REG(base),
(uint8_t)(value)))
#define CMP_CLR_CR0(base, value)  (BME_AND8(&CMP_CR0_REG(base),
(uint8_t)(~(value))))
#define CMP_TOG_CR0(base, value)  (BME_XOR8(&CMP_CR0_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual CMP_CR0 bitfields
 */

/*!!
 * @name Register CMP_CR0, field HYSTCTR[1:0] (RW)
 *
 * Defines the programmable hysteresis level. The hysteresis values
associated
 * with each level are device-specific. See the Data Sheet of the device
for the
 * exact values.
 *
 * Values:
 * - 0b00 - Level 0
 * - 0b01 - Level 1
 * - 0b10 - Level 2
 * - 0b11 - Level 3
 */
/*@{ */
/*! @brief Read current value of the CMP_CR0_HYSTCTR field. */
#define CMP_RD_CR0_HYSTCTR(base) ((CMP_CR0_REG(base) &
CMP_CR0_HYSTCTR_MASK) >> CMP_CR0_HYSTCTR_SHIFT)
#define CMP_BRD_CR0_HYSTCTR(base) (BME_UBFX8(&CMP_CR0_REG(base),
CMP_CR0_HYSTCTR_SHIFT, CMP_CR0_HYSTCTR_WIDTH))

/*!! @brief Set the HYSTCTR field to a new value. */
#define CMP_WR_CR0_HYSTCTR(base, value) (CMP_RMW_CR0(base,
CMP_CR0_HYSTCTR_MASK, CMP_CR0_HYSTCTR(value)))
#define CMP_BWR_CR0_HYSTCTR(base, value) (BME_BFI8(&CMP_CR0_REG(base),
((uint8_t)(value) << CMP_CR0_HYSTCTR_SHIFT), CMP_CR0_HYSTCTR_SHIFT,
CMP_CR0_HYSTCTR_WIDTH))
/*@} */

/*!!
 * @name Register CMP_CR0, field FILTER_CNT[6:4] (RW)
 *
 * Represents the number of consecutive samples that must agree prior to
the
 * comparator output filter accepting a new output state. For information
regarding

```

```

* filter programming and latency, see the Functional description.
*
* Values:
* - 0b000 - Filter is disabled. If SE = 1, then COUT is a logic 0. This
is not
*      a legal state, and is not recommended. If SE = 0, COUT = COUTA.
* - 0b001 - One sample must agree. The comparator output is simply
sampled.
* - 0b010 - 2 consecutive samples must agree.
* - 0b011 - 3 consecutive samples must agree.
* - 0b100 - 4 consecutive samples must agree.
* - 0b101 - 5 consecutive samples must agree.
* - 0b110 - 6 consecutive samples must agree.
* - 0b111 - 7 consecutive samples must agree.
*/
/*@{*/
/*! @brief Read current value of the CMP_CR0_FILTER_CNT field. */
#define CMP_RD_CR0_FILTER_CNT(base) ((CMP_CR0_REG(base) &
CMP_CR0_FILTER_CNT_MASK) >> CMP_CR0_FILTER_CNT_SHIFT)
#define CMP_BRD_CR0_FILTER_CNT(base) (BME_UBFX8(&CMP_CR0_REG(base),
CMP_CR0_FILTER_CNT_SHIFT, CMP_CR0_FILTER_CNT_WIDTH))

/*! @brief Set the FILTER_CNT field to a new value. */
#define CMP_WR_CR0_FILTER_CNT(base, value) (CMP_RMW_CR0(base,
CMP_CR0_FILTER_CNT_MASK, CMP_CR0_FILTER_CNT(value)))
#define CMP_BWR_CR0_FILTER_CNT(base, value) (BME_BFI8(&CMP_CR0_REG(base),
((uint8_t)(value) << CMP_CR0_FILTER_CNT_SHIFT), CMP_CR0_FILTER_CNT_SHIFT,
CMP_CR0_FILTER_CNT_WIDTH))
/*@}*/
*****  

* CMP_CR1 - CMP Control Register 1  

*****  

/*!  

* @brief CMP_CR1 - CMP Control Register 1 (RW)  

*  

* Reset value: 0x00U  

*/  

/*!  

* @name Constants and macros for entire CMP_CR1 register  

*/  

/*@{*/
#define CMP_RD_CR1(base) (CMP_CR1_REG(base))
#define CMP_WR_CR1(base, value) (CMP_CR1_REG(base) = (value))
#define CMP_RMW_CR1(base, mask, value) (CMP_WR_CR1(base,
(CMP_RD_CR1(base) & ~mask) | (value)))
#define CMP_SET_CR1(base, value) (BME_OR8(&CMP_CR1_REG(base),
(uint8_t)(value)))
#define CMP_CLR_CR1(base, value) (BME_AND8(&CMP_CR1_REG(base),
(uint8_t)(~(value)))))


```

```

#define CMP_TOG_CR1(base, value)  (BME_XOR8(&CMP_CR1_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual CMP_CR1 bitfields
 */

/*!!  

 * @name Register CMP_CR1, field EN[0] (RW)  

 *  

 * Enables the Analog Comparator module. When the module is not enabled,  

it  

 * remains in the off state, and consumes no power. When the user selects  

the same  

 * input from analog mux to the positive and negative port, the  

comparator is  

 * disabled automatically.  

 *  

 * Values:  

 * - 0b0 - Analog Comparator is disabled.  

 * - 0b1 - Analog Comparator is enabled.  

 */
/*@{*/  

/*!! @brief Read current value of the CMP_CR1_EN field. */  

#define CMP_RD_CR1_EN(base)  ((CMP_CR1_REG(base) & CMP_CR1_EN_MASK) >>  

CMP_CR1_EN_SHIFT)  

#define CMP_BRD_CR1_EN(base)  (BME_UBFX8(&CMP_CR1_REG(base),  

CMP_CR1_EN_SHIFT, CMP_CR1_EN_WIDTH))  

  

/*!! @brief Set the EN field to a new value. */  

#define CMP_WR_CR1_EN(base, value) (CMP_RMW_CR1(base, CMP_CR1_EN_MASK,  

CMP_CR1_EN(value)))  

#define CMP_BWR_CR1_EN(base, value) (BME_BFI8(&CMP_CR1_REG(base),  

((uint8_t)(value) << CMP_CR1_EN_SHIFT), CMP_CR1_EN_SHIFT,  

CMP_CR1_EN_WIDTH))  

/*@}*/  

  

/*!!  

 * @name Register CMP_CR1, field OPE[1] (RW)  

 *  

 * Values:  

 * - 0b0 - CMPO is not available on the associated CMPO output pin. If  

the  

 *       comparator does not own the pin, this field has no effect.  

 * - 0b1 - CMPO is available on the associated CMPO output pin. The  

comparator  

 *       output (CMPO) is driven out on the associated CMPO output pin if  

the  

 *       comparator owns the pin. If the comparator does not own the field,  

this bit has  

 *       no effect.  

 */
/*@{*/

```

```

/*! @brief Read current value of the CMP_CR1_OPE field. */
#define CMP_RD_CR1_OPE(base) ((CMP_CR1_REG(base) & CMP_CR1_OPE_MASK) >>
    CMP_CR1_OPE_SHIFT)
#define CMP_BRD_CR1_OPE(base) (BME_UBFX8(&CMP_CR1_REG(base),
    CMP_CR1_OPE_SHIFT, CMP_CR1_OPE_WIDTH))

/*! @brief Set the OPE field to a new value. */
#define CMP_WR_CR1_OPE(base, value) (CMP_RMW_CR1(base, CMP_CR1_OPE_MASK,
    CMP_CR1_OPE(value)))
#define CMP_BWR_CR1_OPE(base, value) (BME_BFI8(&CMP_CR1_REG(base),
    ((uint8_t)(value) << CMP_CR1_OPE_SHIFT), CMP_CR1_OPE_SHIFT,
    CMP_CR1_OPE_WIDTH))
/*@}*/

/*!
 * @name Register CMP_CR1, field COS[2] (RW)
 *
 * Values:
 * - 0b0 - Set the filtered comparator output (CMPO) to equal COUT.
 * - 0b1 - Set the unfiltered comparator output (CMPO) to equal COUTA.
 */
/*@{*/
/*! @brief Read current value of the CMP_CR1_COS field. */
#define CMP_RD_CR1_COS(base) ((CMP_CR1_REG(base) & CMP_CR1_COS_MASK) >>
    CMP_CR1_COS_SHIFT)
#define CMP_BRD_CR1_COS(base) (BME_UBFX8(&CMP_CR1_REG(base),
    CMP_CR1_COS_SHIFT, CMP_CR1_COS_WIDTH))

/*! @brief Set the COS field to a new value. */
#define CMP_WR_CR1_COS(base, value) (CMP_RMW_CR1(base, CMP_CR1_COS_MASK,
    CMP_CR1_COS(value)))
#define CMP_BWR_CR1_COS(base, value) (BME_BFI8(&CMP_CR1_REG(base),
    ((uint8_t)(value) << CMP_CR1_COS_SHIFT), CMP_CR1_COS_SHIFT,
    CMP_CR1_COS_WIDTH))
/*@}*/

/*!
 * @name Register CMP_CR1, field INV[3] (RW)
 *
 * Allows selection of the polarity of the analog comparator function. It
 * is
 * also driven to the COUT output, on both the device pin and as
SCR[COUT], when
 * OPE=0.
 *
 * Values:
 * - 0b0 - Does not invert the comparator output.
 * - 0b1 - Inverts the comparator output.
 */
/*@{*/
/*! @brief Read current value of the CMP_CR1_INV field. */
#define CMP_RD_CR1_INV(base) ((CMP_CR1_REG(base) & CMP_CR1_INV_MASK) >>
    CMP_CR1_INV_SHIFT)

```

```

#define CMP_BRD_CR1_INV(base)  (BME_UBFX8(&CMP_CR1_REG(base),
CMP_CR1_INV_SHIFT, CMP_CR1_INV_WIDTH))

/*! @brief Set the INV field to a new value. */
#define CMP_WR_CR1_INV(base, value)  (CMP_RMW_CR1(base, CMP_CR1_INV_MASK,
CMP_CR1_INV(value)))
#define CMP_BWR_CR1_INV(base, value)  (BME_BFI8(&CMP_CR1_REG(base),
((uint8_t)(value) << CMP_CR1_INV_SHIFT), CMP_CR1_INV_SHIFT,
CMP_CR1_INV_WIDTH))
/*@}*/

/*!
 * @name Register CMP_CR1, field PMODE[4] (RW)
 *
 * See the electrical specifications table in the device Data Sheet for
details.
 *
 * Values:
 * - 0b0 - Low-Speed (LS) Comparison mode selected. In this mode, CMP has
slower
 *       output propagation delay and lower current consumption.
 * - 0b1 - High-Speed (HS) Comparison mode selected. In this mode, CMP
has
 *       faster output propagation delay and higher current consumption.
 */
/*@{/*/
/*! @brief Read current value of the CMP_CR1_PMODE field. */
#define CMP_RD_CR1_PMODE(base)  ((CMP_CR1_REG(base) & CMP_CR1_PMODE_MASK)
>> CMP_CR1_PMODE_SHIFT)
#define CMP_BRD_CR1_PMODE(base)  (BME_UBFX8(&CMP_CR1_REG(base),
CMP_CR1_PMODE_SHIFT, CMP_CR1_PMODE_WIDTH))

/*! @brief Set the PMODE field to a new value. */
#define CMP_WR_CR1_PMODE(base, value)  (CMP_RMW_CR1(base,
CMP_CR1_PMODE_MASK, CMP_CR1_PMODE(value)))
#define CMP_BWR_CR1_PMODE(base, value)  (BME_BFI8(&CMP_CR1_REG(base),
((uint8_t)(value) << CMP_CR1_PMODE_SHIFT), CMP_CR1_PMODE_SHIFT,
CMP_CR1_PMODE_WIDTH))
/*@}*/

/*!
 * @name Register CMP_CR1, field TRIGM[5] (RW)
 *
 * CMP and DAC are configured to CMP Trigger mode when CMP_CR1[TRIGM] is
set to
 * 1. In addition, the CMP should be enabled. If the DAC is to be used as
a
 * reference to the CMP, it should also be enabled. CMP Trigger mode
depends on an
 * external timer resource to periodically enable the CMP and 6-bit DAC
in order to
 * generate a triggered compare. Upon setting TRIGM, the CMP and DAC are
placed

```

```

 * in a standby state until an external timer resource trigger is
received. See
 * the chip configuration chapter for details about the external timer
resource.
 *
 * Values:
 * - 0b0 - Trigger mode is disabled.
 * - 0b1 - Trigger mode is enabled.
 */
/*@{*/
/*! @brief Read current value of the CMP_CR1_TRIGM field. */
#define CMP_RD_CR1_TRIGM(base) ((CMP_CR1_REG(base) & CMP_CR1_TRIGM_MASK)
>> CMP_CR1_TRIGM_SHIFT)
#define CMP_BRD_CR1_TRIGM(base) (BME_UBFX8(&CMP_CR1_REG(base),
CMP_CR1_TRIGM_SHIFT, CMP_CR1_TRIGM_WIDTH))

/*! @brief Set the TRIGM field to a new value. */
#define CMP_WR_CR1_TRIGM(base, value) (CMP_RMW_CR1(base,
CMP_CR1_TRIGM_MASK, CMP_CR1_TRIGM(value)))
#define CMP_BWR_CR1_TRIGM(base, value) (BME_BFI8(&CMP_CR1_REG(base),
((uint8_t)(value) << CMP_CR1_TRIGM_SHIFT), CMP_CR1_TRIGM_SHIFT,
CMP_CR1_TRIGM_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register CMP_CR1, field WE[6] (RW)
 *
 * At any given time, either SE or WE can be set. It is mandatory request
to not
 * set SE and WE both at a given time.
 *
 * Values:
 * - 0b0 - Windowing mode is not selected.
 * - 0b1 - Windowing mode is selected.
 */
/*@{*/
/*! @brief Read current value of the CMP_CR1_WE field. */
#define CMP_RD_CR1_WE(base) ((CMP_CR1_REG(base) & CMP_CR1_WE_MASK) >>
CMP_CR1_WE_SHIFT)
#define CMP_BRD_CR1_WE(base) (BME_UBFX8(&CMP_CR1_REG(base),
CMP_CR1_WE_SHIFT, CMP_CR1_WE_WIDTH))

/*! @brief Set the WE field to a new value. */
#define CMP_WR_CR1_WE(base, value) (CMP_RMW_CR1(base, CMP_CR1_WE_MASK,
CMP_CR1_WE(value)))
#define CMP_BWR_CR1_WE(base, value) (BME_BFI8(&CMP_CR1_REG(base),
((uint8_t)(value) << CMP_CR1_WE_SHIFT), CMP_CR1_WE_SHIFT,
CMP_CR1_WE_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register CMP_CR1, field SE[7] (RW)
 *
```

```

 * At any given time, either SE or WE can be set. It is mandatory request
to not
 * set SE and WE both at a given time.
 *
 * Values:
 * - 0b0 - Sampling mode is not selected.
 * - 0b1 - Sampling mode is selected.
 */
/*@{/*/
/*! @brief Read current value of the CMP_CR1_SE field. */
#define CMP_RD_CR1_SE(base) ((CMP_CR1_REG(base) & CMP_CR1_SE_MASK) >>
CMP_CR1_SE_SHIFT)
#define CMP_BRD_CR1_SE(base) (BME_UBFX8(&CMP_CR1_REG(base),
CMP_CR1_SE_SHIFT, CMP_CR1_SE_WIDTH))

/*! @brief Set the SE field to a new value. */
#define CMP_WR_CR1_SE(base, value) (CMP_RMW_CR1(base, CMP_CR1_SE_MASK,
CMP_CR1_SE(value)))
#define CMP_BWR_CR1_SE(base, value) (BME_BFI8(&CMP_CR1_REG(base),
((uint8_t)(value) << CMP_CR1_SE_SHIFT), CMP_CR1_SE_SHIFT,
CMP_CR1_SE_WIDTH))
/*@{/*/
/******
 * CMP_FPR - CMP Filter Period Register
*****/
/*!
 * @brief CMP_FPR - CMP Filter Period Register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire CMP_FPR register
 */
/*@{/*/
#define CMP_RD_FPR(base) (CMP_FPR_REG(base))
#define CMP_WR_FPR(base, value) (CMP_FPR_REG(base) = (value))
#define CMP_RMW_FPR(base, mask, value) (CMP_WR_FPR(base,
(CMP_RD_FPR(base) & ~mask) | (value)))
#define CMP_SET_FPR(base, value) (BME_OR8(&CMP_FPR_REG(base),
(uint8_t)(value)))
#define CMP_CLR_FPR(base, value) (BME_AND8(&CMP_FPR_REG(base),
(uint8_t)(~(value))))
#define CMP_TOG_FPR(base, value) (BME_XOR8(&CMP_FPR_REG(base),
(uint8_t)(value)))
/*@{/*/
/******
 * CMP_SCR - CMP Status and Control Register
*****/

```

```
*****
****/


/*!
 * @brief CMP_SCR - CMP Status and Control Register (RW)
 *
 * Reset value: 0x00U
 */
/*!!
 * @name Constants and macros for entire CMP_SCR register
 */
/*@{*/
#define CMP_RD_SCR(base)          (CMP_SCR_REG(base))
#define CMP_WR_SCR(base, value)   (CMP_SCR_REG(base) = (value))
#define CMP_RMW_SCR(base, mask, value) (CMP_WR_SCR(base,
(CMP_RD_SCR(base) & ~mask) | (value)))
#define CMP_SET_SCR(base, value)  (BME_OR8(&CMP_SCR_REG(base),
(uint8_t)(value)))
#define CMP_CLR_SCR(base, value)  (BME_AND8(&CMP_SCR_REG(base),
(uint8_t)(~(value))))
#define CMP_TOG_SCR(base, value)  (BME_XOR8(&CMP_SCR_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual CMP_SCR bitfields
 */

/*!
 * @name Register CMP_SCR, field COUT[0] (RO)
 *
 * Returns the current value of the Analog Comparator output, when read.
The
 * field is reset to 0 and will read as CR1[INV] when the Analog
Comparator module
 * is disabled, that is, when CR1[EN] = 0. Writes to this field are
ignored.
 */
/*@{*/
/*! @brief Read current value of the CMP_SCR_COUT field. */
#define CMP_RD_SCR_COUT(base) ((CMP_SCR_REG(base) & CMP_SCR_COUT_MASK) >>
CMP_SCR_COUT_SHIFT)
#define CMP_BRD_SCR_COUT(base) (BME_UBFX8(&CMP_SCR_REG(base),
CMP_SCR_COUT_SHIFT, CMP_SCR_COUT_WIDTH))
/*@}*/

/*!
 * @name Register CMP_SCR, field CFF[1] (W1C)
 *
 * Detects a falling-edge on COUT, when set, during normal operation. CFF
is
 * cleared by writing 1 to it. During Stop modes, CFF is level sensitive .
 */

```

```

* Values:
* - 0b0 - Falling-edge on COUT has not been detected.
* - 0b1 - Falling-edge on COUT has occurred.
*/
/*@{*/
/*! @brief Read current value of the CMP_SCR_CFF field. */
#define CMP_RD_SCR_CFF(base) ((CMP_SCR_REG(base) & CMP_SCR_CFF_MASK) >>
    CMP_SCR_CFF_SHIFT)
#define CMP_BRD_SCR_CFF(base) (BME_UBFX8(&CMP_SCR_REG(base),
    CMP_SCR_CFF_SHIFT, CMP_SCR_CFF_WIDTH))

/*! @brief Set the CFF field to a new value. */
#define CMP_WR_SCR_CFF(base, value) (CMP_RMW_SCR(base, (CMP_SCR_CFF_MASK
    | CMP_SCR_CFR_MASK), CMP_SCR_CFF(value)))
#define CMP_BWR_SCR_CFF(base, value) (BME_BFI8(&CMP_SCR_REG(base),
    (uint8_t)(value) << CMP_SCR_CFF_SHIFT), CMP_SCR_CFF_SHIFT,
    CMP_SCR_CFF_WIDTH)
/*@}*/

/*!
* @name Register CMP_SCR, field CFR[2] (W1C)
*
* Detects a rising-edge on COUT, when set, during normal operation. CFR is
* cleared by writing 1 to it. During Stop modes, CFR is level sensitive
.
*
* Values:
* - 0b0 - Rising-edge on COUT has not been detected.
* - 0b1 - Rising-edge on COUT has occurred.
*/
/*@{*/
/*! @brief Read current value of the CMP_SCR_CFR field. */
#define CMP_RD_SCR_CFR(base) ((CMP_SCR_REG(base) & CMP_SCR_CFR_MASK) >>
    CMP_SCR_CFR_SHIFT)
#define CMP_BRD_SCR_CFR(base) (BME_UBFX8(&CMP_SCR_REG(base),
    CMP_SCR_CFR_SHIFT, CMP_SCR_CFR_WIDTH))

/*! @brief Set the CFR field to a new value. */
#define CMP_WR_SCR_CFR(base, value) (CMP_RMW_SCR(base, (CMP_SCR_CFR_MASK
    | CMP_SCR_CFF_MASK), CMP_SCR_CFR(value)))
#define CMP_BWR_SCR_CFR(base, value) (BME_BFI8(&CMP_SCR_REG(base),
    (uint8_t)(value) << CMP_SCR_CFR_SHIFT), CMP_SCR_CFR_SHIFT,
    CMP_SCR_CFR_WIDTH)
/*@}*/

/*!
* @name Register CMP_SCR, field IEF[3] (RW)
*
* Enables the CFF interrupt from the CMP. When this field is set, an
* interrupt
* will be asserted when CFF is set.
*
* Values:

```

```

* - 0b0 - Interrupt is disabled.
* - 0b1 - Interrupt is enabled.
*/
/*@{*/
/*! @brief Read current value of the CMP_SCR_IEF field. */
#define CMP_RD_SCR_IEF(base) ((CMP_SCR_REG(base) & CMP_SCR_IEF_MASK) >>
    CMP_SCR_IEF_SHIFT)
#define CMP_BRD_SCR_IEF(base) (BME_UBFX8(&CMP_SCR_REG(base),
    CMP_SCR_IEF_SHIFT, CMP_SCR_IEF_WIDTH))

/*! @brief Set the IEF field to a new value. */
#define CMP_WR_SCR_IEF(base, value) (CMP_RMW_SCR(base, (CMP_SCR_IEF_MASK
    | CMP_SCR_CFF_MASK | CMP_SCR_CFR_MASK), CMP_SCR_IEF(value)))
#define CMP_BWR_SCR_IEF(base, value) (BME_BFI8(&CMP_SCR_REG(base),
    ((uint8_t)(value) << CMP_SCR_IEF_SHIFT), CMP_SCR_IEF_SHIFT,
    CMP_SCR_IEF_WIDTH))
/*}@*/
```

```

/*!
* @name Register CMP_SCR, field IER[4] (RW)
*
* Enables the CFR interrupt from the CMP. When this field is set, an
interrupt
* will be asserted when CFR is set.
*
* Values:
* - 0b0 - Interrupt is disabled.
* - 0b1 - Interrupt is enabled.
*/
/*@{*/
/*! @brief Read current value of the CMP_SCR_IER field. */
#define CMP_RD_SCR_IER(base) ((CMP_SCR_REG(base) & CMP_SCR_IER_MASK) >>
    CMP_SCR_IER_SHIFT)
#define CMP_BRD_SCR_IER(base) (BME_UBFX8(&CMP_SCR_REG(base),
    CMP_SCR_IER_SHIFT, CMP_SCR_IER_WIDTH))

/*! @brief Set the IER field to a new value. */
#define CMP_WR_SCR_IER(base, value) (CMP_RMW_SCR(base, (CMP_SCR_IER_MASK
    | CMP_SCR_CFF_MASK | CMP_SCR_CFR_MASK), CMP_SCR_IER(value)))
#define CMP_BWR_SCR_IER(base, value) (BME_BFI8(&CMP_SCR_REG(base),
    ((uint8_t)(value) << CMP_SCR_IER_SHIFT), CMP_SCR_IER_SHIFT,
    CMP_SCR_IER_WIDTH))
/*}@*/
```

```

/*!
* @name Register CMP_SCR, field DMAEN[6] (RW)
*
* Enables the DMA transfer triggered from the CMP module. When this
field is
* set, a DMA request is asserted when CFR or CFF is set.
*
* Values:
* - 0b0 - DMA is disabled.
* - 0b1 - DMA is enabled.
```

```

/*
/*@{ */
/*! @brief Read current value of the CMP_SCR_DMAEN field. */
#define CMP_RD_SCR_DMAEN(base) ((CMP_SCR_REG(base) & CMP_SCR_DMAEN_MASK)
>> CMP_SCR_DMAEN_SHIFT)
#define CMP_BRD_SCR_DMAEN(base) (BME_UBFX8(&CMP_SCR_REG(base),
CMP_SCR_DMAEN_SHIFT, CMP_SCR_DMAEN_WIDTH))

/*! @brief Set the DMAEN field to a new value. */
#define CMP_WR_SCR_DMAEN(base, value) (CMP_RMW_SCR(base,
(CMP_SCR_DMAEN_MASK | CMP_SCR_CFF_MASK | CMP_SCR_CFR_MASK),
CMP_SCR_DMAEN(value)))
#define CMP_BWR_SCR_DMAEN(base, value) (BME_BFI8(&CMP_SCR_REG(base),
((uint8_t)(value) << CMP_SCR_DMAEN_SHIFT), CMP_SCR_DMAEN_SHIFT,
CMP_SCR_DMAEN_WIDTH))
/*@} */

/********************* */
***** */
* CMP_DACCR - DAC Control Register

***** */
***** */

/*!
* @brief CMP_DACCR - DAC Control Register (RW)
*
* Reset value: 0x00U
*/
/**!
* @name Constants and macros for entire CMP_DACCR register
*/
/*@{ */
#define CMP_RD_DACCR(base) (CMP_DACCR_REG(base))
#define CMP_WR_DACCR(base, value) (CMP_DACCR_REG(base) = (value))
#define CMP_RMW_DACCR(base, mask, value) (CMP_WR_DACCR(base,
(CMP_RD_DACCR(base) & ~mask) | value))
#define CMP_SET_DACCR(base, value) (BME_OR8(&CMP_DACCR_REG(base),
(uint8_t)(value)))
#define CMP_CLR_DACCR(base, value) (BME_AND8(&CMP_DACCR_REG(base),
(uint8_t)(~value)))
#define CMP_TOG_DACCR(base, value) (BME_XOR8(&CMP_DACCR_REG(base),
(uint8_t)(value)))
/*@} */

/*
* Constants & macros for individual CMP_DACCR bitfields
*/
/*!
* @name Register CMP_DACCR, field VOSEL[5:0] (RW)
*
* Selects an output voltage from one of 64 distinct levels. DACO = (V in
/64) *

```

```

    * (VOSEL[5:0] + 1) , so the DACO range is from V in /64 to V in .
    */
/*@{*/
/*! @brief Read current value of the CMP_DACCR_VOSEL field. */
#define CMP_RD_DACCR_VOSEL(base) ((CMP_DACCR_REG(base) &
CMP_DACCR_VOSEL_MASK) >> CMP_DACCR_VOSEL_SHIFT)
#define CMP_BRD_DACCR_VOSEL(base) (BME_UBFX8(&CMP_DACCR_REG(base),
CMP_DACCR_VOSEL_SHIFT, CMP_DACCR_VOSEL_WIDTH))

/*! @brief Set the VOSEL field to a new value. */
#define CMP_WR_DACCR_VOSEL(base, value) (CMP_RMW_DACCR(base,
CMP_DACCR_VOSEL_MASK, CMP_DACCR_VOSEL(value)))
#define CMP_BWR_DACCR_VOSEL(base, value) (BME_BFI8(&CMP_DACCR_REG(base),
(uint8_t)(value) << CMP_DACCR_VOSEL_SHIFT), CMP_DACCR_VOSEL_SHIFT,
CMP_DACCR_VOSEL_WIDTH))
/*}@*/
```

```

/*!
 * @name Register CMP_DACCR, field VRSEL[6] (RW)
 *
 * Values:
 * - 0b0 - V is selected as resistor ladder network supply reference V.
in1 in
 * - 0b1 - V is selected as resistor ladder network supply reference V.
in2 in
 */
/*@{*/
/*! @brief Read current value of the CMP_DACCR_VRSEL field. */
#define CMP_RD_DACCR_VRSEL(base) ((CMP_DACCR_REG(base) &
CMP_DACCR_VRSEL_MASK) >> CMP_DACCR_VRSEL_SHIFT)
#define CMP_BRD_DACCR_VRSEL(base) (BME_UBFX8(&CMP_DACCR_REG(base),
CMP_DACCR_VRSEL_SHIFT, CMP_DACCR_VRSEL_WIDTH))

/*! @brief Set the VRSEL field to a new value. */
#define CMP_WR_DACCR_VRSEL(base, value) (CMP_RMW_DACCR(base,
CMP_DACCR_VRSEL_MASK, CMP_DACCR_VRSEL(value)))
#define CMP_BWR_DACCR_VRSEL(base, value) (BME_BFI8(&CMP_DACCR_REG(base),
(uint8_t)(value) << CMP_DACCR_VRSEL_SHIFT), CMP_DACCR_VRSEL_SHIFT,
CMP_DACCR_VRSEL_WIDTH))
/*}@*/
```

```

/*!
 * @name Register CMP_DACCR, field DACEN[7] (RW)
 *
 * Enables the DAC. When the DAC is disabled, it is powered down to
conserve
 * power.
 *
 * Values:
 * - 0b0 - DAC is disabled.
 * - 0b1 - DAC is enabled.
 */
/*@{*/
/*! @brief Read current value of the CMP_DACCR_DACEN field. */

```

```

#define CMP_RD_DACCR_DACEN(base) ((CMP_DACCR_REG(base) &
CMP_DACCR_DACEN_MASK) >> CMP_DACCR_DACEN_SHIFT)
#define CMP_BRD_DACCR_DACEN(base) (BME_UBFX8(&CMP_DACCR_REG(base),
CMP_DACCR_DACEN_SHIFT, CMP_DACCR_DACEN_WIDTH))

/*! @brief Set the DACEN field to a new value. */
#define CMP_WR_DACCR_DACEN(base, value) (CMP_RMW_DACCR(base,
CMP_DACCR_DACEN_MASK, CMP_DACCR_DACEN(value)))
#define CMP_BWR_DACCR_DACEN(base, value) (BME_BFI8(&CMP_DACCR_REG(base),
((uint8_t)(value) << CMP_DACCR_DACEN_SHIFT), CMP_DACCR_DACEN_SHIFT,
CMP_DACCR_DACEN_WIDTH))
/*@}*/

/*********************  

*****  

 * CMP_MUXCR - MUX Control Register  

*****/  

  

/*!  

 * @brief CMP_MUXCR - MUX Control Register (RW)  

 *  

 * Reset value: 0x00U  

 */  

/*!  

 * @name Constants and macros for entire CMP_MUXCR register  

 */  

/*@*/  

#define CMP_RD_MUXCR(base) (CMP_MUXCR_REG(base))  

#define CMP_WR_MUXCR(base, value) (CMP_MUXCR_REG(base) = (value))  

#define CMP_RMW_MUXCR(base, mask, value) (CMP_WR_MUXCR(base,  

(CMP_RD_MUXCR(base) & ~mask) | (value)))  

#define CMP_SET_MUXCR(base, value) (BME_OR8(&CMP_MUXCR_REG(base),  

(uint8_t)(value)))  

#define CMP_CLR_MUXCR(base, value) (BME_AND8(&CMP_MUXCR_REG(base),  

(uint8_t)(~(value))))  

#define CMP_TOG_MUXCR(base, value) (BME_XOR8(&CMP_MUXCR_REG(base),  

(uint8_t)(value)))  

/*@*/  

  

/*  

 * Constants & macros for individual CMP_MUXCR bitfields  

 */  

  

/*!  

 * @name Register CMP_MUXCR, field MSEL[2:0] (RW)  

 *  

 * Determines which input is selected for the minus input of the  

comparator. For  

 * INx inputs, see CMP, DAC, and ANMUX block diagrams. When an  

inappropriate  

 * operation selects the same input for both muxes, the comparator  

automatically

```

```

* shuts down to prevent itself from becoming a noise generator.
*
* Values:
* - 0b000 - IN0
* - 0b001 - IN1
* - 0b010 - IN2
* - 0b011 - IN3
* - 0b100 - IN4
* - 0b101 - IN5
* - 0b110 - IN6
* - 0b111 - IN7
*/
/*@{*/
/*! @brief Read current value of the CMP_MUXCR_MSEL field. */
#define CMP_RD_MUXCR_MSEL(base) ((CMP_MUXCR_REG(base) &
CMP_MUXCR_MSEL_MASK) >> CMP_MUXCR_MSEL_SHIFT)
#define CMP_BRD_MUXCR_MSEL(base) (BME_UBFX8(&CMP_MUXCR_REG(base),
CMP_MUXCR_MSEL_SHIFT, CMP_MUXCR_MSEL_WIDTH))

/*! @brief Set the MSEL field to a new value. */
#define CMP_WR_MUXCR_MSEL(base, value) (CMP_RMW_MUXCR(base,
CMP_MUXCR_MSEL_MASK, CMP_MUXCR_MSEL(value)))
#define CMP_BWR_MUXCR_MSEL(base, value) (BME_BFI8(&CMP_MUXCR_REG(base),
((uint8_t)(value) << CMP_MUXCR_MSEL_SHIFT), CMP_MUXCR_MSEL_SHIFT,
CMP_MUXCR_MSEL_WIDTH))
/*}@*/ */

/*!
* @name Register CMP_MUXCR, field PSEL[5:3] (RW)
*
* Determines which input is selected for the plus input of the
comparator. For
* INx inputs, see CMP, DAC, and ANMUX block diagrams. When an
inappropriate
* operation selects the same input for both muxes, the comparator
automatically
* shuts down to prevent itself from becoming a noise generator.
*
* Values:
* - 0b000 - IN0
* - 0b001 - IN1
* - 0b010 - IN2
* - 0b011 - IN3
* - 0b100 - IN4
* - 0b101 - IN5
* - 0b110 - IN6
* - 0b111 - IN7
*/
/*@{*/
/*! @brief Read current value of the CMP_MUXCR_PSEL field. */
#define CMP_RD_MUXCR_PSEL(base) ((CMP_MUXCR_REG(base) &
CMP_MUXCR_PSEL_MASK) >> CMP_MUXCR_PSEL_SHIFT)
#define CMP_BRD_MUXCR_PSEL(base) (BME_UBFX8(&CMP_MUXCR_REG(base),
CMP_MUXCR_PSEL_SHIFT, CMP_MUXCR_PSEL_WIDTH))

```

```

/*! @brief Set the PSEL field to a new value. */
#define CMP_WR_MUXCR_PSEL(base, value) (CMP_RMW_MUXCR(base,
CMP_MUXCR_PSEL_MASK, CMP_MUXCR_PSEL(value)))
#define CMP_BWR_MUXCR_PSEL(base, value) (BME_BFI8(&CMP_MUXCR_REG(base),
((uint8_t)(value) << CMP_MUXCR_PSEL_SHIFT), CMP_MUXCR_PSEL_SHIFT,
CMP_MUXCR_PSEL_WIDTH))
/*@}*/

/*! 
 * @name Register CMP_MUXCR, field PSTM[7] (RW)
 *
 * This bit is used to enable to MUX pass through mode. Pass through mode
is
 * always available but for some devices this feature must be always
disabled due to
 * the lack of package pins.
 *
 * Values:
 * - 0b0 - Pass Through Mode is disabled.
 * - 0b1 - Pass Through Mode is enabled.
 */
/*@{*/
/*! @brief Read current value of the CMP_MUXCR_PSTM field. */
#define CMP_RD_MUXCR_PSTM(base) ((CMP_MUXCR_REG(base) &
CMP_MUXCR_PSTM_MASK) >> CMP_MUXCR_PSTM_SHIFT)
#define CMP_BRD_MUXCR_PSTM(base) (BME_UBFX8(&CMP_MUXCR_REG(base),
CMP_MUXCR_PSTM_SHIFT, CMP_MUXCR_PSTM_WIDTH))

/*! @brief Set the PSTM field to a new value. */
#define CMP_WR_MUXCR_PSTM(base, value) (CMP_RMW_MUXCR(base,
CMP_MUXCR_PSTM_MASK, CMP_MUXCR_PSTM(value)))
#define CMP_BWR_MUXCR_PSTM(base, value) (BME_BFI8(&CMP_MUXCR_REG(base),
((uint8_t)(value) << CMP_MUXCR_PSTM_SHIFT), CMP_MUXCR_PSTM_SHIFT,
CMP_MUXCR_PSTM_WIDTH))
/*@}*/

/*
 * MKL25Z4 DAC
 *
 * 12-Bit Digital-to-Analog Converter
 *
 * Registers defined in this header file:
 * - DAC_DATL - DAC Data Low Register
 * - DAC_DATH - DAC Data High Register
 * - DAC_SR - DAC Status Register
 * - DAC_C0 - DAC Control Register
 * - DAC_C1 - DAC Control Register 1
 * - DAC_C2 - DAC Control Register 2
 */
#define DAC_INSTANCE_COUNT (1U) /*!< Number of instances of the DAC
module. */
#define DAC0_IDX (0U) /*!< Instance number for DAC0. */

```

```

/*****+
*****
 * DAC_DATL - DAC Data Low Register
*****
*/
/*!
 * @brief DAC_DATL - DAC Data Low Register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire DAC_DATL register
 */
/*@{*/
#define DAC_RD_DATL(base, index) (DAC_DATL_REG(base, index))
#define DAC_WR_DATL(base, index, value) (DAC_DATL_REG(base, index) = (value))
#define DAC_RMW_DATL(base, index, mask, value) (DAC_WR_DATL(base, index, (DAC_RD_DATL(base, index) & ~mask) | value))
#define DAC_SET_DATL(base, index, value) (BME_OR8(&DAC_DATL_REG(base, index), (uint8_t)(value)))
#define DAC_CLR_DATL(base, index, value) (BME_AND8(&DAC_DATL_REG(base, index), (uint8_t)(~value)))
#define DAC_TOG_DATL(base, index, value) (BME_XOR8(&DAC_DATL_REG(base, index), (uint8_t)(value)))
/*@}*/
/*****+
*****
 * DAC_DATH - DAC Data High Register
*****
*/
/*!
 * @brief DAC_DATH - DAC Data High Register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire DAC_DATH register
 */
/*@{*/
#define DAC_RD_DATH(base, index) (DAC_DATH_REG(base, index))
#define DAC_WR_DATH(base, index, value) (DAC_DATH_REG(base, index) = (value))
#define DAC_RMW_DATH(base, index, mask, value) (DAC_WR_DATH(base, index, (DAC_RD_DATH(base, index) & ~mask) | value))
#define DAC_SET_DATH(base, index, value) (BME_OR8(&DAC_DATH_REG(base, index), (uint8_t)(value)))

```

```

#define DAC_CLR_DATH(base, index, value) (BME_AND8(&DAC_DATH_REG(base, index), (uint8_t)(~(value))))
#define DAC_TOG_DATH(base, index, value) (BME_XOR8(&DAC_DATH_REG(base, index), (uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual DAC_DATH bitfields
 */

/*!
 * @name Register DAC_DATH, field DATA1[3:0] (RW)
 *
 * When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage
 * based on the following formula. V out = V in * (1 + DACDAT0[11:0])/4096 When the
 * DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer.
 */
/*@{*/
/*! @brief Read current value of the DAC_DATH_DATA1 field. */
#define DAC_RD_DATH_DATA1(base, index) ((DAC_DATH_REG(base, index) & DAC_DATH_DATA1_MASK) >> DAC_DATH_DATA1_SHIFT)
#define DAC_BRD_DATH_DATA1(base, index) (BME_UBFX8(&DAC_DATH_REG(base, index), DAC_DATH_DATA1_SHIFT, DAC_DATH_DATA1_WIDTH))

/*! @brief Set the DATA1 field to a new value. */
#define DAC_WR_DATH_DATA1(base, index, value) (DAC_RMW_DATH(base, index, DAC_DATH_DATA1_MASK, DAC_DATH_DATA1(value)))
#define DAC_BWR_DATH_DATA1(base, index, value)
(BME_BFI8(&DAC_DATH_REG(base, index), ((uint8_t)(value) << DAC_DATH_DATA1_SHIFT), DAC_DATH_DATA1_SHIFT, DAC_DATH_DATA1_WIDTH))
/*@}*/

*****  

*****  

 * DAC_SR - DAC Status Register  

*****  

*****  

/*!  

 * @brief DAC_SR - DAC Status Register (RW)  

 *  

 * Reset value: 0x02U  

 *  

 * If DMA is enabled, the flags can be cleared automatically by DMA when the DMA  

 * request is done. Writing 0 to a field clears it whereas writing 1 has no  

 * effect. After reset, DACBFRPTF is set and can be cleared by software, if needed.  

 * The flags are set only when the data buffer status is changed. Do not use

```

```

 * 32/16-bit accesses to this register.
 */
/*!
 * @name Constants and macros for entire DAC_SR register
 */
/*@{*/
#define DAC_RD_SR(base)          (DAC_SR_REG(base))
#define DAC_WR_SR(base, value)    (DAC_SR_REG(base) = (value))
#define DAC_RMW_SR(base, mask, value) (DAC_WR_SR(base, (DAC_RD_SR(base) &
~(mask)) | (value)))
#define DAC_SET_SR(base, value)   (BME_OR8(&DAC_SR_REG(base),
(uint8_t)(value)))
#define DAC_CLR_SR(base, value)   (BME_AND8(&DAC_SR_REG(base),
(uint8_t)(~(value))))
#define DAC_TOG_SR(base, value)   (BME_XOR8(&DAC_SR_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual DAC_SR bitfields
 */

/*!
 * @name Register DAC_SR, field DACBFRPBF[0] (RW)
 *
 * Values:
 * - 0b0 - The DAC buffer read pointer is not equal to C2[DACBFUP].
 * - 0b1 - The DAC buffer read pointer is equal to C2[DACBFUP].
 */
/*@{*/
/*! @brief Read current value of the DAC_SR_DACBFRPBF field. */
#define DAC_RD_SR_DACBFRPBF(base) ((DAC_SR_REG(base) &
DAC_SR_DACBFRPBF_MASK) >> DAC_SR_DACBFRPBF_SHIFT)
#define DAC_BRD_SR_DACBFRPBF(base) (BME_UBFX8(&DAC_SR_REG(base),
DAC_SR_DACBFRPBF_SHIFT, DAC_SR_DACBFRPBF_WIDTH))

/*! @brief Set the DACBFRPBF field to a new value. */
#define DAC_WR_SR_DACBFRPBF(base, value) (DAC_RMW_SR(base,
DAC_SR_DACBFRPBF_MASK, DAC_SR_DACBFRPBF(value)))
#define DAC_BWR_SR_DACBFRPBF(base, value) (BME_BFI8(&DAC_SR_REG(base),
((uint8_t)(value) << DAC_SR_DACBFRPBF_SHIFT), DAC_SR_DACBFRPBF_SHIFT,
DAC_SR_DACBFRPBF_WIDTH))
/*}@*/
/*
 * @name Register DAC_SR, field DACBFRPTF[1] (RW)
 *
 * Values:
 * - 0b0 - The DAC buffer read pointer is not zero.
 * - 0b1 - The DAC buffer read pointer is zero.
 */
/*@{*/
/*! @brief Read current value of the DAC_SR_DACBFRPTF field. */

```

```

#define DAC_RD_SR_DACBFRPTF(base) ((DAC_SR_REG(base) &
DAC_SR_DACBFRPTF_MASK) >> DAC_SR_DACBFRPTF_SHIFT)
#define DAC_BRD_SR_DACBFRPTF(base) (BME_UBFX8(&DAC_SR_REG(base),
DAC_SR_DACBFRPTF_SHIFT, DAC_SR_DACBFRPTF_WIDTH))

/*! @brief Set the DACBFRPTF field to a new value. */
#define DAC_WR_SR_DACBFRPTF(base, value) (DAC_RMW_SR(base,
DAC_SR_DACBFRPTF_MASK, DAC_SR_DACBFRPTF(value)))
#define DAC_BWR_SR_DACBFRPTF(base, value) (BME_BFI8(&DAC_SR_REG(base),
((uint8_t)(value) << DAC_SR_DACBFRPTF_SHIFT), DAC_SR_DACBFRPTF_SHIFT,
DAC_SR_DACBFRPTF_WIDTH))
/*@}*/

/******************
 * DAC_C0 - DAC Control Register
******************/

/*!!
 * @brief DAC_C0 - DAC Control Register (RW)
 *
 * Reset value: 0x00U
 *
 * Do not use 32- or 16-bit accesses to this register.
 */
/*!
 * @name Constants and macros for entire DAC_C0 register
 */
/*@{*/
#define DAC_RD_C0(base) (DAC_C0_REG(base))
#define DAC_WR_C0(base, value) (DAC_C0_REG(base) = (value))
#define DAC_RMW_C0(base, mask, value) (DAC_WR_C0(base, (DAC_RD_C0(base) &
~(mask)) | (value)))
#define DAC_SET_C0(base, value) (BME_OR8(&DAC_C0_REG(base),
(uint8_t)(value)))
#define DAC_CLR_C0(base, value) (BME_AND8(&DAC_C0_REG(base),
(uint8_t)(~(value))))
#define DAC_TOG_C0(base, value) (BME_XOR8(&DAC_C0_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual DAC_C0 bitfields
 */

/*!!
 * @name Register DAC_C0, field DACBBIEN[0] (RW)
 *
 * Values:
 * - 0b0 - The DAC buffer read pointer bottom flag interrupt is disabled.
 * - 0b1 - The DAC buffer read pointer bottom flag interrupt is enabled.
 */

```

```

/*@{*/
/*! @brief Read current value of the DAC_C0_DACBBIEN field. */
#define DAC_RD_C0_DACBBIEN(base) ((DAC_C0_REG(base) &
DAC_C0_DACBBIEN_MASK) >> DAC_C0_DACBBIEN_SHIFT)
#define DAC_BRD_C0_DACBBIEN(base) (BME_UBFX8(&DAC_C0_REG(base),
DAC_C0_DACBBIEN_SHIFT, DAC_C0_DACBBIEN_WIDTH))

/*! @brief Set the DACBBIEN field to a new value. */
#define DAC_WR_C0_DACBBIEN(base, value) (DAC_RMW_C0(base,
DAC_C0_DACBBIEN_MASK, DAC_C0_DACBBIEN(value)))
#define DAC_BWR_C0_DACBBIEN(base, value) (BME_BFI8(&DAC_C0_REG(base),
((uint8_t)(value) << DAC_C0_DACBBIEN_SHIFT), DAC_C0_DACBBIEN_SHIFT,
DAC_C0_DACBBIEN_WIDTH))
/*}@*/
```

```

/*!
 * @name Register DAC_C0, field DACBTIEN[1] (RW)
 *
 * Values:
 * - 0b0 - The DAC buffer read pointer top flag interrupt is disabled.
 * - 0b1 - The DAC buffer read pointer top flag interrupt is enabled.
 */
/*@{*/
/*! @brief Read current value of the DAC_C0_DACBTIEN field. */
#define DAC_RD_C0_DACBTIEN(base) ((DAC_C0_REG(base) &
DAC_C0_DACBTIEN_MASK) >> DAC_C0_DACBTIEN_SHIFT)
#define DAC_BRD_C0_DACBTIEN(base) (BME_UBFX8(&DAC_C0_REG(base),
DAC_C0_DACBTIEN_SHIFT, DAC_C0_DACBTIEN_WIDTH))

/*! @brief Set the DACBTIEN field to a new value. */
#define DAC_WR_C0_DACBTIEN(base, value) (DAC_RMW_C0(base,
DAC_C0_DACBTIEN_MASK, DAC_C0_DACBTIEN(value)))
#define DAC_BWR_C0_DACBTIEN(base, value) (BME_BFI8(&DAC_C0_REG(base),
((uint8_t)(value) << DAC_C0_DACBTIEN_SHIFT), DAC_C0_DACBTIEN_SHIFT,
DAC_C0_DACBTIEN_WIDTH))
/*}@*/
```

```

/*!
 * @name Register DAC_C0, field LPEN[3] (RW)
 *
 * See the 12-bit DAC electrical characteristics of the device data sheet
for
 * details on the impact of the modes below.
 *
 * Values:
 * - 0b0 - High-Power mode
 * - 0b1 - Low-Power mode
 */
/*@{*/
/*! @brief Read current value of the DAC_C0_LPEN field. */
#define DAC_RD_C0_LPEN(base) ((DAC_C0_REG(base) & DAC_C0_LPEN_MASK) >>
DAC_C0_LPEN_SHIFT)
#define DAC_BRD_C0_LPEN(base) (BME_UBFX8(&DAC_C0_REG(base),
DAC_C0_LPEN_SHIFT, DAC_C0_LPEN_WIDTH))
```

```

/*! @brief Set the LPEN field to a new value. */
#define DAC_WR_C0_LPEN(base, value) (DAC_RMW_C0(base, DAC_C0_LPEN_MASK,
DAC_C0_LPEN(value)))
#define DAC_BWR_C0_LPEN(base, value) (BME_BFI8(&DAC_C0_REG(base),
((uint8_t)(value) << DAC_C0_LPEN_SHIFT), DAC_C0_LPEN_SHIFT,
DAC_C0_LPEN_WIDTH))
/*@}*/

/*!!
 * @name Register DAC_C0, field DACSWTRG[4] (WORZ)
 *
 * Active high. This is a write-only field, which always reads 0. If DAC
 * software trigger is selected and buffer is enabled, writing 1 to this
field will
 * advance the buffer read pointer once.
 *
 * Values:
 * - 0b0 - The DAC soft trigger is not valid.
 * - 0b1 - The DAC soft trigger is valid.
 */
/*@*/
/*! @brief Set the DACSWTRG field to a new value. */
#define DAC_WR_C0_DACSWTRG(base, value) (DAC_RMW_C0(base,
DAC_C0_DACSWTRG_MASK, DAC_C0_DACSWTRG(value)))
#define DAC_BWR_C0_DACSWTRG(base, value) (BME_BFI8(&DAC_C0_REG(base),
((uint8_t)(value) << DAC_C0_DACSWTRG_SHIFT), DAC_C0_DACSWTRG_SHIFT,
DAC_C0_DACSWTRG_WIDTH))
/*@*/

/*!!
 * @name Register DAC_C0, field DACTRGSEL[5] (RW)
 *
 * Values:
 * - 0b0 - The DAC hardware trigger is selected.
 * - 0b1 - The DAC software trigger is selected.
 */
/*@*/
/*! @brief Read current value of the DAC_C0_DACTRGSEL field. */
#define DAC_RD_C0_DACTRGSEL(base) ((DAC_C0_REG(base) &
DAC_C0_DACTRGSEL_MASK) >> DAC_C0_DACTRGSEL_SHIFT)
#define DAC_BRD_C0_DACTRGSEL(base) (BME_UBFX8(&DAC_C0_REG(base),
DAC_C0_DACTRGSEL_SHIFT, DAC_C0_DACTRGSEL_WIDTH))

/*!! @brief Set the DACTRGSEL field to a new value. */
#define DAC_WR_C0_DACTRGSEL(base, value) (DAC_RMW_C0(base,
DAC_C0_DACTRGSEL_MASK, DAC_C0_DACTRGSEL(value)))
#define DAC_BWR_C0_DACTRGSEL(base, value) (BME_BFI8(&DAC_C0_REG(base),
((uint8_t)(value) << DAC_C0_DACTRGSEL_SHIFT), DAC_C0_DACTRGSEL_SHIFT,
DAC_C0_DACTRGSEL_WIDTH))
/*@*/

/*!!
 * @name Register DAC_C0, field DACRFS[6] (RW)

```

```

/*
 * Values:
 * - 0b0 - The DAC selects DACREF_1 as the reference voltage.
 * - 0b1 - The DAC selects DACREF_2 as the reference voltage.
 */
/*@{*/
/*! @brief Read current value of the DAC_C0_DACRFS field. */
#define DAC_RD_C0_DACRFS(base) ((DAC_C0_REG(base) & DAC_C0_DACRFS_MASK) >> DAC_C0_DACRFS_SHIFT)
#define DAC_BRD_C0_DACRFS(base) (BME_UBXF8(&DAC_C0_REG(base), DAC_C0_DACRFS_SHIFT, DAC_C0_DACRFS_WIDTH))

/*! @brief Set the DACRFS field to a new value. */
#define DAC_WR_C0_DACRFS(base, value) (DAC_RMW_C0(base, DAC_C0_DACRFS_MASK, DAC_C0_DACRFS(value)))
#define DAC_BWR_C0_DACRFS(base, value) (BME_BFI8(&DAC_C0_REG(base), (uint8_t)(value) << DAC_C0_DACRFS_SHIFT), DAC_C0_DACRFS_SHIFT, DAC_C0_DACRFS_WIDTH)
/*}@*/ */

/*!
 * @name Register DAC_C0, field DACEN[7] (RW)
 *
 * Starts the Programmable Reference Generator operation.
 *
 * Values:
 * - 0b0 - The DAC system is disabled.
 * - 0b1 - The DAC system is enabled.
 */
/*@{*/
/*! @brief Read current value of the DAC_C0_DACEN field. */
#define DAC_RD_C0_DACEN(base) ((DAC_C0_REG(base) & DAC_C0_DACEN_MASK) >> DAC_C0_DACEN_SHIFT)
#define DAC_BRD_C0_DACEN(base) (BME_UBXF8(&DAC_C0_REG(base), DAC_C0_DACEN_SHIFT, DAC_C0_DACEN_WIDTH))

/*! @brief Set the DACEN field to a new value. */
#define DAC_WR_C0_DACEN(base, value) (DAC_RMW_C0(base, DAC_C0_DACEN_MASK, DAC_C0_DACEN(value)))
#define DAC_BWR_C0_DACEN(base, value) (BME_BFI8(&DAC_C0_REG(base), (uint8_t)(value) << DAC_C0_DACEN_SHIFT), DAC_C0_DACEN_SHIFT, DAC_C0_DACEN_WIDTH)
/*}@*/ */

/********************* DAC_C1 - DAC Control Register 1 ********************
*****
 * DAC_C1 - DAC Control Register 1
*****
 *****/
/*!
 * @brief DAC_C1 - DAC Control Register 1 (RW)
 *

```

```

 * Reset value: 0x00U
 *
 * Do not use 32- or 16-bit accesses to this register.
 */
/*!
 * @name Constants and macros for entire DAC_C1 register
 */
/*@{ */
#define DAC_RD_C1(base)          (DAC_C1_REG(base))
#define DAC_WR_C1(base, value)    (DAC_C1_REG(base) = (value))
#define DAC_RMW_C1(base, mask, value) (DAC_WR_C1(base, (DAC_RD_C1(base) &
~(mask)) | (value)))
#define DAC_SET_C1(base, value)   (BME_OR8(&DAC_C1_REG(base),
(uint8_t)(value)))
#define DAC_CLR_C1(base, value)   (BME_AND8(&DAC_C1_REG(base),
(uint8_t)(~(value))))
#define DAC_TOG_C1(base, value)   (BME_XOR8(&DAC_C1_REG(base),
(uint8_t)(value)))
/*@}*/ */

/*
 * Constants & macros for individual DAC_C1 bitfields
 */

/*!
 * @name Register DAC_C1, field DACBFEN[0] (RW)
 *
 * Values:
 * - 0b0 - Buffer read pointer is disabled. The converted data is always
the
 *         first word of the buffer.
 * - 0b1 - Buffer read pointer is enabled. The converted data is the word
that
 *         the read pointer points to. It means converted data can be from
any word of
 *         the buffer.
 */
/*@{ */
/*! @brief Read current value of the DAC_C1_DACBFEN field. */
#define DAC_RD_C1_DACBFEN(base) ((DAC_C1_REG(base) & DAC_C1_DACBFEN_MASK)
>> DAC_C1_DACBFEN_SHIFT)
#define DAC_BRD_C1_DACBFEN(base) (BME_UBFX8(&DAC_C1_REG(base),
DAC_C1_DACBFEN_SHIFT, DAC_C1_DACBFEN_WIDTH))

/*! @brief Set the DACBFEN field to a new value. */
#define DAC_WR_C1_DACBFEN(base, value) (DAC_RMW_C1(base,
DAC_C1_DACBFEN_MASK, DAC_C1_DACBFEN(value)))
#define DAC_BWR_C1_DACBFEN(base, value) (BME_BFI8(&DAC_C1_REG(base),
((uint8_t)(value) << DAC_C1_DACBFEN_SHIFT), DAC_C1_DACBFEN_SHIFT,
DAC_C1_DACBFEN_WIDTH))
/*@}*/ */

/*!
 * @name Register DAC_C1, field DACBFMD[2] (RW)

```

```

/*
 * Values:
 * - 0b0 - Normal mode
 * - 0b1 - One-Time Scan mode
 */
/*@{*/
/*! @brief Read current value of the DAC_C1_DACBFMD field. */
#define DAC_RD_C1_DACBFMD(base) ((DAC_C1_REG(base) & DAC_C1_DACBFMD_MASK) >> DAC_C1_DACBFMD_SHIFT)
#define DAC_BRD_C1_DACBFMD(base) (BME_UBFX8(&DAC_C1_REG(base), DAC_C1_DACBFMD_SHIFT, DAC_C1_DACBFMD_WIDTH))

/*! @brief Set the DACBFMD field to a new value. */
#define DAC_WR_C1_DACBFMD(base, value) (DAC_RMW_C1(base, DAC_C1_DACBFMD_MASK, DAC_C1_DACBFMD(value)))
#define DAC_BWR_C1_DACBFMD(base, value) (BME_BFI8(&DAC_C1_REG(base), (uint8_t)(value) << DAC_C1_DACBFMD_SHIFT), DAC_C1_DACBFMD_SHIFT, DAC_C1_DACBFMD_WIDTH)
/*@}*/
/*!
 * @name Register DAC_C1, field DMAEN[7] (RW)
 *
 * Values:
 * - 0b0 - DMA is disabled.
 * - 0b1 - DMA is enabled. When DMA is enabled, the DMA request will be
 *         generated by original interrupts. The interrupts will not be
 * presented on this
 *         module at the same time.
 */
/*@{*/
/*! @brief Read current value of the DAC_C1_DMAEN field. */
#define DAC_RD_C1_DMAEN(base) ((DAC_C1_REG(base) & DAC_C1_DMAEN_MASK) >> DAC_C1_DMAEN_SHIFT)
#define DAC_BRD_C1_DMAEN(base) (BME_UBFX8(&DAC_C1_REG(base), DAC_C1_DMAEN_SHIFT, DAC_C1_DMAEN_WIDTH))

/*! @brief Set the DMAEN field to a new value. */
#define DAC_WR_C1_DMAEN(base, value) (DAC_RMW_C1(base, DAC_C1_DMAEN_MASK, DAC_C1_DMAEN(value)))
#define DAC_BWR_C1_DMAEN(base, value) (BME_BFI8(&DAC_C1_REG(base), (uint8_t)(value) << DAC_C1_DMAEN_SHIFT), DAC_C1_DMAEN_SHIFT, DAC_C1_DMAEN_WIDTH)
/*@}*/

/*********************  

*****  

 * DAC_C2 - DAC Control Register 2  

*****/  

/*!  

 * @brief DAC_C2 - DAC Control Register 2 (RW)

```

```

/*
 * Reset value: 0x01U
 *
 * Do not use 32- or 16-bit accesses to this register.
 */
/*!
 * @name Constants and macros for entire DAC_C2 register
 */
/*@*/
#define DAC_RD_C2(base)          (DAC_C2_REG(base))
#define DAC_WR_C2(base, value)    (DAC_C2_REG(base) = (value))
#define DAC_RMW_C2(base, mask, value) (DAC_WR_C2(base, (DAC_RD_C2(base) &
~(mask)) | (value)))
#define DAC_SET_C2(base, value)   (BME_OR8(&DAC_C2_REG(base),
(uint8_t)(value)))
#define DAC_CLR_C2(base, value)   (BME_AND8(&DAC_C2_REG(base),
(uint8_t)(~(value))))
#define DAC_TOG_C2(base, value)   (BME_XOR8(&DAC_C2_REG(base),
(uint8_t)(value)))
/*@*/ */

/*
 * Constants & macros for individual DAC_C2 bitfields
 */
/*!
 * @name Register DAC_C2, field DACBFUP[0] (RW)
 *
 * Selects the upper limit of the DAC buffer. The buffer read pointer
cannot
 * exceed it.
 */
/*@*/
/*! @brief Read current value of the DAC_C2_DACBFUP field. */
#define DAC_RD_C2_DACBFUP(base) ((DAC_C2_REG(base) & DAC_C2_DACBFUP_MASK)
>> DAC_C2_DACBFUP_SHIFT)
#define DAC_BRD_C2_DACBFUP(base) (BME_UBFX8(&DAC_C2_REG(base),
DAC_C2_DACBFUP_SHIFT, DAC_C2_DACBFUP_WIDTH))

/*! @brief Set the DACBFUP field to a new value. */
#define DAC_WR_C2_DACBFUP(base, value) (DAC_RMW_C2(base,
DAC_C2_DACBFUP_MASK, DAC_C2_DACBFUP(value)))
#define DAC_BWR_C2_DACBFUP(base, value) (BME_BFI8(&DAC_C2_REG(base),
((uint8_t)(value) << DAC_C2_DACBFUP_SHIFT), DAC_C2_DACBFUP_SHIFT,
DAC_C2_DACBFUP_WIDTH))
/*@*/ */

/*!
 * @name Register DAC_C2, field DACBFRP[4] (RW)
 *
 * Keeps the current value of the buffer read pointer.
 */
/*@*/
/*! @brief Read current value of the DAC_C2_DACBFRP field. */

```

```

#define DAC_RD_C2_DACBFRP(base) ((DAC_C2_REG(base) & DAC_C2_DACBFRP_MASK) \
>> DAC_C2_DACBFRP_SHIFT)
#define DAC_BRD_C2_DACBFRP(base) (BME_UBFX8(&DAC_C2_REG(base), \
DAC_C2_DACBFRP_SHIFT, DAC_C2_DACBFRP_WIDTH))

/*! @brief Set the DACBFRP field to a new value. */
#define DAC_WR_C2_DACBFRP(base, value) (DAC_RMW_C2(base, \
DAC_C2_DACBFRP_MASK, DAC_C2_DACBFRP(value)))
#define DAC_BWR_C2_DACBFRP(base, value) (BME_BFI8(&DAC_C2_REG(base), \
((uint8_t)(value) << DAC_C2_DACBFRP_SHIFT), DAC_C2_DACBFRP_SHIFT, \
DAC_C2_DACBFRP_WIDTH))
/*@}*/

/*
 * MKL25Z4 DMA
 *
 * DMA Controller
 *
 * Registers defined in this header file:
 * - DMA_SAR - Source Address Register
 * - DMA_DAR - Destination Address Register
 * - DMA_DSR - DMA_DSR0 register.
 * - DMA_DSR_BCR - DMA Status Register / Byte Count Register
 * - DMA_DCR - DMA Control Register
 */
#define DMA_INSTANCE_COUNT (1U) /*!< Number of instances of the DMA module. */
#define DMA_IDX (0U) /*!< Instance number for DMA. */

/********************* DMA_SAR - Source Address Register ********************
*****
 * DMA_SAR - Source Address Register
*****
/********************* DMA_SAR - Source Address Register (RW)
*
* Reset value: 0x00000000U
*
* For this register: Only 32-bit writes are allowed. 16-bit and 8-bit
writes
* result in a bus error. Only four values are allowed to be written to
bits 31-20
* of this register. A write of any other value to these bits causes a
* configuration error when the channel starts to execute. For more
information about the
* configuration error, see the description of the CEConfiguration error
field of
* DSR.
*/
/*!

```

```

 * @name Constants and macros for entire DMA_SAR register
 */
/*@{*/
#define DMA_RD_SAR(base, index)  (DMA_SAR_REG(base, index))
#define DMA_WR_SAR(base, index, value)  (DMA_SAR_REG(base, index) =
(value))
#define DMA_RMW_SAR(base, index, mask, value)  (DMA_WR_SAR(base, index,
(DMA_RD_SAR(base, index) & ~mask) | (value)))
#define DMA_SET_SAR(base, index, value)  (BME_OR32(&DMA_SAR_REG(base,
index), (uint32_t)(value)))
#define DMA_CLR_SAR(base, index, value)  (BME_AND32(&DMA_SAR_REG(base,
index), (uint32_t)(~(value))))
#define DMA_TOG_SAR(base, index, value)  (BME_XOR32(&DMA_SAR_REG(base,
index), (uint32_t)(value)))
/*}@*/



/***** DMA_DAR - Destination Address Register *****/
*****/


/*!
 * @brief DMA_DAR - Destination Address Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * For this register: Only 32-bit writes are allowed. 16-bit and 8-bit
writes
 * result in a bus error. Only four values are allowed to be written to
bits 31-20
 * of this register. A write of any other value to these bits causes a
 * configuration error when the channel starts to execute. For more
information about the
 * configuration error, see the description of the CEConfiguration error
field of
 * DSR.
 */
/*!
 * @name Constants and macros for entire DMA_DAR register
*/
/*@{*/
#define DMA_RD_DAR(base, index)  (DMA_DAR_REG(base, index))
#define DMA_WR_DAR(base, index, value)  (DMA_DAR_REG(base, index) =
(value))
#define DMA_RMW_DAR(base, index, mask, value)  (DMA_WR_DAR(base, index,
(DMA_RD_DAR(base, index) & ~mask) | (value)))
#define DMA_SET_DAR(base, index, value)  (BME_OR32(&DMA_DAR_REG(base,
index), (uint32_t)(value)))
#define DMA_CLR_DAR(base, index, value)  (BME_AND32(&DMA_DAR_REG(base,
index), (uint32_t)(~(value))))
#define DMA_TOG_DAR(base, index, value)  (BME_XOR32(&DMA_DAR_REG(base,
index), (uint32_t)(value)))

```

```

/*@}*/

/*****
 * DMA_DSR_BCR - DMA Status Register / Byte Count Register
 *****/
*****/

/*!
 * @brief DMA_DSR_BCR - DMA Status Register / Byte Count Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * DSR and BCR are two logical registers that occupy one 32-bit address.
DSRn
 * occupies bits 31-24, and BCRn occupies bits 23-0. DSRn contains flags
indicating
 * the channel status, and BCRn contains the number of bytes yet to be
 * transferred for a given block. On the successful completion of the
write transfer, BCRn
 * decrements by 1, 2, or 4 for 8-bit, 16-bit, or 32-bit accesses,
respectively.
 * BCRn is cleared if a 1 is written to DSR[DONE]. In response to an
event, the
 * DMA controller writes to the appropriate DSRn bit. Only a write to
DSRn[DONE]
 * results in action. DSRn[DONE] is set when the block transfer is
complete. When
 * a transfer sequence is initiated and BCRn[BCR] is not a multiple of 4
or 2
 * when the DMA is configured for 32-bit or 16-bit transfers,
respectively,
 * DSRn[CE] is set and no transfer occurs.
 */
/*!
 * @name Constants and macros for entire DMA_DSR_BCR register
 */
/*@{*/
#define DMA_RD_DSR_BCR(base, index) (DMA_DSR_BCR_REG(base, index))
#define DMA_WR_DSR_BCR(base, index, value) (DMA_DSR_BCR_REG(base, index) \
= (value))
#define DMA_RMW_DSR_BCR(base, index, mask, value) (DMA_WR_DSR_BCR(base, \
index, (DMA_RD_DSR_BCR(base, index) & ~mask) | (value)))
#define DMA_SET_DSR_BCR(base, index, value)
(BME_OR32(&DMA_DSR_BCR_REG(base, index), (uint32_t)(value)))
#define DMA_CLR_DSR_BCR(base, index, value)
(BME_AND32(&DMA_DSR_BCR_REG(base, index), (uint32_t)(~(value))))
#define DMA_TOG_DSR_BCR(base, index, value)
(BME_XOR32(&DMA_DSR_BCR_REG(base, index), (uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual DMA_DSR_BCR bitfields

```

```

*/
/*!
 * @name Register DMA_DSR_BCR, field BCR[23:0] (RW)
 *
 * This field contains the number of bytes yet to be transferred for a
given
 * block. BCR must be written with a value equal to or less than
0F_FFFFh. After
 * being written with a value in this range, bits 23-20 of BCR read back
as 1110b. A
 * write to BCR of a value greater than 0F_FFFFh causes a configuration
error
 * when the channel starts to execute. After being written with a value
in this
 * range, bits 23-20 of BCR read back as 1111b.
 */
/*@{*/
/*! @brief Read current value of the DMA_DSR_BCR_BCR field. */
#define DMA_RD_DSR_BCR_BCR(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_BCR_MASK) >> DMA_DSR_BCR_BCR_SHIFT)
#define DMA_BRD_DSR_BCR_BCR(base, index) (DMA_RD_DSR_BCR_BCR(base,
index))

/*! @brief Set the BCR field to a new value. */
#define DMA_WR_DSR_BCR_BCR(base, index, value) (DMA_RMW_DSR_BCR(base,
index, (DMA_DSR_BCR_BCR_MASK | DMA_DSR_BCR_DONE_MASK),
DMA_DSR_BCR_BCR(value)))
#define DMA_BWR_DSR_BCR_BCR(base, index, value) (DMA_WR_DSR_BCR_BCR(base,
index, value))
/*}@*/ */

/*
 * @name Register DMA_DSR_BCR, field DONE[24] (W1C)
 *
 * Set when all DMA controller transactions complete as determined by
transfer
 * count, or based on error conditions. When BCR reaches zero, DONE is
set when
 * the final transfer completes successfully. DONE can also be used to
abort a
 * transfer by resetting the status bits. When a transfer completes,
software must
 * clear DONE before reprogramming the DMA.
 *
 * Values:
 * - 0b0 - DMA transfer is not yet complete. Writing a 0 has no effect.
 * - 0b1 - DMA transfer completed. Writing a 1 to this bit clears all DMA
status
 *        bits and should be used in an interrupt service routine to clear
the DMA
 *        interrupt and error bits.
 */
/*@{*/

```

```

/*! @brief Read current value of the DMA_DSR_BCR_DONE field. */
#define DMA_RD_DSR_BCR_DONE(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_DONE_MASK) >> DMA_DSR_BCR_DONE_SHIFT)
#define DMA_BRD_DSR_BCR_DONE(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_DONE_SHIFT,
DMA_DSR_BCR_DONE_WIDTH))

/*! @brief Set the DONE field to a new value. */
#define DMA_WR_DSR_BCR_DONE(base, index, value) (DMA_RMW_DSR_BCR(base,
index, DMA_DSR_BCR_DONE_MASK, DMA_DSR_BCR_DONE(value)))
#define DMA_BWR_DSR_BCR_DONE(base, index, value)
(BME_BFI32(&DMA_DSR_BCR_REG(base, index), ((uint32_t)(value) <<
DMA_DSR_BCR_DONE_SHIFT), DMA_DSR_BCR_DONE_SHIFT, DMA_DSR_BCR_DONE_WIDTH))
/*@}*/

/*
 * @name Register DMA_DSR_BCR, field BSY[25] (RO)
 *
 * Values:
 * - 0b0 - DMA channel is inactive. Cleared when the DMA has finished the
last
 *         transaction.
 * - 0b1 - BSY is set the first time the channel is enabled after a
transfer is
 *         initiated.
 */
/*@{*/
/*! @brief Read current value of the DMA_DSR_BCR_BSY field. */
#define DMA_RD_DSR_BCR_BSY(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_BSY_MASK) >> DMA_DSR_BCR_BSY_SHIFT)
#define DMA_BRD_DSR_BCR_BSY(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_BSY_SHIFT,
DMA_DSR_BCR_BSY_WIDTH))
/*@}*/

/*
 * @name Register DMA_DSR_BCR, field REQ[26] (RO)
 *
 * Values:
 * - 0b0 - No request is pending or the channel is currently active.
Cleared
 *         when the channel is selected.
 * - 0b1 - The DMA channel has a transfer remaining and the channel is
not
 *         selected.
 */
/*@{*/
/*! @brief Read current value of the DMA_DSR_BCR_REQ field. */
#define DMA_RD_DSR_BCR_REQ(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_REQ_MASK) >> DMA_DSR_BCR_REQ_SHIFT)
#define DMA_BRD_DSR_BCR_REQ(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_REQ_SHIFT,
DMA_DSR_BCR_REQ_WIDTH))
/*@}*/

```

```

/*!
 * @name Register DMA_DSR_BCR, field BED[28] (RO)
 *
 * BED is cleared at hardware reset or by writing a 1 to the DONE bit.
 *
 * Values:
 * - 0b0 - No bus error occurred.
 * - 0b1 - The DMA channel terminated with a bus error during the write
portion
 *      of a transfer.
 */
/*@{*/
/*! @brief Read current value of the DMA_DSR_BCR_BED field. */
#define DMA_RD_DSR_BCR_BED(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_BED_MASK) >> DMA_DSR_BCR_BED_SHIFT)
#define DMA_BRD_DSR_BCR_BED(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_BED_SHIFT,
DMA_DSR_BCR_BED_WIDTH))
/*@}*/
/*!
 * @name Register DMA_DSR_BCR, field BES[29] (RO)
 *
 * BES is cleared at hardware reset or by writing a 1 to the DONE bit.
 *
 * Values:
 * - 0b0 - No bus error occurred.
 * - 0b1 - The DMA channel terminated with a bus error during the read
portion
 *      of a transfer.
 */
/*@{*/
/*! @brief Read current value of the DMA_DSR_BCR_BES field. */
#define DMA_RD_DSR_BCR_BES(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_BES_MASK) >> DMA_DSR_BCR_BES_SHIFT)
#define DMA_BRD_DSR_BCR_BES(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_BES_SHIFT,
DMA_DSR_BCR_BES_WIDTH))
/*@}*/
/*!
 * @name Register DMA_DSR_BCR, field CE[30] (RO)
 *
 * Any of the following conditions causes a configuration error: BCR,
SAR, or
 * DAR does not match the requested transfer size. A value greater than
0F_FFFFh is
 * written to BCR. Bits 31-20 of SAR or DAR are written with a value
other than
 * one of the allowed values. See SAR and DAR . SSIZE or DSIZE is set to
an
 * unsupported value. BCR equals 0 when the DMA receives a start
condition. CE is

```

```

* cleared at hardware reset or by writing a 1 to the DONE bit.
*
* Values:
* - 0b0 - No configuration error exists.
* - 0b1 - A configuration error has occurred.
*/
/*@{ */
/*! @brief Read current value of the DMA_DSR_BCR_CE field. */
#define DMA_RD_DSR_BCR_CE(base, index) ((DMA_DSR_BCR_REG(base, index) &
DMA_DSR_BCR_CE_MASK) >> DMA_DSR_BCR_CE_SHIFT)
#define DMA_BRD_DSR_BCR_CE(base, index)
(BME_UBFX32(&DMA_DSR_BCR_REG(base, index), DMA_DSR_BCR_CE_SHIFT,
DMA_DSR_BCR_CE_WIDTH))
/*@} */

/********************* DMA_DSR - DMA_DSR0 register. *****

* DMA_DSR - DMA_DSR0 register. (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire DMA_DSR register
*/
/*@{ */
#define DMA_RD_DSR(base, index) (DMA_DSR_REG(base, index))
#define DMA_WR_DSR(base, index, value) (DMA_DSR_REG(base, index) =
(value))
#define DMA_RMW_DSR(base, index, mask, value) (DMA_WR_DSR(base, index,
(DMA_RD_DSR(base, index) & ~mask) | value)))
#define DMA_SET_DSR(base, index, value) (BME_OR8(&DMA_DSR_REG(base,
index), (uint8_t)(value)))
#define DMA_CLR_DSR(base, index, value) (BME_AND8(&DMA_DSR_REG(base,
index), (uint8_t)(~value)))
#define DMA_TOG_DSR(base, index, value) (BME_XOR8(&DMA_DSR_REG(base,
index), (uint8_t)(value)))
/*@} */

/********************* DMA_DCR - DMA Control Register *****

* DMA_DCR - DMA Control Register (RW)
*

```

```

 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire DMA_DCR register
 */
/*@{*/
#define DMA_RD_DCR(base, index) (DMA_DCR_REG(base, index))
#define DMA_WR_DCR(base, index, value) (DMA_DCR_REG(base, index) = (value))
#define DMA_RMW_DCR(base, index, mask, value) (DMA_WR_DCR(base, index, (DMA_RD_DCR(base, index) & ~mask) | (value)))
#define DMA_SET_DCR(base, index, value) (BME_OR32(&DMA_DCR_REG(base, index), (uint32_t)(value)))
#define DMA_CLR_DCR(base, index, value) (BME_AND32(&DMA_DCR_REG(base, index), (uint32_t)(~(value))))
#define DMA_TOG_DCR(base, index, value) (BME_XOR32(&DMA_DCR_REG(base, index), (uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual DMA_DCR bitfields
 */

/*!
 * @name Register DMA_DCR, field LCH2[1:0] (RW)
 *
 * Indicates the DMA channel assigned as link channel 2. The link channel number
 * cannot be the same as the currently executing channel, and generates a
 * configuration error if this is attempted (DSRn[CE] is set).
 *
 * Values:
 * - 0b00 - DMA Channel 0
 * - 0b01 - DMA Channel 1
 * - 0b10 - DMA Channel 2
 * - 0b11 - DMA Channel 3
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_LCH2 field. */
#define DMA_RD_DCR_LCH2(base, index) ((DMA_DCR_REG(base, index) & DMA_DCR_LCH2_MASK) >> DMA_DCR_LCH2_SHIFT)
#define DMA_BRD_DCR_LCH2(base, index) (BME_UBFX32(&DMA_DCR_REG(base, index), DMA_DCR_LCH2_SHIFT, DMA_DCR_LCH2_WIDTH))

/*! @brief Set the LCH2 field to a new value. */
#define DMA_WR_DCR_LCH2(base, index, value) (DMA_RMW_DCR(base, index, DMA_DCR_LCH2_MASK, DMA_DCR_LCH2(value)))
#define DMA_BWR_DCR_LCH2(base, index, value) (BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) << DMA_DCR_LCH2_SHIFT), DMA_DCR_LCH2_SHIFT, DMA_DCR_LCH2_WIDTH))
/*}@*/ */

/*
 * @name Register DMA_DCR, field LCH1[3:2] (RW)

```

```

/*
 * Indicates the DMA channel assigned as link channel 1. The link channel
number
 * cannot be the same as the currently executing channel, and generates a
 * configuration error if this is attempted (DSRn[CE] is set).
 *
 * Values:
 * - 0b00 - DMA Channel 0
 * - 0b01 - DMA Channel 1
 * - 0b10 - DMA Channel 2
 * - 0b11 - DMA Channel 3
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_LCH1 field. */
#define DMA_RD_DCR_LCH1(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_LCH1_MASK) >> DMA_DCR_LCH1_SHIFT)
#define DMA_BRD_DCR_LCH1(base, index) (BME_UBX32(&DMA_DCR_REG(base,
index), DMA_DCR_LCH1_SHIFT, DMA_DCR_LCH1_WIDTH))

/*! @brief Set the LCH1 field to a new value. */
#define DMA_WR_DCR_LCH1(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_LCH1_MASK, DMA_DCR_LCH1(value)))
#define DMA_BWR_DCR_LCH1(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_LCH1_SHIFT), DMA_DCR_LCH1_SHIFT, DMA_DCR_LCH1_WIDTH))
/*@}*/

/*
 * @name Register DMA_DCR, field LINKCC[5:4] (RW)
 *
 * Allows DMA channels to have their transfers linked. The current DMA
channel
 * triggers a DMA request to the linked channels (LCH1 or LCH2) depending
on the
 * condition described by the LINKCC bits. If not in cycle steal mode
(DCRn[CS]=0)
 * and LINKCC equals 01 or 10, no link to LCH1 occurs. If LINKCC equals
01, a
 * link to LCH1 is created after each cycle-steal transfer performed by
the current
 * DMA channel is completed. As the last cycle-steal is performed and the
BCR
 * reaches zero, then the link to LCH1 is closed and a link to LCH2 is
created.
 *
 * Values:
 * - 0b00 - No channel-to-channel linking
 * - 0b01 - Perform a link to channel LCH1 after each cycle-steal
transfer
 *      followed by a link to LCH2 after the BCR decrements to zero
 * - 0b10 - Perform a link to channel LCH1 after each cycle-steal
transfer
 * - 0b11 - Perform a link to channel LCH1 after the BCR decrements to
zero

```

```

/*
/*@{*/
/*! @brief Read current value of the DMA_DCR_LINKCC field. */
#define DMA_RD_DCR_LINKCC(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_LINKCC_MASK) >> DMA_DCR_LINKCC_SHIFT)
#define DMA_BRD_DCR_LINKCC(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_LINKCC_SHIFT, DMA_DCR_LINKCC_WIDTH))

/*! @brief Set the LINKCC field to a new value. */
#define DMA_WR_DCR_LINKCC(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_LINKCC_MASK, DMA_DCR_LINKCC(value)))
#define DMA_BWR_DCR_LINKCC(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_LINKCC_SHIFT), DMA_DCR_LINKCC_SHIFT, DMA_DCR_LINKCC_WIDTH))
/*}@*/
```

/\*!

- \* @name Register DMA\_DCR, field D\_REQ[7] (RW)
- \*
- \* DMA hardware automatically clears the corresponding DCRn[ERQ] bit when the byte count register reaches zero.
- \*
- \* Values:
  - \* - 0b0 - ERQ bit is not affected.
  - \* - 0b1 - ERQ bit is cleared when the BCR is exhausted.
- \*/

```
/*@{*/
/*! @brief Read current value of the DMA_DCR_D_REQ field. */
#define DMA_RD_DCR_D_REQ(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_D_REQ_MASK) >> DMA_DCR_D_REQ_SHIFT)
#define DMA_BRD_DCR_D_REQ(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_D_REQ_SHIFT, DMA_DCR_D_REQ_WIDTH))

/*! @brief Set the D_REQ field to a new value. */
#define DMA_WR_DCR_D_REQ(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_D_REQ_MASK, DMA_DCR_D_REQ(value)))
#define DMA_BWR_DCR_D_REQ(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_D_REQ_SHIFT), DMA_DCR_D_REQ_SHIFT, DMA_DCR_D_REQ_WIDTH))
/*}@*/
```

/\*!

- \* @name Register DMA\_DCR, field DMOD[11:8] (RW)
- \*
- \* Defines the size of the destination data circular buffer used by the DMA
  - \* Controller. If enabled (DMOD value is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary depends on the
    - \* initial destination address (DAR). The base address should be aligned to a

```

 * 0-modulo-(circular buffer size) boundary. Misaligned buffers are not
possible.
 * The boundary is forced to the value determined by the upper address
bits in the
 * field selection.
 *
 * Values:
 * - 0b0000 - Buffer disabled
 * - 0b0001 - Circular buffer size is 16 bytes
 * - 0b0010 - Circular buffer size is 32 bytes
 * - 0b0011 - Circular buffer size is 64 bytes
 * - 0b0100 - Circular buffer size is 128 bytes
 * - 0b0101 - Circular buffer size is 256 bytes
 * - 0b0110 - Circular buffer size is 512 bytes
 * - 0b0111 - Circular buffer size is 1 KB
 * - 0b1000 - Circular buffer size is 2 KB
 * - 0b1001 - Circular buffer size is 4 KB
 * - 0b1010 - Circular buffer size is 8 KB
 * - 0b1011 - Circular buffer size is 16 KB
 * - 0b1100 - Circular buffer size is 32 KB
 * - 0b1101 - Circular buffer size is 64 KB
 * - 0b1110 - Circular buffer size is 128 KB
 * - 0b1111 - Circular buffer size is 256 KB
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_DMOD field. */
#define DMA_RD_DCR_DMOD(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_DMOD_MASK) >> DMA_DCR_DMOD_SHIFT)
#define DMA_BRD_DCR_DMOD(base, index) (BME_UBXF32(&DMA_DCR_REG(base,
index), DMA_DCR_DMOD_SHIFT, DMA_DCR_DMOD_WIDTH))

/*! @brief Set the DMOD field to a new value. */
#define DMA_WR_DCR_DMOD(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_DMOD_MASK, DMA_DCR_DMOD(value)))
#define DMA_BWR_DCR_DMOD(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_DMOD_SHIFT), DMA_DCR_DMOD_SHIFT, DMA_DCR_DMOD_WIDTH))
/*@}*/
/*!
 * @name Register DMA_DCR, field SMOD[15:12] (RW)
 *
 * Defines the size of the source data circular buffer used by the DMA
 * Controller. If enabled (SMOD is non-zero), the buffer base address is
located on a
 * boundary of the buffer size. The value of this boundary is based upon
the initial
 * source address (SAR). The base address should be aligned to a
 * 0-modulo-(circular buffer size) boundary. Misaligned buffers are not
possible. The boundary is
 * forced to the value determined by the upper address bits in the field
 * selection.
 *
 * Values:

```

```

* - 0b0000 - Buffer disabled
* - 0b0001 - Circular buffer size is 16 bytes
* - 0b0010 - Circular buffer size is 32 bytes
* - 0b0011 - Circular buffer size is 64 bytes
* - 0b0100 - Circular buffer size is 128 bytes
* - 0b0101 - Circular buffer size is 256 bytes
* - 0b0110 - Circular buffer size is 512 bytes
* - 0b0111 - Circular buffer size is 1 KB
* - 0b1000 - Circular buffer size is 2 KB
* - 0b1001 - Circular buffer size is 4 KB
* - 0b1010 - Circular buffer size is 8 KB
* - 0b1011 - Circular buffer size is 16 KB
* - 0b1100 - Circular buffer size is 32 KB
* - 0b1101 - Circular buffer size is 64 KB
* - 0b1110 - Circular buffer size is 128 KB
* - 0b1111 - Circular buffer size is 256 KB
*/
/*@{*/
/*! @brief Read current value of the DMA_DCR_SMOD field. */
#define DMA_RD_DCR_SMOD(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_SMOD_MASK) >> DMA_DCR_SMOD_SHIFT)
#define DMA_BRD_DCR_SMOD(base, index) (BME_BFX32(&DMA_DCR_REG(base,
index), DMA_DCR_SMOD_SHIFT, DMA_DCR_SMOD_WIDTH))

/*! @brief Set the SMOD field to a new value. */
#define DMA_WR_DCR_SMOD(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_SMOD_MASK, DMA_DCR_SMOD(value)))
#define DMA_BWR_DCR_SMOD(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_SMOD_SHIFT), DMA_DCR_SMOD_SHIFT, DMA_DCR_SMOD_WIDTH))
/*}@*/ */

/*!
* @name Register DMA_DCR, field START[16] (WORZ)
*
* Values:
* - 0b0 - DMA inactive
* - 0b1 - The DMA begins the transfer in accordance to the values in the
TCDn.
*     START is cleared automatically after one module clock and always
reads as
*     logic 0.
*/
/*@{*/
/*! @brief Set the START field to a new value. */
#define DMA_WR_DCR_START(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_START_MASK, DMA_DCR_START(value)))
#define DMA_BWR_DCR_START(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_START_SHIFT), DMA_DCR_START_SHIFT, DMA_DCR_START_WIDTH))
/*}@*/ */

/*!
* @name Register DMA_DCR, field DSIZE[18:17] (RW)

```

```

/*
 * Determines the data size of the destination bus cycle for the DMA
controller.
 *
 * Values:
 * - 0b00 - 32-bit
 * - 0b01 - 8-bit
 * - 0b10 - 16-bit
 * - 0b11 - Reserved (generates a configuration error (DSRn[CE]) if
incorrectly
 *           specified at time of channel activation)
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_DSIZEx field. */
#define DMA_RD_DCR_DSIZEx(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_DSIZEx_MASK) >> DMA_DCR_DSIZEx_SHIFT)
#define DMA_BRD_DCR_DSIZEx(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_DSIZEx_SHIFT, DMA_DCR_DSIZEx_WIDTH))

/*! @brief Set the DSIZEx field to a new value. */
#define DMA_WR_DCR_DSIZEx(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_DSIZEx_MASK, DMA_DCR_DSIZEx(value)))
#define DMA_BWR_DCR_DSIZEx(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_DSIZEx_SHIFT), DMA_DCR_DSIZEx_SHIFT, DMA_DCR_DSIZEx_WIDTH))
/*}@*/ */

/*!
 * @name Register DMA_DCR, field DINC[19] (RW)
 *
 * Controls whether the destination address increments after each
successful
 * transfer.
 *
 * Values:
 * - 0b0 - No change to the DAR after a successful transfer.
 * - 0b1 - The DAR increments by 1, 2, 4 depending upon the size of the
transfer.
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_DINC field. */
#define DMA_RD_DCR_DINC(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_DINC_MASK) >> DMA_DCR_DINC_SHIFT)
#define DMA_BRD_DCR_DINC(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_DINC_SHIFT, DMA_DCR_DINC_WIDTH))

/*! @brief Set the DINC field to a new value. */
#define DMA_WR_DCR_DINC(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_DINC_MASK, DMA_DCR_DINC(value)))
#define DMA_BWR_DCR_DINC(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_DINC_SHIFT), DMA_DCR_DINC_SHIFT, DMA_DCR_DINC_WIDTH))
/*}@*/

```

```

/*!
 * @name Register DMA_DCR, field SSIZE[21:20] (RW)
 *
 * Determines the data size of the source bus cycle for the DMA
controller.
 *
 * Values:
 * - 0b00 - 32-bit
 * - 0b01 - 8-bit
 * - 0b10 - 16-bit
 * - 0b11 - Reserved (generates a configuration error (DSRn[CE])) if
incorrectly
 *           specified at time of channel activation)
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_SSIZEx field. */
#define DMA_RD_DCR_SSIZEx(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_SSIZEx_MASK) >> DMA_DCR_SSIZEx_SHIFT)
#define DMA_BRD_DCR_SSIZEx(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_SSIZEx_SHIFT, DMA_DCR_SSIZEx_WIDTH))

/*! @brief Set the SSIZEx field to a new value. */
#define DMA_WR_DCR_SSIZEx(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_SSIZEx_MASK, DMA_DCR_SSIZEx(value)))
#define DMA_BWR_DCR_SSIZEx(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_SSIZEx_SHIFT), DMA_DCR_SSIZEx_SHIFT, DMA_DCR_SSIZEx_WIDTH))
/*@}*/
/*!
 * @name Register DMA_DCR, field SINC[22] (RW)
 *
 * Controls whether the source address increments after each successful
transfer.
 *
 * Values:
 * - 0b0 - No change to SAR after a successful transfer.
 * - 0b1 - The SAR increments by 1, 2, 4 as determined by the transfer
size.
 */
/*@{*/
/*! @brief Read current value of the DMA_DCR_SINC field. */
#define DMA_RD_DCR_SINC(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_SINC_MASK) >> DMA_DCR_SINC_SHIFT)
#define DMA_BRD_DCR_SINC(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_SINC_SHIFT, DMA_DCR_SINC_WIDTH))

/*! @brief Set the SINC field to a new value. */
#define DMA_WR_DCR_SINC(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_SINC_MASK, DMA_DCR_SINC(value)))
#define DMA_BWR_DCR_SINC(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_SINC_SHIFT), DMA_DCR_SINC_SHIFT, DMA_DCR_SINC_WIDTH))
/*@}*/

```

```

/*!!
 * @name Register DMA_DCR, field EADREQ[23] (RW)
 *
 * Enables the channel to support asynchronous DREQs while the MCU is in
Stop
* mode.
*
* Values:
* - 0b0 - Disabled
* - 0b1 - Enabled
*/
/*@{*/
/*! @brief Read current value of the DMA_DCR_EADREQ field. */
#define DMA_RD_DCR_EADREQ(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_EADREQ_MASK) >> DMA_DCR_EADREQ_SHIFT)
#define DMA_BRD_DCR_EADREQ(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_EADREQ_SHIFT, DMA_DCR_EADREQ_WIDTH))

/*! @brief Set the EADREQ field to a new value. */
#define DMA_WR_DCR_EADREQ(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_EADREQ_MASK, DMA_DCR_EADREQ(value)))
#define DMA_BWR_DCR_EADREQ(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) <<
DMA_DCR_EADREQ_SHIFT), DMA_DCR_EADREQ_SHIFT, DMA_DCR_EADREQ_WIDTH))
/*@}*/
/*!
 * @name Register DMA_DCR, field AA[28] (RW)
*
* AA and SIZE bits determine whether the source or destination is auto-
aligned;
* that is, transfers are optimized based on the address and size.
*
* Values:
* - 0b0 - Auto-align disabled
* - 0b1 - If SSIZE indicates a transfer no smaller than DSIZE, source
accesses
*         are auto-aligned; otherwise, destination accesses are auto-
aligned. Source
*         alignment takes precedence over destination alignment. If auto-
alignment
*         is enabled, the appropriate address register increments,
regardless of
*         DINC or SINC.
*/
/*@{*/
/*! @brief Read current value of the DMA_DCR_AA field. */
#define DMA_RD_DCR_AA(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_AA_MASK) >> DMA_DCR_AA_SHIFT)
#define DMA_BRD_DCR_AA(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_AA_SHIFT, DMA_DCR_AA_WIDTH))

/*! @brief Set the AA field to a new value. */

```

```

#define DMA_WR_DCR_AA(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_AA_MASK, DMA_DCR_AA(value)))
#define DMA_BWR_DCR_AA(base, index, value) (BME_BFI32(&DMA_DCR_REG(base,
index), ((uint32_t)(value) << DMA_DCR_AA_SHIFT), DMA_DCR_AA_SHIFT,
DMA_DCR_AA_WIDTH))
/*@}*/

/*
 * @name Register DMA_DCR, field CS[29] (RW)
 *
 * Values:
 * - 0b0 - DMA continuously makes read/write transfers until the BCR
decrements
 *      to 0.
 * - 0b1 - Forces a single read/write transfer per request.
 */
/*@*/
/*! @brief Read current value of the DMA_DCR_CS field. */
#define DMA_RD_DCR_CS(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_CS_MASK) >> DMA_DCR_CS_SHIFT)
#define DMA_BRD_DCR_CS(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_CS_SHIFT, DMA_DCR_CS_WIDTH))

/*! @brief Set the CS field to a new value. */
#define DMA_WR_DCR_CS(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_CS_MASK, DMA_DCR_CS(value)))
#define DMA_BWR_DCR_CS(base, index, value) (BME_BFI32(&DMA_DCR_REG(base,
index), ((uint32_t)(value) << DMA_DCR_CS_SHIFT), DMA_DCR_CS_SHIFT,
DMA_DCR_CS_WIDTH))
/*@*/

/*
 * @name Register DMA_DCR, field ERQ[30] (RW)
 *
 * Be careful: a collision can occur between the START bit and D_REQ when
the
 * ERQ bit is 1.
 *
 * Values:
 * - 0b0 - Peripheral request is ignored.
 * - 0b1 - Enables peripheral request to initiate transfer. A software-
initiated
 *      request (setting the START bit) is always enabled.
 */
/*@*/
/*! @brief Read current value of the DMA_DCR_ERQ field. */
#define DMA_RD_DCR_ERQ(base, index) ((DMA_DCR_REG(base, index) &
DMA_DCR_ERQ_MASK) >> DMA_DCR_ERQ_SHIFT)
#define DMA_BRD_DCR_ERQ(base, index) (BME_UBFX32(&DMA_DCR_REG(base,
index), DMA_DCR_ERQ_SHIFT, DMA_DCR_ERQ_WIDTH))

/*! @brief Set the ERQ field to a new value. */
#define DMA_WR_DCR_ERQ(base, index, value) (DMA_RMW_DCR(base, index,
DMA_DCR_ERQ_MASK, DMA_DCR_ERQ(value)))

```

```

#define DMA_BWR_DCR_ERQ(base, index, value) (BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) << DMA_DCR_ERQ_SHIFT), DMA_DCR_ERQ_SHIFT, DMA_DCR_ERQ_WIDTH))
/*@}*/

/*
 * @name Register DMA_DCR, field EINT[31] (RW)
 *
 * Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition.
 *
 * Values:
 * - 0b0 - No interrupt is generated.
 * - 0b1 - Interrupt signal is enabled.
 */
/*@*/
/*! @brief Read current value of the DMA_DCR_EINT field. */
#define DMA_RD_DCR_EINT(base, index) ((DMA_DCR_REG(base, index) & DMA_DCR_EINT_MASK) >> DMA_DCR_EINT_SHIFT)
#define DMA_BRD_DCR_EINT(base, index) (BME_UBFX32(&DMA_DCR_REG(base, index), DMA_DCR_EINT_SHIFT, DMA_DCR_EINT_WIDTH))

/*! @brief Set the EINT field to a new value. */
#define DMA_WR_DCR_EINT(base, index, value) (DMA_RMW_DCR(base, index, DMA_DCR_EINT_MASK, DMA_DCR_EINT(value)))
#define DMA_BWR_DCR_EINT(base, index, value)
(BME_BFI32(&DMA_DCR_REG(base, index), ((uint32_t)(value) << DMA_DCR_EINT_SHIFT), DMA_DCR_EINT_SHIFT, DMA_DCR_EINT_WIDTH))
/*@*/

/*
 * MKL25Z4 DMAMUX
 *
 * DMA channel multiplexor
 *
 * Registers defined in this header file:
 * - DMAMUX_CHCFG - Channel Configuration register
 */
/* ***** */

#define DMAMUX_INSTANCE_COUNT (1U) /*!< Number of instances of the DMAMUX module. */
#define DMAMUX0_IDX (0U) /*!< Instance number for DMAMUX0. */

/* **** */
* DMAMUX_CHCFG - Channel Configuration register
/* **** */
* / ****

/*!
 * @brief DMAMUX_CHCFG - Channel Configuration register (RW)
 *

```

```

 * Reset value: 0x00U
 *
 * Each of the DMA channels can be independently enabled/disabled and
associated
 * with one of the DMA slots (peripheral slots or always-on slots) in the
 * system. Setting multiple CHCFG registers with the same Source value
will result in
 * unpredictable behavior. Before changing the trigger or source settings
a DMA
 * channel must be disabled via the CHCFGn[ENBL] bit.
 */
/*!
 * @name Constants and macros for entire DMAMUX_CHCFG register
 */
/*@{*/
#define DMAMUX_RD_CHCFG(base, index) (DMAMUX_CHCFG_REG(base, index))
#define DMAMUX_WR_CHCFG(base, index, value) (DMAMUX_CHCFG_REG(base,
index) = (value))
#define DMAMUX_RMW_CHCFG(base, index, mask, value) (DMAMUX_WR_CHCFG(base,
index, (DMAMUX_RD_CHCFG(base, index) & ~mask) | value)))
#define DMAMUX_SET_CHCFG(base, index, value)
(BME_OR8(&DMAMUX_CHCFG_REG(base, index), (uint8_t)(value)))
#define DMAMUX_CLR_CHCFG(base, index, value)
(BME_AND8(&DMAMUX_CHCFG_REG(base, index), (uint8_t)(~value)))
#define DMAMUX_TOG_CHCFG(base, index, value)
(BME_XOR8(&DMAMUX_CHCFG_REG(base, index), (uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual DMAMUX_CHCFG bitfields
*/
/*!
 * @name Register DMAMUX_CHCFG, field SOURCE[5:0] (RW)
 *
 * Specifies which DMA source, if any, is routed to a particular DMA
channel.
 * See your device's chip configuration details for further details about
the
 * peripherals and their slot numbers.
 */
/*@{*/
/*! @brief Read current value of the DMAMUX_CHCFG_SOURCE field. */
#define DMAMUX_RD_CHCFG_SOURCE(base, index) ((DMAMUX_CHCFG_REG(base,
index) & DMAMUX_CHCFG_SOURCE_MASK) >> DMAMUX_CHCFG_SOURCE_SHIFT)
#define DMAMUX_BRD_CHCFG_SOURCE(base, index)
(BME_UBFX8(&DMAMUX_CHCFG_REG(base, index), DMAMUX_CHCFG_SOURCE_SHIFT,
DMAMUX_CHCFG_SOURCE_WIDTH))

/*! @brief Set the SOURCE field to a new value. */
#define DMAMUX_WR_CHCFG_SOURCE(base, index, value)
(DMAMUX_RMW_CHCFG(base, index, DMAMUX_CHCFG_SOURCE_MASK,
DMAMUX_CHCFG_SOURCE(value)))

```

```

#define DMAMUX_BWR_CHCFG_SOURCE(base, index, value)
(BME_BFI8(&DMAMUX_CHCFG_REG(base, index), ((uint8_t)(value) <<
DMAMUX_CHCFG_SOURCE_SHIFT), DMAMUX_CHCFG_SOURCE_SHIFT,
DMAMUX_CHCFG_SOURCE_WIDTH))
/*@}*/

/*!!
 * @name Register DMAMUX_CHCFG, field TRIG[6] (RW)
 *
 * Enables the periodic trigger capability for the triggered DMA channel.
 *
 * Values:
 * - 0b0 - Triggering is disabled. If triggering is disabled, and the ENBL bit
 *   is set, the DMA Channel will simply route the specified source to the DMA
 *   channel. (Normal mode)
 * - 0b1 - Triggering is enabled. If triggering is enabled, and the ENBL bit is
 *   set, the DMAMUX is in Periodic Trigger mode.
 */
/*@{*/
/*! @brief Read current value of the DMAMUX_CHCFG_TRIG field. */
#define DMAMUX_RD_CHCFG_TRIG(base, index) ((DMAMUX_CHCFG_REG(base, index)
& DMAMUX_CHCFG_TRIG_MASK) >> DMAMUX_CHCFG_TRIG_SHIFT)
#define DMAMUX_BRD_CHCFG_TRIG(base, index)
(BME_UBFX8(&DMAMUX_CHCFG_REG(base, index), DMAMUX_CHCFG_TRIG_SHIFT,
DMAMUX_CHCFG_TRIG_WIDTH))

/*! @brief Set the TRIG field to a new value. */
#define DMAMUX_WR_CHCFG_TRIG(base, index, value) (DMAMUX_RMW_CHCFG(base,
index, DMAMUX_CHCFG_TRIG_MASK, DMAMUX_CHCFG_TRIG(value)))
#define DMAMUX_BWR_CHCFG_TRIG(base, index, value)
(BME_BFI8(&DMAMUX_CHCFG_REG(base, index), ((uint8_t)(value) <<
DMAMUX_CHCFG_TRIG_SHIFT), DMAMUX_CHCFG_TRIG_SHIFT,
DMAMUX_CHCFG_TRIG_WIDTH))
/*@}*/

/*!!
 * @name Register DMAMUX_CHCFG, field ENBL[7] (RW)
 *
 * Enables the DMA channel.
 *
 * Values:
 * - 0b0 - DMA channel is disabled. This mode is primarily used during
 *   configuration of the DMA Mux. The DMA has separate channel enables/disables, which
 *   should be used to disable or re-configure a DMA channel.
 * - 0b1 - DMA channel is enabled
 */
/*@{*/
/*! @brief Read current value of the DMAMUX_CHCFG_ENBL field. */
#define DMAMUX_RD_CHCFG_ENBL(base, index) ((DMAMUX_CHCFG_REG(base, index)
& DMAMUX_CHCFG_ENBL_MASK) >> DMAMUX_CHCFG_ENBL_SHIFT)

```

```

#define DMAMUX_BRD_CHCFG_ENBL(base, index)
(BME_UBFX8(&DMAMUX_CHCFG_REG(base, index), DMAMUX_CHCFG_ENBL_SHIFT,
DMAMUX_CHCFG_ENBL_WIDTH))

/*! @brief Set the ENBL field to a new value. */
#define DMAMUX_WR_CHCFG_ENBL(base, index, value) (DMAMUX_RMW_CHCFG(base,
index, DMAMUX_CHCFG_ENBL_MASK, DMAMUX_CHCFG_ENBL(value)))
#define DMAMUX_BWR_CHCFG_ENBL(base, index, value)
(BME_BFI8(&DMAMUX_CHCFG_REG(base, index), ((uint8_t)(value) <<
DMAMUX_CHCFG_ENBL_SHIFT), DMAMUX_CHCFG_ENBL_SHIFT,
DMAMUX_CHCFG_ENBL_WIDTH))
/*@}*/

/*
 * MKL25Z4 GPIO
 *
 * General Purpose Input/Output
 *
 * Registers defined in this header file:
 * - GPIO_PDOR - Port Data Output Register
 * - GPIO_PSOR - Port Set Output Register
 * - GPIO_PCOR - Port Clear Output Register
 * - GPIO_PTOR - Port Toggle Output Register
 * - GPIO_PDIR - Port Data Input Register
 * - GPIO_PDDR - Port Data Direction Register
 */

#define GPIO_INSTANCE_COUNT (5U) /*!< Number of instances of the GPIO
module. */

#define GPIOA_IDX (0U) /*!< Instance number for GPIOA. */
#define GPIOB_IDX (1U) /*!< Instance number for GPIOB. */
#define GPIOC_IDX (2U) /*!< Instance number for GPIOC. */
#define GPIOD_IDX (3U) /*!< Instance number for GPIOD. */
#define GPIOE_IDX (4U) /*!< Instance number for GPIOE. */

*****  

* GPIO_PDOR - Port Data Output Register  

*****  

/*!  

 * @brief GPIO_PDOR - Port Data Output Register (RW)  

 * Reset value: 0x00000000U  

 * This register configures the logic levels that are driven on each  

 * general-purpose output pins.  

 */  

/*!  

 * @name Constants and macros for entire GPIO_PDOR register  

 */  

/*@{ */

```

```

#define FGPIO_RD_PDOR(base)      (FGPIO_PDOR_REG(base))
#define FGPIO_WR_PDOR(base, value) (FGPIO_PDOR_REG(base) = (value))
#define FGPIO_RMW_PDOR(base, mask, value) (FGPIO_WR_PDOR(base,
(FGpio_RD_PDOR(base) & ~mask) | (value)))
#define FGPIO_SET_PDOR(base, value) (FGPIO_WR_PDOR(base,
FGPIO_RD_PDOR(base) | (value)))
#define FGPIO_CLR_PDOR(base, value) (FGPIO_WR_PDOR(base,
FGPIO_RD_PDOR(base) & ~value))
#define FGPIO_TOG_PDOR(base, value) (FGPIO_WR_PDOR(base,
FGPIO_RD_PDOR(base) ^ (value)))
/*@}*/

/*****!
 * @brief FGPIO_PSOR - Port Set Output Register
 *****/
/*!
 * @name Constants and macros for entire FGPIO_PSOR register
 */
/*!
 * @brief FGPIO_PSOR - Port Set Output Register (WORZ)
 *
 * Reset value: 0x00000000U
 *
 * This register configures whether to set the fields of the PDOR.
 */
/*!
 * @name Constants and macros for entire FGPIO_PSOR register
 */
/*@{*/
#define FGPIO_RD_PSOR(base)      (FGPIO_PSOR_REG(base))
#define FGPIO_WR_PSOR(base, value) (FGPIO_PSOR_REG(base) = (value))
#define FGPIO_RMW_PSOR(base, mask, value) (FGPIO_WR_PSOR(base,
(FGpio_RD_PSOR(base) & ~mask) | (value)))
/*@}*/

/*****!
 * @brief FGPIO_PCOR - Port Clear Output Register
 *****/
/*!
 * @name Constants and macros for entire FGPIO_PCOR register
 */
/*!
 * @brief FGPIO_PCOR - Port Clear Output Register (WORZ)
 *
 * Reset value: 0x00000000U
 *
 * This register configures whether to clear the fields of PDOR.
 */
/*!
 * @name Constants and macros for entire FGPIO_PCOR register
 */
/*@{*/

```

```

#define FGPIO_RD_PCOR(base)      (FGPIO_PCOR_REG(base))
#define FGPIO_WR_PCOR(base, value) (FGPIO_PCOR_REG(base) = (value))
#define FGPIO_RMW_PCOR(base, mask, value) (FGPIO_WR_PCOR(base,
(FGpio_RD_PCOR(base) & ~mask)) | (value)))
/*@}*/

/***** *
 * FGPIO_PTOR - Port Toggle Output Register
 *****/
/*!
 * @brief FGPIO_PTOR - Port Toggle Output Register (WORZ)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire FGPIO_PTOR register
 */
/*@*/
#define FGPIO_RD_PTOR(base)      (FGPIO_PTOR_REG(base))
#define FGPIO_WR_PTOR(base, value) (FGPIO_PTOR_REG(base) = (value))
#define FGPIO_RMW_PTOR(base, mask, value) (FGPIO_WR_PTOR(base,
(FGpio_RD_PTOR(base) & ~mask)) | (value)))
/*@*/

/***** *
 * FGPIO_PDIR - Port Data Input Register
 *****/
/*!
 * @brief FGPIO_PDIR - Port Data Input Register (RO)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire FGPIO_PDIR register
 */
/*@*/
#define FGPIO_RD_PDIR(base)      (FGPIO_PDIR_REG(base))
/*@*/

/***** *
 * FGPIO_PDDR - Port Data Direction Register
 *****/
/*@*/

```

```

/*
 * @brief GPIO_PDDR - Port Data Direction Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * The PDDR configures the individual port pins for input or output.
 */
/*!
 * @name Constants and macros for entire GPIO_PDDR register
 */
/*@{ */
#define GPIO_RD_PDDR(base)      (GPIO_PDDR_REG(base))
#define GPIO_WR_PDDR(base, value) (GPIO_PDDR_REG(base) = (value))
#define GPIO_RMW_PDDR(base, mask, value) (GPIO_WR_PDDR(base,
(GPIO_RD_PDDR(base) & ~mask)) | (value)))
#define GPIO_SET_PDDR(base, value) (GPIO_WR_PDDR(base,
GPIO_RD_PDDR(base) | (value)))
#define GPIO_CLR_PDDR(base, value) (GPIO_WR_PDDR(base,
GPIO_RD_PDDR(base) & ~(value)))
#define GPIO_TOG_PDDR(base, value) (GPIO_WR_PDDR(base,
GPIO_RD_PDDR(base) ^ (value)))
/*@} */

/*
 * MKL25Z4 FTFA
 *
 * Flash Memory Interface
 *
 * Registers defined in this header file:
 * - FTFA_FSTAT - Flash Status Register
 * - FTFA_FCNFG - Flash Configuration Register
 * - FTFA_FSEC - Flash Security Register
 * - FTFA_FOPT - Flash Option Register
 * - FTFA_FCCOB3 - Flash Common Command Object Registers
 * - FTFA_FCCOB2 - Flash Common Command Object Registers
 * - FTFA_FCCOB1 - Flash Common Command Object Registers
 * - FTFA_FCCOB0 - Flash Common Command Object Registers
 * - FTFA_FCCOB7 - Flash Common Command Object Registers
 * - FTFA_FCCOB6 - Flash Common Command Object Registers
 * - FTFA_FCCOB5 - Flash Common Command Object Registers
 * - FTFA_FCCOB4 - Flash Common Command Object Registers
 * - FTFA_FCCOB8 - Flash Common Command Object Registers
 * - FTFA_FCCOB9 - Flash Common Command Object Registers
 * - FTFA_FCCOB2 - Flash Common Command Object Registers
 * - FTFA_FCCOB1 - Flash Common Command Object Registers
 * - FTFA_FCCOB0 - Flash Common Command Object Registers
 * - FTFA_FPROT3 - Program Flash Protection Registers
 * - FTFA_FPROT2 - Program Flash Protection Registers
 * - FTFA_FPROT1 - Program Flash Protection Registers
 * - FTFA_FPROT0 - Program Flash Protection Registers
*/
#define FTFA_INSTANCE_COUNT (1U) /*!< Number of instances of the FTFA
module. */
#define FTFA_IDX (0U) /*!< Instance number for FTFA. */

```

```

/*****
 * FTFA_FSTAT - Flash Status Register
 *****/
/*!
 * @brief FTFA_FSTAT - Flash Status Register (RW)
 *
 * Reset value: 0x00U
 *
 * The FSTAT register reports the operational status of the flash memory
 * module.
 * The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable.
 * MGSTAT0 bit is read only. The unassigned bits read 0 and are not
 * writable. When
 * set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL)
 * bits in
 * this register prevent the launch of any more commands until the flag
 * is
 * cleared (by writing a one to it).
 */
/*!
 * @name Constants and macros for entire FTFA_FSTAT register
 */
/*@{*/
#define FTFA_RD_FSTAT(base)      (FTFA_FSTAT_REG(base))
#define FTFA_WR_FSTAT(base, value) (FTFA_FSTAT_REG(base) = (value))
#define FTFA_RMW_FSTAT(base, mask, value) (FTFA_WR_FSTAT(base,
(FTFA_RD_FSTAT(base) & ~(mask)) | (value)))
#define FTFA_SET_FSTAT(base, value) (BME_OR8(&FTFA_FSTAT_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FSTAT(base, value) (BME_AND8(&FTFA_FSTAT_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FSTAT(base, value) (BME_XOR8(&FTFA_FSTAT_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual FTFA_FSTAT bitfields
 */
/*!
 * @name Register FTFA_FSTAT, field MGSTAT0[0] (RO)
 *
 * The MGSTAT0 status flag is set if an error is detected during
 * execution of a
 * flash command or during the flash reset sequence. As a status flag,
 * this bit
 * cannot (and need not) be cleared by the user like the other error
 * flags in this

```

```

 * register. The value of the MGSTAT0 bit for "command-N" is valid only
at the
 * end of the "command-N" execution when CCIF=1 and before the next
command has
 * been launched. At some point during the execution of "command-N+1,"
the previous
 * result is discarded and any previous error is cleared.
 */
/*@{*/
/*! @brief Read current value of the FTFA_FSTAT_MGSTAT0 field. */
#define FTFA_RD_FSTAT_MGSTAT0(base) ((FTFA_FSTAT_REG(base) &
FTFA_FSTAT_MGSTAT0_MASK) >> FTFA_FSTAT_MGSTAT0_SHIFT)
#define FTFA_BRD_FSTAT_MGSTAT0(base) (BME_UBFX8(&FTFA_FSTAT_REG(base),
FTFA_FSTAT_MGSTAT0_SHIFT, FTFA_FSTAT_MGSTAT0_WIDTH))
/*@}*/

/*
 * @name Register FTFA_FSTAT, field FPVIOL[4] (W1C)
 *
 * The FPVIOL error bit indicates an attempt was made to program or erase
an
 * address in a protected area of program flash memory during a command
write
 * sequence . While FPVIOL is set, the CCIF flag cannot be cleared to
launch a command.
 * The FPVIOL bit is cleared by writing a 1 to it. Writing a 0 to the
FPVIOL bit
 * has no effect.
 *
 * Values:
 * - 0b0 - No protection violation detected
 * - 0b1 - Protection violation detected
 */
/*@{*/
/*! @brief Read current value of the FTFA_FSTAT_FPVIOL field. */
#define FTFA_RD_FSTAT_FPVIOL(base) ((FTFA_FSTAT_REG(base) &
FTFA_FSTAT_FPVIOL_MASK) >> FTFA_FSTAT_FPVIOL_SHIFT)
#define FTFA_BRD_FSTAT_FPVIOL(base) (BME_UBFX8(&FTFA_FSTAT_REG(base),
FTFA_FSTAT_FPVIOL_SHIFT, FTFA_FSTAT_FPVIOL_WIDTH))

/*
 * @brief Set the FPVIOL field to a new value. */
#define FTFA_WR_FSTAT_FPVIOL(base, value) (FTFA_RMW_FSTAT(base,
(FTFA_FSTAT_FPVIOL_MASK | FTFA_FSTAT_ACCERR_MASK |
FTFA_FSTAT_RDCOLERR_MASK | FTFA_FSTAT_CCIF_MASK),
FTFA_FSTAT_FPVIOL(value)))
#define FTFA_BWR_FSTAT_FPVIOL(base, value)
(BME_BFI8(&FTFA_FSTAT_REG(base), ((uint8_t)(value) <<
FTFA_FSTAT_FPVIOL_SHIFT), FTFA_FSTAT_FPVIOL_SHIFT,
FTFA_FSTAT_FPVIOL_WIDTH))
/*@}*/

/*
 * @name Register FTFA_FSTAT, field ACCERR[5] (W1C)
 *

```

```

 * The ACCERR error bit indicates an illegal access has occurred to a
flash
 * memory resource caused by a violation of the command write sequence or
issuing an
 * illegal flash command. While ACCERR is set, the CCIF flag cannot be
cleared to
 * launch a command. The ACCERR bit is cleared by writing a 1 to it.
Writing a 0
 * to the ACCERR bit has no effect.
*
* Values:
* - 0b0 - No access error detected
* - 0b1 - Access error detected
*/
/*@{*/
/*! @brief Read current value of the FTFA_FSTAT_ACCERR field. */
#define FTFA_RD_FSTAT_ACCERR(base) ((FTFA_FSTAT_REG(base) &
FTFA_FSTAT_ACCERR_MASK) >> FTFA_FSTAT_ACCERR_SHIFT)
#define FTFA_BRD_FSTAT_ACCERR(base) (BME_UBFX8(&FTFA_FSTAT_REG(base),
FTFA_FSTAT_ACCERR_SHIFT, FTFA_FSTAT_ACCERR_WIDTH))

/*! @brief Set the ACCERR field to a new value. */
#define FTFA_WR_FSTAT_ACCERR(base, value) (FTFA_RMW_FSTAT(base,
(FTFA_FSTAT_ACCERR_MASK | FTFA_FSTAT_FPVOL_MASK |
FTFA_FSTAT_RDCOLERR_MASK | FTFA_FSTAT_CCIF_MASK),
FTFA_FSTAT_ACCERR(value)))
#define FTFA_BWR_FSTAT_ACCERR(base, value)
(BME_BFI8(&FTFA_FSTAT_REG(base), ((uint8_t)(value) <<
FTFA_FSTAT_ACCERR_SHIFT), FTFA_FSTAT_ACCERR_SHIFT,
FTFA_FSTAT_ACCERR_WIDTH))
/*}@*/ */

/*!
 * @name Register FTFA_FSTAT, field RDCOLERR[6] (W1C)
 *
 * The RDCOLERR error bit indicates that the MCU attempted a read from a
flash
 * memory resource that was being manipulated by a flash command
(CCIF=0). Any
 * simultaneous access is detected as a collision error by the block
arbitration
 * logic. The read data in this case cannot be guaranteed. The RDCOLERR
bit is
 * cleared by writing a 1 to it. Writing a 0 to RDCOLERR has no effect.
*
* Values:
* - 0b0 - No collision error detected
* - 0b1 - Collision error detected
*/
/*@{*/
/*! @brief Read current value of the FTFA_FSTAT_RDCOLERR field. */
#define FTFA_RD_FSTAT_RDCOLERR(base) ((FTFA_FSTAT_REG(base) &
FTFA_FSTAT_RDCOLERR_MASK) >> FTFA_FSTAT_RDCOLERR_SHIFT)

```

```

#define FTFA_BRD_FSTAT_RDCOLERR(base) (BME_UBFX8(&FTFA_FSTAT_REG(base),  

FTFA_FSTAT_RDCOLERR_SHIFT, FTFA_FSTAT_RDCOLERR_WIDTH))

/*! @brief Set the RDCOLERR field to a new value. */  

#define FTFA_WR_FSTAT_RDCOLERR(base, value) (FTFA_RMW_FSTAT(base,  

(FTFA_FSTAT_RDCOLERR_MASK | FTFA_FSTAT_FPVIOL_MASK |  

FTFA_FSTAT_ACCERR_MASK | FTFA_FSTAT_CCIF_MASK),  

FTFA_FSTAT_RDCOLERR(value)))  

#define FTFA_BWR_FSTAT_RDCOLERR(base, value)  

(BME_BFI8(&FTFA_FSTAT_REG(base), ((uint8_t)(value) <<  

FTFA_FSTAT_RDCOLERR_SHIFT), FTFA_FSTAT_RDCOLERR_SHIFT,  

FTFA_FSTAT_RDCOLERR_WIDTH))  

/*@*/
```

/\*!

- \* @name Register FTFA\_FSTAT, field CCIF[7] (W1C)
- \*
- \* The CCIF flag indicates that a flash command has completed. The CCIF flag is
- \* cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until
- \* command completion or command violation. The CCIF bit is reset to 0 but is set
- \* to 1 by the memory controller at the end of the reset initialization sequence.
- \* Depending on how quickly the read occurs after reset release, the user may or
- \* may not see the 0 hardware reset value.
- \*
- \* Values:
- \* - 0b0 - Flash command in progress
- \* - 0b1 - Flash command has completed
- \*/

```
/*@*/
/*! @brief Read current value of the FTFA_FSTAT_CCIF field. */  

#define FTFA_RD_FSTAT_CCIF(base) ((FTFA_FSTAT_REG(base) &  

FTFA_FSTAT_CCIF_MASK) >> FTFA_FSTAT_CCIF_SHIFT)  

#define FTFA_BRD_FSTAT_CCIF(base) (BME_UBFX8(&FTFA_FSTAT_REG(base),  

FTFA_FSTAT_CCIF_SHIFT, FTFA_FSTAT_CCIF_WIDTH))

/*! @brief Set the CCIF field to a new value. */  

#define FTFA_WR_FSTAT_CCIF(base, value) (FTFA_RMW_FSTAT(base,  

(FTFA_FSTAT_CCIF_MASK | FTFA_FSTAT_FPVIOL_MASK | FTFA_FSTAT_ACCERR_MASK |  

FTFA_FSTAT_RDCOLERR_MASK), FTFA_FSTAT_CCIF(value)))  

#define FTFA_BWR_FSTAT_CCIF(base, value) (BME_BFI8(&FTFA_FSTAT_REG(base),  

((uint8_t)(value) << FTFA_FSTAT_CCIF_SHIFT), FTFA_FSTAT_CCIF_SHIFT,  

FTFA_FSTAT_CCIF_WIDTH))  

/*@*/
```

---

```
*****
*****  

* FTFA_FCNFG - Flash Configuration Register
```

```
*****
**** */

/*!
 * @brief FTFA_FCNFG - Flash Configuration Register (RW)
 *
 * Reset value: 0x00U
 *
 * This register provides information on the current functional state of
 * the
 * flash memory module. The erase control bits (ERSAREQ and ERSSUSP) have
 * write
 * restrictions. The unassigned bits read as noted and are not writable.
 */
/*!
 * @name Constants and macros for entire FTFA_FCNFG register
 */
/*@{ */
#define FTFA_RD_FCNFG(base)          (FTFA_FCNFG_REG(base))
#define FTFA_WR_FCNFG(base, value)   (FTFA_FCNFG_REG(base) = (value))
#define FTFA_RMW_FCNFG(base, mask, value) (FTFA_WR_FCNFG(base,
(FTFA_RD_FCNFG(base) & ~mask)) | (value)))
#define FTFA_SET_FCNFG(base, value)  (BME_OR8(&FTFA_FCNFG_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCNFG(base, value)  (BME_AND8(&FTFA_FCNFG_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCNFG(base, value)  (BME_XOR8(&FTFA_FCNFG_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual FTFA_FCNFG bitfields
 */

/*!
 * @name Register FTFA_FCNFG, field ERSSUSP[4] (RW)
 *
 * The ERSSUSP bit allows the user to suspend (interrupt) the Erase Flash
 * Sector
 * command while it is executing.
 *
 * Values:
 * - 0b0 - No suspend requested
 * - 0b1 - Suspend the current Erase Flash Sector command execution.
 */
/*@{ */
/*! @brief Read current value of the FTFA_FCNFG_ERSSUSP field. */
#define FTFA_RD_FCNFG_ERSSUSP(base) ((FTFA_FCNFG_REG(base) &
FTFA_FCNFG_ERSSUSP_MASK) >> FTFA_FCNFG_ERSSUSP_SHIFT)
#define FTFA_BRD_FCNFG_ERSSUSP(base) (BME_UBX8(&FTFA_FCNFG_REG(base),
FTFA_FCNFG_ERSSUSP_SHIFT, FTFA_FCNFG_ERSSUSP_WIDTH))

/*! @brief Set the ERSSUSP field to a new value. */

```

```

#define FTFA_WR_FCNFG_ERSSUSP(base, value) (FTFA_RMW_FCNFG(base,
FTFA_FCNFG_ERSSUSP_MASK, FTFA_FCNFG_ERSSUSP(value)))
#define FTFA_BWR_FCNFG_ERSSUSP(base, value)
(BME_BFI8(&FTFA_FCNFG_REG(base), ((uint8_t)(value) <<
FTFA_FCNFG_ERSSUSP_SHIFT), FTFA_FCNFG_ERSSUSP_SHIFT,
FTFA_FCNFG_ERSSUSP_WIDTH))
/*@}*/

/*!!
 * @name Register FTFA_FCNFG, field ERSAREQ[5] (RO)
 *
 * This bit issues a request to the memory controller to execute the
Erase All
 * Blocks command and release security. ERSAREQ is not directly writable
but is
 * under indirect user control. Refer to the device's Chip Configuration
details on
 * how to request this command. The ERSAREQ bit sets when an erase all
request
 * is triggered external to the flash memory module and CCIF is set (no
command is
 * currently being executed). ERSAREQ is cleared by the flash memory
module when
 * the operation completes.
 *
 * Values:
 * - 0b0 - No request or request complete
 * - 0b1 - Request to: run the Erase All Blocks command, verify the
erased
 * state, program the security byte in the Flash Configuration Field
to the
 * unsecure state, and release MCU security by setting the FSEC[SEC]
field to the
 * unsecure state.
 */
/*@{*/
/*! @brief Read current value of the FTFA_FCNFG_ERSAREQ field. */
#define FTFA_RD_FCNFG_ERSAREQ(base) ((FTFA_FCNFG_REG(base) &
FTFA_FCNFG_ERSAREQ_MASK) >> FTFA_FCNFG_ERSAREQ_SHIFT)
#define FTFA_BRD_FCNFG_ERSAREQ(base) (BME_UBFX8(&FTFA_FCNFG_REG(base),
FTFA_FCNFG_ERSAREQ_SHIFT, FTFA_FCNFG_ERSAREQ_WIDTH))
/*@*/ */

/*!!
 * @name Register FTFA_FCNFG, field RDCOLLIE[6] (RW)
 *
 * The RDCOLLIE bit controls interrupt generation when a flash memory
read
 * collision error occurs.
 *
 * Values:
 * - 0b0 - Read collision error interrupt disabled
 * - 0b1 - Read collision error interrupt enabled. An interrupt request
is

```

```

*      generated whenever a flash memory read collision error is detected
(see the
*      description of FSTAT[RDCOLERR]) .
*/
/*@{ */
/*! @brief Read current value of the FTFA_FCNFG_RDCOLLIE field. */
#define FTFA_RD_FCNFG_RDCOLLIE(base) ((FTFA_FCNFG_REG(base) &
FTFA_FCNFG_RDCOLLIE_MASK) >> FTFA_FCNFG_RDCOLLIE_SHIFT)
#define FTFA_BRD_FCNFG_RDCOLLIE(base) (BME_UBFX8(&FTFA_FCNFG_REG(base),
FTFA_FCNFG_RDCOLLIE_SHIFT, FTFA_FCNFG_RDCOLLIE_WIDTH))

/*! @brief Set the RDCOLLIE field to a new value. */
#define FTFA_WR_FCNFG_RDCOLLIE(base, value) (FTFA_RMW_FCNFG(base,
FTFA_FCNFG_RDCOLLIE_MASK, FTFA_FCNFG_RDCOLLIE(value)))
#define FTFA_BWR_FCNFG_RDCOLLIE(base, value)
(BME_BFI8(&FTFA_FCNFG_REG(base), ((uint8_t)(value) <<
FTFA_FCNFG_RDCOLLIE_SHIFT), FTFA_FCNFG_RDCOLLIE_SHIFT,
FTFA_FCNFG_RDCOLLIE_WIDTH))
/*@} */

/*!
* @name Register FTFA_FCNFG, field CCIE[7] (RW)
*
* The CCIE bit controls interrupt generation when a flash command
completes.
*
* Values:
* - 0b0 - Command complete interrupt disabled
* - 0b1 - Command complete interrupt enabled. An interrupt request is
generated
*      whenever the FSTAT[CCIF] flag is set.
*/
/*@{ */
/*! @brief Read current value of the FTFA_FCNFG_CCIE field. */
#define FTFA_RD_FCNFG_CCIE(base) ((FTFA_FCNFG_REG(base) &
FTFA_FCNFG_CCIE_MASK) >> FTFA_FCNFG_CCIE_SHIFT)
#define FTFA_BRD_FCNFG_CCIE(base) (BME_UBFX8(&FTFA_FCNFG_REG(base),
FTFA_FCNFG_CCIE_SHIFT, FTFA_FCNFG_CCIE_WIDTH))

/*! @brief Set the CCIE field to a new value. */
#define FTFA_WR_FCNFG_CCIE(base, value) (FTFA_RMW_FCNFG(base,
FTFA_FCNFG_CCIE_MASK, FTFA_FCNFG_CCIE(value)))
#define FTFA_BWR_FCNFG_CCIE(base, value) (BME_BFI8(&FTFA_FCNFG_REG(base),
((uint8_t)(value) << FTFA_FCNFG_CCIE_SHIFT), FTFA_FCNFG_CCIE_SHIFT,
FTFA_FCNFG_CCIE_WIDTH))
/*@} */

*****  

*****  

* FTFA_FSEC - Flash Security Register  

*****  

*****  

*****/

```

```

/*
 * @brief FTFA_FSEC - Flash Security Register (RO)
 *
 * Reset value: 0x00U
 *
 * This read-only register holds all bits associated with the security of
the
 * MCU and flash memory module. During the reset sequence, the register
is loaded
 * with the contents of the flash security byte in the Flash
Configuration Field
 * located in program flash memory. The flash basis for the values is
signified by
 * X in the reset value.
 */
/*!
 * @name Constants and macros for entire FTFA_FSEC register
 */
/*@{ */
#define FTFA_RD_FSEC(base)          (FTFA_FSEC_REG(base))
/*}@*/



/*
 * Constants & macros for individual FTFA_FSEC bitfields
 */

/*!
 * @name Register FTFA_FSEC, field SEC[1:0] (RO)
 *
 * These bits define the security state of the MCU. In the secure state,
the MCU
 * limits access to flash memory module resources. The limitations are
defined
 * per device and are detailed in the Chip Configuration details. If the
flash
 * memory module is unsecured using backdoor key access, the SEC bits are
forced to
 * 10b.
 *
 * Values:
 * - 0b00 - MCU security status is secure
 * - 0b01 - MCU security status is secure
 * - 0b10 - MCU security status is unsecure (The standard shipping
condition of
 *           the flash memory module is unsecure.)
 * - 0b11 - MCU security status is secure
 */
/*@{ */
/*! @brief Read current value of the FTFA_FSEC_SEC field. */
#define FTFA_RD_FSEC_SEC(base) ((FTFA_FSEC_REG(base) &
FTFA_FSEC_SEC_MASK) >> FTFA_FSEC_SEC_SHIFT)
#define FTFA_BRD_FSEC_SEC(base) (BME_UBFX8(&FTFA_FSEC_REG(base),
FTFA_FSEC_SEC_SHIFT, FTFA_FSEC_SEC_WIDTH))
/*}@*/

```

```

/*!!
 * @name Register FTFA_FSEC, field FSLACC[3:2] (RO)
 *
 * These bits enable or disable access to the flash memory contents
during
 * returned part failure analysis at Freescale. When SEC is secure and
FSLACC is
 * denied, access to the program flash contents is denied and any failure
analysis
 * performed by Freescale factory test must begin with a full erase to
unsecure the
 * part. When access is granted (SEC is unsecure, or SEC is secure and
FSLACC is
 * granted), Freescale factory testing has visibility of the current
flash
 * contents. The state of the FSLACC bits is only relevant when the SEC
bits are set to
 * secure. When the SEC field is set to unsecure, the FSLACC setting does
not
 * matter.
 *
 * Values:
 * - 0b00 - Freescale factory access granted
 * - 0b01 - Freescale factory access denied
 * - 0b10 - Freescale factory access denied
 * - 0b11 - Freescale factory access granted
 */
/*@{*/
/*! @brief Read current value of the FTFA_FSEC_FSLACC field. */
#define FTFA_RD_FSEC_FSLACC(base) ((FTFA_FSEC_REG(base) &
FTFA_FSEC_FSLACC_MASK) >> FTFA_FSEC_FSLACC_SHIFT)
#define FTFA_BRD_FSEC_FSLACC(base) (BME_UBX8(&FTFA_FSEC_REG(base),
FTFA_FSEC_FSLACC_SHIFT, FTFA_FSEC_FSLACC_WIDTH))
/*}@*/ */

/*!!
 * @name Register FTFA_FSEC, field MEEN[5:4] (RO)
 *
 * Enables and disables mass erase capability of the flash memory module.
The
 * state of the MEEN bits is only relevant when the SEC bits are set to
secure
 * outside of NVM Normal Mode. When the SEC field is set to unsecure, the
MEEN
 * setting does not matter.
 *
 * Values:
 * - 0b00 - Mass erase is enabled
 * - 0b01 - Mass erase is enabled
 * - 0b10 - Mass erase is disabled
 * - 0b11 - Mass erase is enabled
 */
/*@{*/

```

```

/*! @brief Read current value of the FTFA_FSEC_MEEN field. */
#define FTFA_RD_FSEC_MEEN(base) ((FTFA_FSEC_REG(base) &
FTFA_FSEC_MEEN_MASK) >> FTFA_FSEC_MEEN_SHIFT)
#define FTFA_BRD_FSEC_MEEN(base) (BME_UBFX8(&FTFA_FSEC_REG(base),
FTFA_FSEC_MEEN_SHIFT, FTFA_FSEC_MEEN_WIDTH))
/*@}*/

/*!!
 * @name Register FTFA_FSEC, field KEYEN[7:6] (RO)
 *
 * These bits enable and disable backdoor key access to the flash memory
module.
 *
 * Values:
 * - 0b00 - Backdoor key access disabled
 * - 0b01 - Backdoor key access disabled (preferred KEYEN state to
enable
 *          backdoor key access)
 * - 0b10 - Backdoor key access enabled
 * - 0b11 - Backdoor key access disabled
 */
/*@*/
/*! @brief Read current value of the FTFA_FSEC_KEYEN field. */
#define FTFA_RD_FSEC_KEYEN(base) ((FTFA_FSEC_REG(base) &
FTFA_FSEC_KEYEN_MASK) >> FTFA_FSEC_KEYEN_SHIFT)
#define FTFA_BRD_FSEC_KEYEN(base) (BME_UBFX8(&FTFA_FSEC_REG(base),
FTFA_FSEC_KEYEN_SHIFT, FTFA_FSEC_KEYEN_WIDTH))
/*@*/

/***** *****
 * FTFA_FOPT - Flash Option Register
***** */

/*!!
 * @brief FTFA_FOPT - Flash Option Register (RO)
 *
 * Reset value: 0x00U
 *
 * The flash option register allows the MCU to customize its operations
by
 * examining the state of these read-only bits, which are loaded from NVM
at reset.
 * The function of the bits is defined in the device's Chip Configuration
details.
 * All bits in the register are read-only . During the reset sequence,
the
 * register is loaded from the flash nonvolatile option byte in the Flash
Configuration
 * Field located in program flash memory. The flash basis for the values
is
 * signified by X in the reset value.

```

```

        */
/*!
 * @name Constants and macros for entire FTFA_FOPT register
 */
/*@{ */
#define FTFA_RD_FOPT(base)          (FTFA_FOPT_REG(base))
/*}@*/



/********************* Flash Common Command Object Registers *****

 * FTFA_FCCOB3 - Flash Common Command Object Registers

*****/


/*!!
 * @brief FTFA_FCCOB3 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOB8.
 */
/*!
 * @name Constants and macros for entire FTFA_FCCOB3 register
 */
/*@{ */

#define FTFA_RD_FCCOB3(base)          (FTFA_FCCOB3_REG(base))
#define FTFA_WR_FCCOB3(base, value)   (FTFA_FCCOB3_REG(base) = (value))
#define FTFA_RMW_FCCOB3(base, mask, value) (FTFA_WR_FCCOB3(base,
(FTFA_RD_FCCOB3(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB3(base, value)   (BME_OR8(&FTFA_FCCOB3_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB3(base, value)   (BME_AND8(&FTFA_FCCOB3_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOB3(base, value)   (BME_XOR8(&FTFA_FCCOB3_REG(base),
(uint8_t)(value)))
/*}@*/



/********************* Flash Common Command Object Registers *****

 * FTFA_FCCOB2 - Flash Common Command Object Registers

*****/


/*!!
 * @brief FTFA_FCCOB2 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 */

```

```

 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/**!
 * @name Constants and macros for entire FTFA_FCCOB2 register
 */
/*@{*/
#define FTFA_RD_FCCOB2(base)      (FTFA_FCCOB2_REG(base))
#define FTFA_WR_FCCOB2(base, value) (FTFA_FCCOB2_REG(base) = (value))
#define FTFA_RMW_FCCOB2(base, mask, value) (FTFA_WR_FCCOB2(base,
(FTFA_RD_FCCOB2(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB2(base, value) (BME_OR8(&FTFA_FCCOB2_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB2(base, value) (BME_AND8(&FTFA_FCCOB2_REG(base),
(uint8_t)(~value)))
#define FTFA_TOG_FCCOB2(base, value) (BME_XOR8(&FTFA_FCCOB2_REG(base),
(uint8_t)(value)))
/*@}*/
/******
 * FTFA_FCCOB1 - Flash Common Command Object Registers
*****/
/*!
 * @brief FTFA_FCCOB1 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/**!
 * @name Constants and macros for entire FTFA_FCCOB1 register
 */
/*@{*/
#define FTFA_RD_FCCOB1(base)      (FTFA_FCCOB1_REG(base))
#define FTFA_WR_FCCOB1(base, value) (FTFA_FCCOB1_REG(base) = (value))
#define FTFA_RMW_FCCOB1(base, mask, value) (FTFA_WR_FCCOB1(base,
(FTFA_RD_FCCOB1(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB1(base, value) (BME_OR8(&FTFA_FCCOB1_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB1(base, value) (BME_AND8(&FTFA_FCCOB1_REG(base),
(uint8_t)(~value)))
#define FTFA_TOG_FCCOB1(base, value) (BME_XOR8(&FTFA_FCCOB1_REG(base),
(uint8_t)(value)))

```

```

/*@}*/

/***** 
 * FTFA_FCCOB0 - Flash Common Command Object Registers
***** 

/*!!
 * @brief FTFA_FCCOB0 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/*!
 * @name Constants and macros for entire FTFA_FCCOB0 register
 */
/*{@{*/
#define FTFA_RD_FCCOB0(base)      (FTFA_FCCOB0_REG(base))
#define FTFA_WR_FCCOB0(base, value) (FTFA_FCCOB0_REG(base) = (value))
#define FTFA_RMW_FCCOB0(base, mask, value) (FTFA_WR_FCCOB0(base,
(FTFA_RD_FCCOB0(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB0(base, value) (BME_OR8(&FTFA_FCCOB0_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB0(base, value) (BME_AND8(&FTFA_FCCOB0_REG(base),
(uint8_t)(~value)))
#define FTFA_TOG_FCCOB0(base, value) (BME_XOR8(&FTFA_FCCOB0_REG(base),
(uint8_t)(value)))
/*@}*/

/***** 
 * FTFA_FCCOB7 - Flash Common Command Object Registers
***** 

/*!!
 * @brief FTFA_FCCOB7 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */

```

```

/*!
 * @name Constants and macros for entire FTFA_FCCOB7 register
 */
/*@{*/
#define FTFA_RD_FCCOB7(base)      (FTFA_FCCOB7_REG(base))
#define FTFA_WR_FCCOB7(base, value) (FTFA_FCCOB7_REG(base) = (value))
#define FTFA_RMW_FCCOB7(base, mask, value) (FTFA_WR_FCCOB7(base,
(FTFA_RD_FCCOB7(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB7(base, value) (BME_OR8(&FTFA_FCCOB7_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB7(base, value) (BME_AND8(&FTFA_FCCOB7_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOB7(base, value) (BME_XOR8(&FTFA_FCCOB7_REG(base),
(uint8_t)(value)))
/*}@*/
*****  

* FTFA_FCCOB6 - Flash Common Command Object Registers  

*****  

/*!  

 * @brief FTFA_FCCOB6 - Flash Common Command Object Registers (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The FCCOB register group provides 12 bytes for command codes and  

parameters.  

 * The individual bytes within the set append a 0-B hex identifier to the  

FCCOB  

 * register name: FCCOB0, FCCOB1, ..., FCCOB8.  

 */  

/*!  

 * @name Constants and macros for entire FTFA_FCCOB6 register
 */
/*@{*/
#define FTFA_RD_FCCOB6(base)      (FTFA_FCCOB6_REG(base))
#define FTFA_WR_FCCOB6(base, value) (FTFA_FCCOB6_REG(base) = (value))
#define FTFA_RMW_FCCOB6(base, mask, value) (FTFA_WR_FCCOB6(base,
(FTFA_RD_FCCOB6(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB6(base, value) (BME_OR8(&FTFA_FCCOB6_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB6(base, value) (BME_AND8(&FTFA_FCCOB6_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOB6(base, value) (BME_XOR8(&FTFA_FCCOB6_REG(base),
(uint8_t)(value)))
/*}@*/
*****  

* FTFA_FCCOB5 - Flash Common Command Object Registers

```

```
*****
*****/



/*!
 * @brief FTFA_FCCOB5 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/**!
 * @name Constants and macros for entire FTFA_FCCOB5 register
 */
/*@{ */
#define FTFA_RD_FCCOB5(base)      (FTFA_FCCOB5_REG(base))
#define FTFA_WR_FCCOB5(base, value) (FTFA_FCCOB5_REG(base) = (value))
#define FTFA_RMW_FCCOB5(base, mask, value) (FTFA_WR_FCCOB5(base,
(FTFA_RD_FCCOB5(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB5(base, value) (BME_OR8(&FTFA_FCCOB5_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB5(base, value) (BME_AND8(&FTFA_FCCOB5_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOB5(base, value) (BME_XOR8(&FTFA_FCCOB5_REG(base),
(uint8_t)(value)))
/*@} */

/*****
*****
```

\* FTFA\_FCCOB4 - Flash Common Command Object Registers

```
*****
*****/



/*!
 * @brief FTFA_FCCOB4 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/**!
 * @name Constants and macros for entire FTFA_FCCOB4 register
 */
/*@{ */
#define FTFA_RD_FCCOB4(base)      (FTFA_FCCOB4_REG(base))
```

```

#define FTFA_WR_FCCOB4(base, value)  (FTFA_FCCOB4_REG(base) = (value))
#define FTFA_RMW_FCCOB4(base, mask, value)  (FTFA_WR_FCCOB4(base,
(FTFA_RD_FCCOB4(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB4(base, value)  (BME_OR8(&FTFA_FCCOB4_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOB4(base, value)  (BME_AND8(&FTFA_FCCOB4_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOB4(base, value)  (BME_XOR8(&FTFA_FCCOB4_REG(base),
(uint8_t)(value)))
/*@}*/

/*********************  

*****  

* FTFA_FCCOBB - Flash Common Command Object Registers  

*****  

*****/  

  

/*!  

* @brief FTFA_FCCOBB - Flash Common Command Object Registers (RW)  

*  

* Reset value: 0x00U  

*  

* The FCCOB register group provides 12 bytes for command codes and  

parameters.  

* The individual bytes within the set append a 0-B hex identifier to the  

FCCOB  

* register name: FCCOB0, FCCOB1, ..., FCCOBB.  

*/  

/**!  

* @name Constants and macros for entire FTFA_FCCOBB register  

*/  

/*@*/  

#define FTFA_RD_FCCOBB(base)      (FTFA_FCCOBB_REG(base))  

#define FTFA_WR_FCCOBB(base, value)  (FTFA_FCCOBB_REG(base) = (value))  

#define FTFA_RMW_FCCOBB(base, mask, value)  (FTFA_WR_FCCOBB(base,
(FTFA_RD_FCCOBB(base) & ~mask) | (value)))  

#define FTFA_SET_FCCOBB(base, value)  (BME_OR8(&FTFA_FCCOBB_REG(base),
(uint8_t)(value)))  

#define FTFA_CLR_FCCOBB(base, value)  (BME_AND8(&FTFA_FCCOBB_REG(base),
(uint8_t)(~(value))))  

#define FTFA_TOG_FCCOBB(base, value)  (BME_XOR8(&FTFA_FCCOBB_REG(base),
(uint8_t)(value)))  

/*@*/  

  

/*********************  

*****  

* FTFA_FCCOBA - Flash Common Command Object Registers  

*****  

*****/  

  

/*!  

* @brief FTFA_FCCOBA - Flash Common Command Object Registers (RW)

```

```

/*
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/*!
 * @name Constants and macros for entire FTFA_FCCOBA register
 */
/*@{*/
#define FTFA_RD_FCCOBA(base)      (FTFA_FCCOBA_REG(base))
#define FTFA_WR_FCCOBA(base, value) (FTFA_FCCOBA_REG(base) = (value))
#define FTFA_RMW_FCCOBA(base, mask, value) (FTFA_WR_FCCOBA(base,
(FTFA_RD_FCCOBA(base) & ~mask) | (value)))
#define FTFA_SET_FCCOBA(base, value) (BME_OR8(&FTFA_FCCOBA_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FCCOBA(base, value) (BME_AND8(&FTFA_FCCOBA_REG(base),
(uint8_t)(~(value))))
#define FTFA_TOG_FCCOBA(base, value) (BME_XOR8(&FTFA_FCCOBA_REG(base),
(uint8_t)(value)))
/*@}*/
/******
 * FTFA_FCCOB9 - Flash Common Command Object Registers
*****/
/*!
 * @brief FTFA_FCCOB9 - Flash Common Command Object Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FCCOB register group provides 12 bytes for command codes and
parameters.
 * The individual bytes within the set append a 0-B hex identifier to the
FCCOB
 * register name: FCCOB0, FCCOB1, ..., FCCOBB.
 */
/*!
 * @name Constants and macros for entire FTFA_FCCOB9 register
 */
/*@{*/
#define FTFA_RD_FCCOB9(base)      (FTFA_FCCOB9_REG(base))
#define FTFA_WR_FCCOB9(base, value) (FTFA_FCCOB9_REG(base) = (value))
#define FTFA_RMW_FCCOB9(base, mask, value) (FTFA_WR_FCCOB9(base,
(FTFA_RD_FCCOB9(base) & ~mask) | (value)))
#define FTFA_SET_FCCOB9(base, value) (BME_OR8(&FTFA_FCCOB9_REG(base),
(uint8_t)(value)))

```

```

#define FTFA_CLR_FCCOB9(base, value) (BME_AND8(&FTFA_FCCOB9_REG(base),  

(uint8_t)(~(value))))  

#define FTFA_TOG_FCCOB9(base, value) (BME_XOR8(&FTFA_FCCOB9_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*****  

*****  

* FTFA_FCCOB8 - Flash Common Command Object Registers  

*****  

****/  

  

/*!  

* @brief FTFA_FCCOB8 - Flash Common Command Object Registers (RW)  

*  

* Reset value: 0x00U  

*  

* The FCCOB register group provides 12 bytes for command codes and  

parameters.  

* The individual bytes within the set append a 0-B hex identifier to the  

FCCOB  

* register name: FCCOB0, FCCOB1, ..., FCCOBB.  

*/  

/*!  

* @name Constants and macros for entire FTFA_FCCOB8 register  

*/  

/*@{*/  

#define FTFA_RD_FCCOB8(base) (FTFA_FCCOB8_REG(base))  

#define FTFA_WR_FCCOB8(base, value) (FTFA_FCCOB8_REG(base) = (value))  

#define FTFA_RMW_FCCOB8(base, mask, value) (FTFA_WR_FCCOB8(base,  

(FTFA_RD_FCCOB8(base) & ~mask) | (value)))  

#define FTFA_SET_FCCOB8(base, value) (BME_OR8(&FTFA_FCCOB8_REG(base),  

(uint8_t)(value)))  

#define FTFA_CLR_FCCOB8(base, value) (BME_AND8(&FTFA_FCCOB8_REG(base),  

(uint8_t)(~(value))))  

#define FTFA_TOG_FCCOB8(base, value) (BME_XOR8(&FTFA_FCCOB8_REG(base),  

(uint8_t)(value)))  

/*@}/

/*****  

*****  

* FTFA_FPROT3 - Program Flash Protection Registers  

*****  

****/  

  

/*!  

* @brief FTFA_FPROT3 - Program Flash Protection Registers (RW)  

*  

* Reset value: 0x00U  

*  

* The FPROT registers define which logical program flash regions are  

protected

```

```

* from program and erase operations. Protected flash regions cannot have
their
* content changed; that is, these regions cannot be programmed and
cannot be
* erased by any flash command. Unprotected regions can be changed by
program and
* erase operations. The four FPROT registers allow up to 32 protectable
regions.
* Each bit protects a 1/32 region of the program flash memory except for
memory
* configurations with less than 32 Kbytes of program flash where each
assigned bit
* protects 1 Kbyte . For configurations with 24 Kbytes of program flash
memory
* or less, FPROT0 is not used. For configurations with 16 Kbytes of
program
* flash memory or less, FPROT1 is not used. For configurations with 8
Kbytes of
* program flash memory, FPROT2 is not used. The bitfields are defined in
each
* register as follows: Program flash protection register Program flash
protection bits
* FPROT0 PROT[31:24] FPROT1 PROT[23:16] FPROT2 PROT[15:8] FPROT3
PROT[7:0]
* During the reset sequence, the FPROT registers are loaded with the
contents of the
* program flash protection bytes in the Flash Configuration Field as
indicated
* in the following table. Program flash protection register Flash
Configuration
* Field offset address FPROT0 0x000B FPROT1 0x000A FPROT2 0x0009 FPROT3
0x0008
* To change the program flash protection that is loaded during the reset
* sequence, unprotect the sector of program flash memory that contains
the Flash
* Configuration Field. Then, reprogram the program flash protection
byte.
*/
/*!
* @name Constants and macros for entire FTFA_FPROT3 register
*/
/*@{*/
#define FTFA_RD_FPROT3(base)      (FTFA_FPROT3_REG(base))
#define FTFA_WR_FPROT3(base, value) (FTFA_FPROT3_REG(base) = (value))
#define FTFA_RMW_FPROT3(base, mask, value) (FTFA_WR_FPROT3(base,
(FTFA_RD_FPROT3(base) & ~mask) | value)))
#define FTFA_SET_FPROT3(base, value) (BME_OR8(&FTFA_FPROT3_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FPROT3(base, value) (BME_AND8(&FTFA_FPROT3_REG(base),
(uint8_t)(~value))))
#define FTFA_TOG_FPROT3(base, value) (BME_XOR8(&FTFA_FPROT3_REG(base),
(uint8_t)(value)))
/*@}*/

```

```

*****
 * FTFA_FPROT2 - Program Flash Protection Registers
*****
**** */

/*!
 * @brief FTFA_FPROT2 - Program Flash Protection Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FPROT registers define which logical program flash regions are
protected
 * from program and erase operations. Protected flash regions cannot have
their
 * content changed; that is, these regions cannot be programmed and
cannot be
 * erased by any flash command. Unprotected regions can be changed by
program and
 * erase operations. The four FPROT registers allow up to 32 protectable
regions.
 * Each bit protects a 1/32 region of the program flash memory except for
memory
 * configurations with less than 32 Kbytes of program flash where each
assigned bit
 * protects 1 Kbyte . For configurations with 24 Kbytes of program flash
memory
 * or less, FPROT0 is not used. For configurations with 16 Kbytes of
program
 * flash memory or less, FPROT1 is not used. For configurations with 8
Kbytes of
 * program flash memory, FPROT2 is not used. The bitfields are defined in
each
 * register as follows: Program flash protection register Program flash
protection bits
 * FPROT0 PROT[31:24] FPROT1 PROT[23:16] FPROT2 PROT[15:8] FPROT3
PROT[7:0]
 * During the reset sequence, the FPROT registers are loaded with the
contents of the
 * program flash protection bytes in the Flash Configuration Field as
indicated
 * in the following table. Program flash protection register Flash
Configuration
 * Field offset address FPROT0 0x000B FPROT1 0x000A FPROT2 0x0009 FPROT3
0x0008
 * To change the program flash protection that is loaded during the reset
 * sequence, unprotect the sector of program flash memory that contains
the Flash
 * Configuration Field. Then, reprogram the program flash protection
byte.
 */
/*!
 * @name Constants and macros for entire FTFA_FPROT2 register

```

```

        */
/*@{*/
#define FTFA_RD_FPROT2(base)      (FTFA_FPROT2_REG(base))
#define FTFA_WR_FPROT2(base, value) (FTFA_FPROT2_REG(base) = (value))
#define FTFA_RMW_FPROT2(base, mask, value) (FTFA_WR_FPROT2(base,
(FTFA_RD_FPROT2(base) & ~mask) | value)))
#define FTFA_SET_FPROT2(base, value) (BME_OR8(&FTFA_FPROT2_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FPROT2(base, value) (BME_AND8(&FTFA_FPROT2_REG(base),
(uint8_t)(~value))))
#define FTFA_TOG_FPROT2(base, value) (BME_XOR8(&FTFA_FPROT2_REG(base),
(uint8_t)(value)))
/*}@*/
*****  

* FTFA_FPROT1 - Program Flash Protection Registers  

*****  

/*!  

 * @brief FTFA_FPROT1 - Program Flash Protection Registers (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The FPROT registers define which logical program flash regions are  

protected  

 * from program and erase operations. Protected flash regions cannot have  

their  

 * content changed; that is, these regions cannot be programmed and  

cannot be  

 * erased by any flash command. Unprotected regions can be changed by  

program and  

 * erase operations. The four FPROT registers allow up to 32 protectable  

regions.  

 * Each bit protects a 1/32 region of the program flash memory except for  

memory  

 * configurations with less than 32 Kbytes of program flash where each  

assigned bit  

 * protects 1 Kbyte . For configurations with 24 Kbytes of program flash  

memory  

 * or less, FPROT0 is not used. For configurations with 16 Kbytes of  

program  

 * flash memory or less, FPROT1 is not used. For configurations with 8  

Kbytes of  

 * program flash memory, FPROT2 is not used. The bitfields are defined in  

each  

 * register as follows: Program flash protection register Program flash  

protection bits  

 * FPROT0 PROT[31:24] FPROT1 PROT[23:16] FPROT2 PROT[15:8] FPROT3  

PROT[7:0]  

 * During the reset sequence, the FPROT registers are loaded with the  

contents of the

```

```

 * program flash protection bytes in the Flash Configuration Field as
indicated
 * in the following table. Program flash protection register Flash
Configuration
 * Field offset address FPROT0 0x000B FPROT1 0x000A FPROT2 0x0009 FPROT3
0x0008
 * To change the program flash protection that is loaded during the reset
 * sequence, unprotect the sector of program flash memory that contains
the Flash
 * Configuration Field. Then, reprogram the program flash protection
byte.
 */
/*!
 * @name Constants and macros for entire FTFA_FPROT1 register
 */
/*@{*/
#define FTFA_RD_FPROT1(base)      (FTFA_FPROT1_REG(base))
#define FTFA_WR_FPROT1(base, value) (FTFA_FPROT1_REG(base) = (value))
#define FTFA_RMW_FPROT1(base, mask, value) (FTFA_WR_FPROT1(base,
(FTFA_RD_FPROT1(base) & ~mask) | value)))
#define FTFA_SET_FPROT1(base, value) (BME_OR8(&FTFA_FPROT1_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FPROT1(base, value) (BME_AND8(&FTFA_FPROT1_REG(base),
(uint8_t)(~value)))
#define FTFA_TOG_FPROT1(base, value) (BME_XOR8(&FTFA_FPROT1_REG(base),
(uint8_t)(value)))
/*}@}*/
/*****
*****
 * FTFA_FPROT0 - Program Flash Protection Registers
*****
*/
/*!
 * @brief FTFA_FPROT0 - Program Flash Protection Registers (RW)
 *
 * Reset value: 0x00U
 *
 * The FPROT registers define which logical program flash regions are
protected
 * from program and erase operations. Protected flash regions cannot have
their
 * content changed; that is, these regions cannot be programmed and
cannot be
 * erased by any flash command. Unprotected regions can be changed by
program and
 * erase operations. The four FPROT registers allow up to 32 protectable
regions.
 * Each bit protects a 1/32 region of the program flash memory except for
memory
 * configurations with less than 32 Kbytes of program flash where each
assigned bit

```

```

 * protects 1 Kbyte . For configurations with 24 Kbytes of program flash
memory
 * or less, FPROT0 is not used. For configurations with 16 Kbytes of
program
 * flash memory or less, FPROT1 is not used. For configurations with 8
Kbytes of
 * program flash memory, FPROT2 is not used. The bitfields are defined in
each
 * register as follows: Program flash protection register Program flash
protection bits
 * FPROT0 PROT[31:24] FPROT1 PROT[23:16] FPROT2 PROT[15:8] FPROT3
PROT[7:0]
 * During the reset sequence, the FPROT registers are loaded with the
contents of the
 * program flash protection bytes in the Flash Configuration Field as
indicated
 * in the following table. Program flash protection register Flash
Configuration
 * Field offset address FPROT0 0x000B FPROT1 0x000A FPROT2 0x0009 FPROT3
0x0008
 * To change the program flash protection that is loaded during the reset
 * sequence, unprotect the sector of program flash memory that contains
the Flash
 * Configuration Field. Then, reprogram the program flash protection
byte.
 */
/*!
 * @name Constants and macros for entire FTFA_FPROT0 register
 */
/*@{*/
#define FTFA_RD_FPROT0(base)      (FTFA_FPROT0_REG(base))
#define FTFA_WR_FPROT0(base, value) (FTFA_FPROT0_REG(base) = (value))
#define FTFA_RMW_FPROT0(base, mask, value) (FTFA_WR_FPROT0(base,
(FTFA_RD_FPROT0(base) & ~mask) | (value)))
#define FTFA_SET_FPROT0(base, value) (BME_OR8(&FTFA_FPROT0_REG(base),
(uint8_t)(value)))
#define FTFA_CLR_FPROT0(base, value) (BME_AND8(&FTFA_FPROT0_REG(base),
(uint8_t)(~value)))
#define FTFA_TOG_FPROT0(base, value) (BME_XOR8(&FTFA_FPROT0_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * MKL25Z4 GPIO
 *
 * General Purpose Input/Output
 *
 * Registers defined in this header file:
 * - GPIO_PDOR - Port Data Output Register
 * - GPIO_PSOR - Port Set Output Register
 * - GPIO_PCOR - Port Clear Output Register
 * - GPIO_PTOR - Port Toggle Output Register
 * - GPIO_PDIR - Port Data Input Register
 * - GPIO_PDDR - Port Data Direction Register

```

```

*/



#define GPIO_INSTANCE_COUNT (5U) /*!< Number of instances of the GPIO
module. */
#define GPIOA_IDX (0U) /*!< Instance number for GPIOA. */
#define GPIOB_IDX (1U) /*!< Instance number for GPIOB. */
#define GPIOC_IDX (2U) /*!< Instance number for GPIOC. */
#define GPIOD_IDX (3U) /*!< Instance number for GPIOD. */
#define GPIOE_IDX (4U) /*!< Instance number for GPIOE. */

*****  

****  

 * GPIO_PDOR - Port Data Output Register  

*****  

****/  

  

/*!  

 * @brief GPIO_PDOR - Port Data Output Register (RW)  

 *  

 * Reset value: 0x00000000U  

 *  

 * This register configures the logic levels that are driven on each  

 * general-purpose output pins. Do not modify pin configuration registers  

associated with  

 * pins not available in your selected package. All un-bonded pins not  

available in  

 * your package will default to DISABLE state for lowest power  

consumption.  

 */  

/*!  

 * @name Constants and macros for entire GPIO_PDOR register  

 */  

/*@{*/  

#define GPIO_RD_PDOR(base) (GPIO_PDOR_REG(base))  

#define GPIO_WR_PDOR(base, value) (GPIO_PDOR_REG(base) = (value))  

#define GPIO_RMW_PDOR(base, mask, value) (GPIO_WR_PDOR(base,  

(GPIO_RD_PDOR(base) & ~mask) | (value)))  

#define GPIO_SET_PDOR(base, value) (BME_OR32(&GPIO_PDOR_REG(base),  

(uint32_t)(value)))  

#define GPIO_CLR_PDOR(base, value) (BME_AND32(&GPIO_PDOR_REG(base),  

(uint32_t)(~(value))))  

#define GPIO_TOG_PDOR(base, value) (BME_XOR32(&GPIO_PDOR_REG(base),  

(uint32_t)(value)))  

/*}@*/  

  

*****  

****  

 * GPIO_PSOR - Port Set Output Register  

*****  

****/  

  

/*!

```

```

* @brief GPIO_PSOR - Port Set Output Register (WORZ)
*
* Reset value: 0x00000000U
*
* This register configures whether to set the fields of the PDOR.
*/
/*!
* @name Constants and macros for entire GPIO_PSOR register
*/
/*@{*/
#define GPIO_RD_PSOR(base)      (GPIO_PSOR_REG(base))
#define GPIO_WR_PSOR(base, value) (GPIO_PSOR_REG(base) = (value))
#define GPIO_RMW_PSOR(base, mask, value) (GPIO_WR_PSOR(base,
(GPIO_RD_PSOR(base) & ~mask) | (value)))
/*@}*/
*****  

* GPIO_PCOR - Port Clear Output Register  

*****  

/*!
* @brief GPIO_PCOR - Port Clear Output Register (WORZ)
*
* Reset value: 0x00000000U
*
* This register configures whether to clear the fields of PDOR.
*/
/*!
* @name Constants and macros for entire GPIO_PCOR register
*/
/*@{*/
#define GPIO_RD_PCOR(base)      (GPIO_PCOR_REG(base))
#define GPIO_WR_PCOR(base, value) (GPIO_PCOR_REG(base) = (value))
#define GPIO_RMW_PCOR(base, mask, value) (GPIO_WR_PCOR(base,
(GPIO_RD_PCOR(base) & ~mask) | (value)))
/*@}*/
*****  

* GPIO_PTOR - Port Toggle Output Register  

*****  

/*!
* @brief GPIO_PTOR - Port Toggle Output Register (WORZ)
*
* Reset value: 0x00000000U
*/
/*!
* @name Constants and macros for entire GPIO_PTOR register

```

```

        */
/*@{*/
#define GPIO_RD_PTOR(base)          (GPIO_PTOR_REG(base))
#define GPIO_WR_PTOR(base, value)   (GPIO_PTOR_REG(base) = (value))
#define GPIO_RMW_PTOR(base, mask, value) (GPIO_WR_PTOR(base,
(GPIO_RD_PTOR(base) & ~mask) | (value)))
/*@}*/
/******
 * GPIO_PDIR - Port Data Input Register
*****/
/*!
 * @brief GPIO_PDIR - Port Data Input Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * Do not modify pin configuration registers associated with pins not
available
 * in your selected package. All un-bonded pins not available in your
package
 * will default to DISABLE state for lowest power consumption.
 */
/*!
 * @name Constants and macros for entire GPIO_PDIR register
 */
/*@*/
#define GPIO_RD_PDIR(base)          (GPIO_PDIR_REG(base))
/*@}*/

/******
 * GPIO_PDDR - Port Data Direction Register
*****/
/*!
 * @brief GPIO_PDDR - Port Data Direction Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * The PDDR configures the individual port pins for input or output.
 */
/*!
 * @name Constants and macros for entire GPIO_PDDR register
 */
/*@*/
#define GPIO_RD_PDDR(base)          (GPIO_PDDR_REG(base))
#define GPIO_WR_PDDR(base, value)   (GPIO_PDDR_REG(base) = (value))

```

```

#define GPIO_RMW_PDDR(base, mask, value) (GPIO_WR_PDDR(base, (GPIO_RD_PDDR(base) & ~mask)) | (value)))
#define GPIO_SET_PDDR(base, value) (BME_OR32(&GPIO_PDDR_REG(base), (uint32_t)(value)))
#define GPIO_CLR_PDDR(base, value) (BME_AND32(&GPIO_PDDR_REG(base), (uint32_t)(~(value))))
#define GPIO_TOG_PDDR(base, value) (BME_XOR32(&GPIO_PDDR_REG(base), (uint32_t)(value)))
/*@}*/

/*
 * MKL25Z4 I2C
 *
 * Inter-Integrated Circuit
 *
 * Registers defined in this header file:
 * - I2C_A1 - I2C Address Register 1
 * - I2C_F - I2C Frequency Divider register
 * - I2C_C1 - I2C Control Register 1
 * - I2C_S - I2C Status register
 * - I2C_D - I2C Data I/O register
 * - I2C_C2 - I2C Control Register 2
 * - I2C_FLT - I2C Programmable Input Glitch Filter register
 * - I2C_RA - I2C Range Address register
 * - I2C_SMB - I2C SMBus Control and Status register
 * - I2C_A2 - I2C Address Register 2
 * - I2C_SLTH - I2C SCL Low Timeout Register High
 * - I2C_SLTL - I2C SCL Low Timeout Register Low
*/
#define I2C_INSTANCE_COUNT (2U) /*!< Number of instances of the I2C module. */
#define I2C0_IDX (0U) /*!< Instance number for I2C0. */
#define I2C1_IDX (1U) /*!< Instance number for I2C1. */

*****  

* I2C_A1 - I2C Address Register 1  

*****  

/*!  

 * @brief I2C_A1 - I2C Address Register 1 (RW)  

 * Reset value: 0x00U  

 * This register contains the slave address to be used by the I2C module.  

 */  

/*!  

 * @name Constants and macros for entire I2C_A1 register  

 */  

/*@*/
#define I2C_RD_A1(base) (I2C_A1_REG(base))

```

```

#define I2C_WR_A1(base, value)      (I2C_A1_REG(base) = (value))
#define I2C_RMW_A1(base, mask, value) (I2C_WR_A1(base, (I2C_RD_A1(base) &
~(mask)) | (value)))
#define I2C_SET_A1(base, value)     (BME_OR8(&I2C_A1_REG(base),
(uint8_t)(value)))
#define I2C_CLR_A1(base, value)     (BME_AND8(&I2C_A1_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_A1(base, value)     (BME_XOR8(&I2C_A1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual I2C_A1 bitfields
 */

/*!!
 * @name Register I2C_A1, field AD[7:1] (RW)
 *
 * Contains the primary slave address used by the I2C module when it is
 * addressed as a slave. This field is used in the 7-bit address scheme
and the lower
 * seven bits in the 10-bit address scheme.
 */
/*@{*/
/*! @brief Read current value of the I2C_A1_AD field. */
#define I2C_RD_A1_AD(base)    ((I2C_A1_REG(base) & I2C_A1_AD_MASK) >>
I2C_A1_AD_SHIFT)
#define I2C_BRD_A1_AD(base)   (BME_UBXF8(&I2C_A1_REG(base),
I2C_A1_AD_SHIFT, I2C_A1_AD_WIDTH))

/*! @brief Set the AD field to a new value. */
#define I2C_WR_A1_AD(base, value) (I2C_RMW_A1(base, I2C_A1_AD_MASK,
I2C_A1_AD(value)))
#define I2C_BWR_A1_AD(base, value) (BME_BFI8(&I2C_A1_REG(base),
((uint8_t)(value) << I2C_A1_AD_SHIFT), I2C_A1_AD_SHIFT, I2C_A1_AD_WIDTH))
/*@}*/

*****  

*****  

 * I2C_F - I2C Frequency Divider register  

*****  

*****  

/*!!
 * @brief I2C_F - I2C Frequency Divider register (RW)
 *
 * Reset value: 0x00U
 */
/*!!
 * @name Constants and macros for entire I2C_F register
 */
/*@{*/
#define I2C_RD_F(base)          (I2C_F_REG(base))

```

```

#define I2C_WR_F(base, value)      (I2C_F_REG(base) = (value))
#define I2C_RMW_F(base, mask, value) (I2C_WR_F(base, (I2C_RD_F(base) &
~(mask)) | (value)))
#define I2C_SET_F(base, value)     (BME_OR8(&I2C_F_REG(base),
(uint8_t)(value)))
#define I2C_CLR_F(base, value)     (BME_AND8(&I2C_F_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_F(base, value)     (BME_XOR8(&I2C_F_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual I2C_F bitfields
 */

/*!
 * @name Register I2C_F, field ICR[5:0] (RW)
 *
 * Prescales the bus clock for bit rate selection. This field and the
MULT field
 * determine the I2C baud rate, the SDA hold time, the SCL start hold
time, and
 * the SCL stop hold time. For a list of values corresponding to each ICR
 * setting, see I2C divider and hold values. The SCL divider multiplied
by multiplier
 * factor (mul) determines the I2C baud rate. I2C baud rate = bus speed
(Hz) / (mul *
 * SCL divider) The SDA hold time is the delay from the falling edge of
SCL (I2C
 * clock) to the changing of SDA (I2C data). SDA hold time = bus period
(s) *
 * mul * SDA hold value The SCL start hold time is the delay from the
falling edge
 * of SDA (I2C data) while SCL is high (start condition) to the falling
edge of
 * SCL (I2C clock). SCL start hold time = bus period (s) * mul * SCL
start hold
 * value The SCL stop hold time is the delay from the rising edge of SCL
(I2C
 * clock) to the rising edge of SDA (I2C data) while SCL is high (stop
condition). SCL
 * stop hold time = bus period (s) * mul * SCL stop hold value For
example, if
 * the bus speed is 8 MHz, the following table shows the possible hold
time values
 * with different ICR and MULT selections to achieve an I2C baud rate of
100
 * kbps. MULT ICR Hold times (us) SDA SCL Start SCL Stop 2h 00h 3.500
3.000 5.500 1h
 * 07h 2.500 4.000 5.250 1h 0Bh 2.250 4.000 5.250 0h 14h 2.125 4.250
5.125 0h
 * 18h 1.125 4.750 5.125
 */
/*@{*/

```

```

/*! @brief Read current value of the I2C_F_ICR field. */
#define I2C_RD_F_ICR(base)    ((I2C_F_REG(base) & I2C_F_ICR_MASK) >>
I2C_F_ICR_SHIFT)
#define I2C_BRD_F_ICR(base)   (BME_UBFX8(&I2C_F_REG(base),
I2C_F_ICR_SHIFT, I2C_F_ICR_WIDTH))

/*! @brief Set the ICR field to a new value. */
#define I2C_WR_F_ICR(base, value) (I2C_RMW_F(base, I2C_F_ICR_MASK,
I2C_F_ICR(value)))
#define I2C_BWR_F_ICR(base, value) (BME_BFI8(&I2C_F_REG(base),
((uint8_t)(value) << I2C_F_ICR_SHIFT), I2C_F_ICR_SHIFT, I2C_F_ICR_WIDTH))
/*@}*/

/*!
 * @name Register I2C_F, field MULT[7:6] (RW)
 *
 * The MULT bits define the multiplier factor mul. This factor is used
along
 * with the SCL divider to generate the I2C baud rate.
 *
 * Values:
 * - 0b00 - mul = 1
 * - 0b01 - mul = 2
 * - 0b10 - mul = 4
 * - 0b11 - Reserved
 */
/*@*/
/*! @brief Read current value of the I2C_F_MULT field. */
#define I2C_RD_F_MULT(base)   ((I2C_F_REG(base) & I2C_F_MULT_MASK) >>
I2C_F_MULT_SHIFT)
#define I2C_BRD_F_MULT(base)  (BME_UBFX8(&I2C_F_REG(base),
I2C_F_MULT_SHIFT, I2C_F_MULT_WIDTH))

/*! @brief Set the MULT field to a new value. */
#define I2C_WR_F_MULT(base, value) (I2C_RMW_F(base, I2C_F_MULT_MASK,
I2C_F_MULT(value)))
#define I2C_BWR_F_MULT(base, value) (BME_BFI8(&I2C_F_REG(base),
((uint8_t)(value) << I2C_F_MULT_SHIFT), I2C_F_MULT_SHIFT,
I2C_F_MULT_WIDTH))
/*@*/

*****  

*****  

 * I2C_C1 - I2C Control Register 1  

*****  

*****/  

  

/*!
 * @brief I2C_C1 - I2C Control Register 1 (RW)
 *
 * Reset value: 0x00U
 */
/*!

```

```

 * @name Constants and macros for entire I2C_C1 register
 */
/*@{*/
#define I2C_RD_C1(base)          (I2C_C1_REG(base))
#define I2C_WR_C1(base, value)   (I2C_C1_REG(base) = (value))
#define I2C_RMW_C1(base, mask, value) (I2C_WR_C1(base, (I2C_RD_C1(base) &
~(mask)) | (value)))
#define I2C_SET_C1(base, value)   (BME_OR8(&I2C_C1_REG(base),
(uint8_t)(value)))
#define I2C_CLR_C1(base, value)   (BME_AND8(&I2C_C1_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_C1(base, value)   (BME_XOR8(&I2C_C1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual I2C_C1 bitfields
 */

/*!
 * @name Register I2C_C1, field DMAEN[0] (RW)
 *
 * The DMAEN bit enables or disables the DMA function.
 *
 * Values:
 * - 0b0 - All DMA signalling disabled.
 * - 0b1 - DMA transfer is enabled and the following conditions trigger
the DMA
 *         request: While FACK = 0, a data byte is received, either address
or data
 *         is transmitted. (ACK/NACK automatic) While FACK = 0, the first
byte
 *         received matches the A1 register or is general call address. If
any address
 *         matching occurs, IAAS and TCF are set. If the direction of
transfer is known
 *         from master to slave, then it is not required to check the SRW.
With this
 *         assumption, DMA can also be used in this case. In other cases, if
the master
 *         reads data from the slave, then it is required to rewrite the C1
register
 *         operation. With this assumption, DMA cannot be used. When FACK =
1, an
 *         address or a data byte is transmitted.
 */
/*@{*/
/*! @brief Read current value of the I2C_C1_DMAEN field. */
#define I2C_RD_C1_DMAEN(base) ((I2C_C1_REG(base) & I2C_C1_DMAEN_MASK) >>
I2C_C1_DMAEN_SHIFT)
#define I2C_BRD_C1_DMAEN(base) (BME_UBFX8(&I2C_C1_REG(base),
I2C_C1_DMAEN_SHIFT, I2C_C1_DMAEN_WIDTH))

/*! @brief Set the DMAEN field to a new value. */

```

```

#define I2C_WR_C1_DMAEN(base, value) (I2C_RMW_C1(base, I2C_C1_DMAEN_MASK, I2C_C1_DMAEN(value)))
#define I2C_BWR_C1_DMAEN(base, value) (BME_BFI8(&I2C_C1_REG(base), ((uint8_t)(value) << I2C_C1_DMAEN_SHIFT), I2C_C1_DMAEN_SHIFT, I2C_C1_DMAEN_WIDTH))
/*@}*/

/*
 * @name Register I2C_C1, field WUEN[1] (RW)
 *
 * The I2C module can wake the MCU from low power mode with no peripheral bus
 * running when slave address matching occurs.
 *
 * Values:
 * - 0b0 - Normal operation. No interrupt generated when address matching in low
 *         power mode.
 * - 0b1 - Enables the wakeup function in low power mode.
 */
/*@*/
/*! @brief Read current value of the I2C_C1_WUEN field. */
#define I2C_RD_C1_WUEN(base) ((I2C_C1_REG(base) & I2C_C1_WUEN_MASK) >> I2C_C1_WUEN_SHIFT)
#define I2C_BRD_C1_WUEN(base) (BME_UBFX8(&I2C_C1_REG(base), I2C_C1_WUEN_SHIFT, I2C_C1_WUEN_WIDTH))

/*! @brief Set the WUEN field to a new value. */
#define I2C_WR_C1_WUEN(base, value) (I2C_RMW_C1(base, I2C_C1_WUEN_MASK, I2C_C1_WUEN(value)))
#define I2C_BWR_C1_WUEN(base, value) (BME_BFI8(&I2C_C1_REG(base), ((uint8_t)(value) << I2C_C1_WUEN_SHIFT), I2C_C1_WUEN_SHIFT, I2C_C1_WUEN_WIDTH))
/*@*/

/*
 * @name Register I2C_C1, field RSTA[2] (WORZ)
 *
 * Writing a one to this bit generates a repeated START condition provided it is
 * the current master. This bit will always be read as zero. Attempting a repeat
 * at the wrong time results in loss of arbitration.
 */
/*@*/
/*! @brief Set the RSTA field to a new value. */
#define I2C_WR_C1_RSTA(base, value) (I2C_RMW_C1(base, I2C_C1_RSTA_MASK, I2C_C1_RSTA(value)))
#define I2C_BWR_C1_RSTA(base, value) (BME_BFI8(&I2C_C1_REG(base), ((uint8_t)(value) << I2C_C1_RSTA_SHIFT), I2C_C1_RSTA_SHIFT, I2C_C1_RSTA_WIDTH))
/*@*/

/*

```

```

* @name Register I2C_C1, field TXAK[3] (RW)
*
* Specifies the value driven onto the SDA during data acknowledge cycles
for
* both master and slave receivers. The value of the FACK bit affects
NACK/ACK
* generation. SCL is held low until TXAK is written.
*
* Values:
* - 0b0 - An acknowledge signal is sent to the bus on the following
receiving
*      byte (if FACK is cleared) or the current receiving byte (if FACK
is set).
* - 0b1 - No acknowledge signal is sent to the bus on the following
receiving
*      data byte (if FACK is cleared) or the current receiving data byte
(if FACK
*      is set).
*/
/*@{*/
/*! @brief Read current value of the I2C_C1_TXAK field. */
#define I2C_RD_C1_TXAK(base) ((I2C_C1_REG(base) & I2C_C1_TXAK_MASK) >>
I2C_C1_TXAK_SHIFT)
#define I2C_BRD_C1_TXAK(base) (BME_UBFX8(&I2C_C1_REG(base),
I2C_C1_TXAK_SHIFT, I2C_C1_TXAK_WIDTH))

/*! @brief Set the TXAK field to a new value. */
#define I2C_WR_C1_TXAK(base, value) (I2C_RMW_C1(base, I2C_C1_TXAK_MASK,
I2C_C1_TXAK(value)))
#define I2C_BWR_C1_TXAK(base, value) (BME_BFI8(&I2C_C1_REG(base),
((uint8_t)(value) << I2C_C1_TXAK_SHIFT), I2C_C1_TXAK_SHIFT,
I2C_C1_TXAK_WIDTH))
/*@}*/

/*
* @name Register I2C_C1, field TX[4] (RW)
*
* Selects the direction of master and slave transfers. In master mode
this bit
* must be set according to the type of transfer required. Therefore, for
address
* cycles, this bit is always set. When addressed as a slave this bit
must be
* set by software according to the SRW bit in the status register.
*
* Values:
* - 0b0 - Receive
* - 0b1 - Transmit
*/
/*@{*/
/*! @brief Read current value of the I2C_C1_TX field. */
#define I2C_RD_C1_TX(base) ((I2C_C1_REG(base) & I2C_C1_TX_MASK) >>
I2C_C1_TX_SHIFT)

```

```

#define I2C_BRD_C1_TX(base)  (BME_UBFX8(&I2C_C1_REG(base),  

I2C_C1_TX_SHIFT, I2C_C1_TX_WIDTH))

/*! @brief Set the TX field to a new value. */
#define I2C_WR_C1_TX(base, value) (I2C_RMW_C1(base, I2C_C1_TX_MASK,  

I2C_C1_TX(value)))
#define I2C_BWR_C1_TX(base, value) (BME_BFI8(&I2C_C1_REG(base),  

((uint8_t)(value) << I2C_C1_TX_SHIFT), I2C_C1_TX_SHIFT, I2C_C1_TX_WIDTH))
/*@}*/

/*
 * @name Register I2C_C1, field MST[5] (RW)
 *
 * When the MST bit is changed from a 0 to a 1, a START signal is
generated on
 * the bus and master mode is selected. When this bit changes from a 1 to
a 0, a
 * STOP signal is generated and the mode of operation changes from master
to slave.
 *
 * Values:
 * - 0b0 - Slave mode
 * - 0b1 - Master mode
 */
/*@*/
/*! @brief Read current value of the I2C_C1_MST field. */
#define I2C_RD_C1_MST(base) ((I2C_C1_REG(base) & I2C_C1_MST_MASK) >>
I2C_C1_MST_SHIFT)
#define I2C_BRD_C1_MST(base) (BME_UBFX8(&I2C_C1_REG(base),  

I2C_C1_MST_SHIFT, I2C_C1_MST_WIDTH))

/*! @brief Set the MST field to a new value. */
#define I2C_WR_C1_MST(base, value) (I2C_RMW_C1(base, I2C_C1_MST_MASK,  

I2C_C1_MST(value)))
#define I2C_BWR_C1_MST(base, value) (BME_BFI8(&I2C_C1_REG(base),  

((uint8_t)(value) << I2C_C1_MST_SHIFT), I2C_C1_MST_SHIFT,  

I2C_C1_MST_WIDTH))
/*@*/

/*
 * @name Register I2C_C1, field IICIE[6] (RW)
 *
 * Enables I2C interrupt requests.
 *
 * Values:
 * - 0b0 - Disabled
 * - 0b1 - Enabled
 */
/*@*/
/*! @brief Read current value of the I2C_C1_IICIE field. */
#define I2C_RD_C1_IICIE(base) ((I2C_C1_REG(base) & I2C_C1_IICIE_MASK) >>
I2C_C1_IICIE_SHIFT)
#define I2C_BRD_C1_IICIE(base) (BME_UBFX8(&I2C_C1_REG(base),  

I2C_C1_IICIE_SHIFT, I2C_C1_IICIE_WIDTH))

```

```

/*! @brief Set the IICIE field to a new value. */
#define I2C_WR_C1_IICIE(base, value) (I2C_RMW_C1(base, I2C_C1_IICIE_MASK,
I2C_C1_IICIE(value)))
#define I2C_BWR_C1_IICIE(base, value) (BME_BFI8(&I2C_C1_REG(base),
((uint8_t)(value) << I2C_C1_IICIE_SHIFT), I2C_C1_IICIE_SHIFT,
I2C_C1_IICIE_WIDTH))
/*@}*/

/*!!
 * @name Register I2C_C1, field IICEN[7] (RW)
 *
 * Enables I2C module operation.
 *
 * Values:
 * - 0b0 - Disabled
 * - 0b1 - Enabled
 */
/*@{*/
/*! @brief Read current value of the I2C_C1_IICEN field. */
#define I2C_RD_C1_IICEN(base) ((I2C_C1_REG(base) & I2C_C1_IICEN_MASK) >>
I2C_C1_IICEN_SHIFT)
#define I2C_BRD_C1_IICEN(base) (BME_UBFX8(&I2C_C1_REG(base),
I2C_C1_IICEN_SHIFT, I2C_C1_IICEN_WIDTH))

/*! @brief Set the IICEN field to a new value. */
#define I2C_WR_C1_IICEN(base, value) (I2C_RMW_C1(base, I2C_C1_IICEN_MASK,
I2C_C1_IICEN(value)))
#define I2C_BWR_C1_IICEN(base, value) (BME_BFI8(&I2C_C1_REG(base),
((uint8_t)(value) << I2C_C1_IICEN_SHIFT), I2C_C1_IICEN_SHIFT,
I2C_C1_IICEN_WIDTH))
/*@}*/

*****  

* I2C_S - I2C Status register  

*****  

/*!  

 * @brief I2C_S - I2C Status register (RW)  

 *  

 * Reset value: 0x80U  

 */
/*!  

 * @name Constants and macros for entire I2C_S register  

 */
/*@{*/
#define I2C_RD_S(base) (I2C_S_REG(base))
#define I2C_WR_S(base, value) (I2C_S_REG(base) = (value))
#define I2C_RMW_S(base, mask, value) (I2C_WR_S(base, (I2C_RD_S(base) &
~(mask)) | (value)))

```

```

#define I2C_SET_S(base, value)      (BME_OR8(&I2C_S_REG(base),  

(uint8_t)(value)))  

#define I2C_CLR_S(base, value)      (BME_AND8(&I2C_S_REG(base),  

(uint8_t)(~(value))))  

#define I2C_TOG_S(base, value)      (BME_XOR8(&I2C_S_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual I2C_S bitfields
 */

/*!!
 * @name Register I2C_S, field RXAK[0] (RO)
 *
 * Values:
 * - 0b0 - Acknowledge signal was received after the completion of one
byte of
 *         data transmission on the bus
 * - 0b1 - No acknowledge signal detected
 */
/*@{*/
/*! @brief Read current value of the I2C_S_RXAK field. */
#define I2C_RD_S_RXAK(base) ((I2C_S_REG(base) & I2C_S_RXAK_MASK) >>  

I2C_S_RXAK_SHIFT)  

#define I2C_BRD_S_RXAK(base) (BME_UBFX8(&I2C_S_REG(base),  

I2C_S_RXAK_SHIFT, I2C_S_RXAK_WIDTH))  

/*@}*/

/*!!
 * @name Register I2C_S, field IICIF[1] (W1C)
 *
 * This bit sets when an interrupt is pending. This bit must be cleared
by
 * software by writing a 1 to it, such as in the interrupt routine. One
of the
 * following events can set this bit: One byte transfer, including
ACK/NACK bit,
 * completes if FACK is 0. An ACK or NACK is sent on the bus by writing 0
or 1 to TXAK
 * after this bit is set in receive mode. One byte transfer, excluding
ACK/NACK
 * bit, completes if FACK is 1. Match of slave address to calling address
including
 * primary slave address, range slave address , alert response address,
second
 * slave address, or general call address. Arbitration lost In SMBus
mode, any
 * timeouts except SCL and SDA high timeouts I2C bus stop detection if
the STOPIE
 * bit in the Input Glitch Filter register is 1 To clear the I2C bus stop
detection
 * interrupt: In the interrupt service routine, first clear the STOPF bit
in the

```

```

 * Input Glitch Filter register by writing 1 to it, and then clear the
IICIF
 * bit. If this sequence is reversed, the IICIF bit is asserted again.
 *
 * Values:
 * - 0b0 - No interrupt pending
 * - 0b1 - Interrupt pending
 */
/*@{*/
/*! @brief Read current value of the I2C_S_IICIF field. */
#define I2C_RD_S_IICIF(base) ((I2C_S_REG(base) & I2C_S_IICIF_MASK) >>
I2C_S_IICIF_SHIFT)
#define I2C_BRD_S_IICIF(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_IICIF_SHIFT, I2C_S_IICIF_WIDTH))

/*! @brief Set the IICIF field to a new value. */
#define I2C_WR_S_IICIF(base, value) (I2C_RMW_S(base, (I2C_S_IICIF_MASK |
I2C_S_ARBL_MASK), I2C_S_IICIF(value)))
#define I2C_BWR_S_IICIF(base, value) (BME_BFI8(&I2C_S_REG(base),
((uint8_t)(value) << I2C_S_IICIF_SHIFT), I2C_S_IICIF_SHIFT,
I2C_S_IICIF_WIDTH))
/*@}*/
/*!
 * @name Register I2C_S, field SRW[2] (RO)
 *
 * When addressed as a slave, SRW indicates the value of the R/W command
bit of
 * the calling address sent to the master.
 *
 * Values:
 * - 0b0 - Slave receive, master writing to slave
 * - 0b1 - Slave transmit, master reading from slave
 */
/*@{*/
/*! @brief Read current value of the I2C_S_SRW field. */
#define I2C_RD_S_SRW(base) ((I2C_S_REG(base) & I2C_S_SRW_MASK) >>
I2C_S_SRW_SHIFT)
#define I2C_BRD_S_SRW(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_SRW_SHIFT, I2C_S_SRW_WIDTH))
/*@}*/

/*!
 * @name Register I2C_S, field RAM[3] (RW)
 *
 * This bit is set to 1 by any of the following conditions: Any nonzero
calling
 * address is received that matches the address in the RA register. The
RMEN bit
 * is set and the calling address is within the range of values of the A1
and RA
 * registers. For the RAM bit to be set to 1 correctly, C1[IICIE] must be
set to
 * 1. Writing the C1 register with any value clears this bit to 0.

```

```

/*
 * Values:
 * - 0b0 - Not addressed
 * - 0b1 - Addressed as a slave
 */
/*@{*/
/*! @brief Read current value of the I2C_S_RAM field. */
#define I2C_RD_S_RAM(base) ((I2C_S_REG(base) & I2C_S_RAM_MASK) >>
I2C_S_RAM_SHIFT)
#define I2C_BRD_S_RAM(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_RAM_SHIFT, I2C_S_RAM_WIDTH))

/*! @brief Set the RAM field to a new value. */
#define I2C_WR_S_RAM(base, value) (I2C_RMW_S(base, (I2C_S_RAM_MASK |
I2C_S_IICIF_MASK | I2C_S_ARBL_MASK), I2C_S_RAM(value)))
#define I2C_BWR_S_RAM(base, value) (BME_BFI8(&I2C_S_REG(base),
((uint8_t)(value) << I2C_S_RAM_SHIFT), I2C_S_RAM_SHIFT, I2C_S_RAM_WIDTH))
/*}@*/ */

/*!
 * @name Register I2C_S, field ARBL[4] (W1C)
 *
 * This bit is set by hardware when the arbitration procedure is lost.
The ARBL
 * bit must be cleared by software, by writing a one to it.
 *
 * Values:
 * - 0b0 - Standard bus operation.
 * - 0b1 - Loss of arbitration.
 */
/*@{*/
/*! @brief Read current value of the I2C_S_ARBL field. */
#define I2C_RD_S_ARBL(base) ((I2C_S_REG(base) & I2C_S_ARBL_MASK) >>
I2C_S_ARBL_SHIFT)
#define I2C_BRD_S_ARBL(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_ARBL_SHIFT, I2C_S_ARBL_WIDTH))

/*! @brief Set the ARBL field to a new value. */
#define I2C_WR_S_ARBL(base, value) (I2C_RMW_S(base, (I2C_S_ARBL_MASK |
I2C_S_IICIF_MASK), I2C_S_ARBL(value)))
#define I2C_BWR_S_ARBL(base, value) (BME_BFI8(&I2C_S_REG(base),
((uint8_t)(value) << I2C_S_ARBL_SHIFT), I2C_S_ARBL_SHIFT,
I2C_S_ARBL_WIDTH))
/*}@*/ */

/*!
 * @name Register I2C_S, field BUSY[5] (RO)
 *
 * Indicates the status of the bus regardless of slave or master mode.
This bit
 * is set when a START signal is detected and cleared when a STOP signal
is
 * detected.
 *

```

```

 * Values:
 * - 0b0 - Bus is idle
 * - 0b1 - Bus is busy
 */
/*@{*/
/*! @brief Read current value of the I2C_S_BUSY field. */
#define I2C_RD_S_BUSY(base) ((I2C_S_REG(base) & I2C_S_BUSY_MASK) >>
I2C_S_BUSY_SHIFT)
#define I2C_BRD_S_BUSY(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_BUSY_SHIFT, I2C_S_BUSY_WIDTH))
/*}@*/
```

```

/*!
 * @name Register I2C_S, field IAAS[6] (RW)
 *
 * This bit is set by one of the following conditions: The calling
address
 * matches the programmed slave primary address in the A1 register or
range address in
 * the RA register (which must be set to a nonzero value). GCAEN is set
and a
 * general call is received. SIICAEN is set and the calling address
matches the
 * second programmed slave address. ALERTEN is set and an SMBus alert
response
 * address is received RMEN is set and an address is received that is
within the range
 * between the values of the A1 and RA registers. This bit sets before
the ACK
 * bit. The CPU must check the SRW bit and set TX/RX accordingly. Writing
the C1
 * register with any value clears this bit.
 *
 * Values:
 * - 0b0 - Not addressed
 * - 0b1 - Addressed as a slave
*/
/*@{*/
/*! @brief Read current value of the I2C_S_IAAS field. */
#define I2C_RD_S_IAAS(base) ((I2C_S_REG(base) & I2C_S_IAAS_MASK) >>
I2C_S_IAAS_SHIFT)
#define I2C_BRD_S_IAAS(base) (BME_UBFX8(&I2C_S_REG(base),
I2C_S_IAAS_SHIFT, I2C_S_IAAS_WIDTH))
```

```

/*! @brief Set the IAAS field to a new value. */
#define I2C_WR_S_IAAS(base, value) (I2C_RMW_S(base, (I2C_S_IAAS_MASK |
I2C_S_IICIF_MASK | I2C_S_ARBL_MASK), I2C_S_IAAS(value)))
#define I2C_BWR_S_IAAS(base, value) (BME_BFI8(&I2C_S_REG(base),
((uint8_t)(value) << I2C_S_IAAS_SHIFT), I2C_S_IAAS_SHIFT,
I2C_S_IAAS_WIDTH))
/*}@*/
```

```

/*!
 * @name Register I2C_S, field TCF[7] (RO)
```

```

/*
 * This bit sets on the completion of a byte and acknowledge bit
transfer. This
 * bit is valid only during or immediately following a transfer to or
from the
 * I2C module. The TCF bit is cleared by reading the I2C data register in
receive
 * mode or by writing to the I2C data register in transmit mode.
*
* Values:
* - 0b0 - Transfer in progress
* - 0b1 - Transfer complete
*/
/*@{/*/
/*! @brief Read current value of the I2C_S_TCF field. */
#define I2C_RD_S_TCF(base)    ((I2C_S_REG(base) & I2C_S_TCF_MASK) >>
I2C_S_TCF_SHIFT)
#define I2C_BRD_S_TCF(base)   (BME_UBFX8(&I2C_S_REG(base),
I2C_S_TCF_SHIFT, I2C_S_TCF_WIDTH))
/*@*/}

/***** ****
* I2C_D - I2C Data I/O register
***** */

/*!
* @brief I2C_D - I2C Data I/O register (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire I2C_D register
*/
/*@{/*/
#define I2C_RD_D(base)          (I2C_D_REG(base))
#define I2C_WR_D(base, value)   (I2C_D_REG(base) = (value))
#define I2C_RMW_D(base, mask, value) (I2C_WR_D(base, (I2C_RD_D(base) &
~(mask)) | (value)))
#define I2C_SET_D(base, value)   (BME_OR8(&I2C_D_REG(base),
(uint8_t)(value)))
#define I2C_CLR_D(base, value)   (BME_AND8(&I2C_D_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_D(base, value)   (BME_XOR8(&I2C_D_REG(base),
(uint8_t)(value)))
/*@*/}

/***** ****
* I2C_C2 - I2C Control Register 2
***** */

```

```
*****
 ****/
/*!
 * @brief I2C_C2 - I2C Control Register 2 (RW)
 *
 * Reset value: 0x00U
 */
/*!!
 * @name Constants and macros for entire I2C_C2 register
 */
/*@{*/
#define I2C_RD_C2(base)          (I2C_C2_REG(base))
#define I2C_WR_C2(base, value)    (I2C_C2_REG(base) = (value))
#define I2C_RMW_C2(base, mask, value) (I2C_WR_C2(base, (I2C_RD_C2(base) &
~(mask)) | (value)))
#define I2C_SET_C2(base, value)   (BME_OR8(&I2C_C2_REG(base),
(uint8_t)(value)))
#define I2C_CLR_C2(base, value)   (BME_AND8(&I2C_C2_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_C2(base, value)   (BME_XOR8(&I2C_C2_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual I2C_C2 bitfields
 */
/*!
 * @name Register I2C_C2, field AD[2:0] (RW)
 *
 * Contains the upper three bits of the slave address in the 10-bit
address
 * scheme. This field is valid only while the ADEXT bit is set.
 */
/*@*/
/*! @brief Read current value of the I2C_C2_AD field. */
#define I2C_RD_C2_AD(base) ((I2C_C2_REG(base) & I2C_C2_AD_MASK) >>
I2C_C2_AD_SHIFT)
#define I2C_BRD_C2_AD(base) (BME_UBFX8(&I2C_C2_REG(base),
I2C_C2_AD_SHIFT, I2C_C2_AD_WIDTH))

/*! @brief Set the AD field to a new value. */
#define I2C_WR_C2_AD(base, value) (I2C_RMW_C2(base, I2C_C2_AD_MASK,
I2C_C2_AD(value)))
#define I2C_BWR_C2_AD(base, value) (BME_BFI8(&I2C_C2_REG(base),
(uint8_t)(value) << I2C_C2_AD_SHIFT), I2C_C2_AD_SHIFT, I2C_C2_AD_WIDTH))
/*@}*/

/*
 * @name Register I2C_C2, field RMEN[3] (RW)
 */

```

```

 * This bit controls slave address matching for addresses between the
values of
 * the A1 and RA registers. When this bit is set, a slave address match
occurs
 * for any address greater than the value of the A1 register and less
than or equal
 * to the value of the RA register.
 *
 * Values:
 * - 0b0 - Range mode disabled. No address match occurs for an address
within
 *      the range of values of the A1 and RA registers.
 * - 0b1 - Range mode enabled. Address matching occurs when a slave
receives an
 *      address within the range of values of the A1 and RA registers.
 */
/*@{*/
/*! @brief Read current value of the I2C_C2_RMEN field. */
#define I2C_RD_C2_RMEN(base) ((I2C_C2_REG(base) & I2C_C2_RMEN_MASK) >>
I2C_C2_RMEN_SHIFT)
#define I2C_BRD_C2_RMEN(base) (BME_UBFX8(&I2C_C2_REG(base),
I2C_C2_RMEN_SHIFT, I2C_C2_RMEN_WIDTH))

/*! @brief Set the RMEN field to a new value. */
#define I2C_WR_C2_RMEN(base, value) (I2C_RMW_C2(base, I2C_C2_RMEN_MASK,
I2C_C2_RMEN(value)))
#define I2C_BWR_C2_RMEN(base, value) (BME_BFI8(&I2C_C2_REG(base),
((uint8_t)(value) << I2C_C2_RMEN_SHIFT), I2C_C2_RMEN_SHIFT,
I2C_C2_RMEN_WIDTH))
/*@}*/
/*
 * @name Register I2C_C2, field SBRC[4] (RW)
 *
 * Enables independent slave mode baud rate at maximum frequency, which
forces
 * clock stretching on SCL in very fast I2C modes. To a slave, an example
of a
 * "very fast" mode is when the master transfers at 40 kbps but the slave
can
 * capture the master's data at only 10 kbps.
 *
 * Values:
 * - 0b0 - The slave baud rate follows the master baud rate and clock
stretching
 *      may occur
 * - 0b1 - Slave baud rate is independent of the master baud rate
 */
/*@{*/
/*! @brief Read current value of the I2C_C2_SBRC field. */
#define I2C_RD_C2_SBRC(base) ((I2C_C2_REG(base) & I2C_C2_SBRC_MASK) >>
I2C_C2_SBRC_SHIFT)
#define I2C_BRD_C2_SBRC(base) (BME_UBFX8(&I2C_C2_REG(base),
I2C_C2_SBRC_SHIFT, I2C_C2_SBRC_WIDTH))

```

```

/*! @brief Set the SBRC field to a new value. */
#define I2C_WR_C2_SBRC(base, value) (I2C_RMW_C2(base, I2C_C2_SBRC_MASK,
I2C_C2_SBRC(value)))
#define I2C_BWR_C2_SBRC(base, value) (BME_BFI8(&I2C_C2_REG(base),
((uint8_t)(value) << I2C_C2_SBRC_SHIFT), I2C_C2_SBRC_SHIFT,
I2C_C2_SBRC_WIDTH))
/*@}*/

/*!!
 * @name Register I2C_C2, field HDRS[5] (RW)
 *
 * Controls the drive capability of the I2C pads.
 *
 * Values:
 * - 0b0 - Normal drive mode
 * - 0b1 - High drive mode
 */
/*@{*/
/*! @brief Read current value of the I2C_C2_HDRS field. */
#define I2C_RD_C2_HDRS(base) ((I2C_C2_REG(base) & I2C_C2_HDRS_MASK) >>
I2C_C2_HDRS_SHIFT)
#define I2C_BRD_C2_HDRS(base) (BME_UBFX8(&I2C_C2_REG(base),
I2C_C2_HDRS_SHIFT, I2C_C2_HDRS_WIDTH))

/*! @brief Set the HDRS field to a new value. */
#define I2C_WR_C2_HDRS(base, value) (I2C_RMW_C2(base, I2C_C2_HDRS_MASK,
I2C_C2_HDRS(value)))
#define I2C_BWR_C2_HDRS(base, value) (BME_BFI8(&I2C_C2_REG(base),
((uint8_t)(value) << I2C_C2_HDRS_SHIFT), I2C_C2_HDRS_SHIFT,
I2C_C2_HDRS_WIDTH))
/*@}*/

/*!!
 * @name Register I2C_C2, field ADEXT[6] (RW)
 *
 * Controls the number of bits used for the slave address.
 *
 * Values:
 * - 0b0 - 7-bit address scheme
 * - 0b1 - 10-bit address scheme
 */
/*@{*/
/*! @brief Read current value of the I2C_C2_ADEXT field. */
#define I2C_RD_C2_ADEXT(base) ((I2C_C2_REG(base) & I2C_C2_ADEXT_MASK) >>
I2C_C2_ADEXT_SHIFT)
#define I2C_BRD_C2_ADEXT(base) (BME_UBFX8(&I2C_C2_REG(base),
I2C_C2_ADEXT_SHIFT, I2C_C2_ADEXT_WIDTH))

/*! @brief Set the ADEXT field to a new value. */
#define I2C_WR_C2_ADEXT(base, value) (I2C_RMW_C2(base, I2C_C2_ADEXT_MASK,
I2C_C2_ADEXT(value)))

```

```

#define I2C_BWR_C2_ADEXT(base, value) (BME_BFI8(&I2C_C2_REG(base),  

((uint8_t)(value) << I2C_C2_ADEXT_SHIFT), I2C_C2_ADEXT_SHIFT,  

I2C_C2_ADEXT_WIDTH))  

/*@}*/

/*!  

 * @name Register I2C_C2, field GCAEN[7] (RW)  

 *  

 * Enables general call address.  

 *  

 * Values:  

 * - 0b0 - Disabled  

 * - 0b1 - Enabled  

 */  

/*@{*/  

/*! @brief Read current value of the I2C_C2_GCAEN field. */  

#define I2C_RD_C2_GCAEN(base) ((I2C_C2_REG(base) & I2C_C2_GCAEN_MASK) >>  

I2C_C2_GCAEN_SHIFT)  

#define I2C_BRD_C2_GCAEN(base) (BME_UBFX8(&I2C_C2_REG(base),  

I2C_C2_GCAEN_SHIFT, I2C_C2_GCAEN_WIDTH))

/*! @brief Set the GCAEN field to a new value. */  

#define I2C_WR_C2_GCAEN(base, value) (I2C_RMW_C2(base, I2C_C2_GCAEN_MASK,  

I2C_C2_GCAEN(value)))  

#define I2C_BWR_C2_GCAEN(base, value) (BME_BFI8(&I2C_C2_REG(base),  

((uint8_t)(value) << I2C_C2_GCAEN_SHIFT), I2C_C2_GCAEN_SHIFT,  

I2C_C2_GCAEN_WIDTH))  

/*@}*/

/*********************  

*****  

 * I2C_FLT - I2C Programmable Input Glitch Filter register  

*****  

******************/
```

```

/*!  

 * @brief I2C_FLT - I2C Programmable Input Glitch Filter register (RW)  

 *  

 * Reset value: 0x00U  

 */  

/*!  

 * @name Constants and macros for entire I2C_FLT register  

 */  

/*@{*/  

#define I2C_RD_FLT(base) (I2C_FLT_REG(base))  

#define I2C_WR_FLT(base, value) (I2C_FLT_REG(base) = (value))  

#define I2C_RMW_FLT(base, mask, value) (I2C_WR_FLT(base,  

(I2C_RD_FLT(base) & ~mask) | (value)))  

#define I2C_SET_FLT(base, value) (BME_OR8(&I2C_FLT_REG(base),  

(uint8_t)(value)))  

#define I2C_CLR_FLT(base, value) (BME_AND8(&I2C_FLT_REG(base),  

(uint8_t)(~(value))))
```

```

#define I2C_TOG_FLT(base, value) (BME_XOR8(&I2C_FLT_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual I2C_FLT bitfields
 */

/*!  

 * @name Register I2C_FLT, field FLT[4:0] (RW)  

 *  

 * Controls the width of the glitch, in terms of bus clock cycles, that  

the  

 * filter must absorb. For any glitch whose size is less than or equal to  

this width  

 * setting, the filter does not allow the glitch to pass.  

 *  

 * Values:  

 * - 0b00000 - No filter/bypass  

 */
/*@{*/  

/*! @brief Read current value of the I2C_FLT_FLT field. */  

#define I2C_RD_FLT_FLT(base) ((I2C_FLT_REG(base) & I2C_FLT_FLT_MASK) >>  

I2C_FLT_FLT_SHIFT)  

#define I2C_BRD_FLT_FLT(base) (BME_UBFX8(&I2C_FLT_REG(base),  

I2C_FLT_FLT_SHIFT, I2C_FLT_FLT_WIDTH))  

/*! @brief Set the FLT field to a new value. */  

#define I2C_WR_FLT_FLT(base, value) (I2C_RMW_FLT(base, (I2C_FLT_FLT_MASK  

| I2C_FLT_STOPF_MASK), I2C_FLT_FLT(value)))  

#define I2C_BWR_FLT_FLT(base, value) (BME_BFI8(&I2C_FLT_REG(base),  

((uint8_t)(value) << I2C_FLT_FLT_SHIFT), I2C_FLT_FLT_SHIFT,  

I2C_FLT_FLT_WIDTH))  

/*@}*/

/*
 * @name Register I2C_FLT, field STOPIE[5] (RW)  

 *  

 * This bit enables the interrupt for I2C bus stop detection. To clear  

the I2C  

 * bus stop detection interrupt: In the interrupt service routine, first  

clear the  

 * STOPF bit by writing 1 to it, and then clear the IICIF bit in the  

status  

 * register. If this sequence is reversed, the IICIF bit is asserted  

again.  

 *  

 * Values:  

 * - 0b0 - Stop detection interrupt is disabled  

 * - 0b1 - Stop detection interrupt is enabled  

 */
/*@{*/  

/*! @brief Read current value of the I2C_FLT_STOPIE field. */

```

```

#define I2C_RD_FLT_STOPIE(base) ((I2C_FLT_REG(base) &
I2C_FLT_STOPIE_MASK) >> I2C_FLT_STOPIE_SHIFT)
#define I2C_BRD_FLT_STOPIE(base) (BME_UBFX8(&I2C_FLT_REG(base),
I2C_FLT_STOPIE_SHIFT, I2C_FLT_STOPIE_WIDTH))

/*! @brief Set the STOPIE field to a new value. */
#define I2C_WR_FLT_STOPIE(base, value) (I2C_RMW_FLT(base,
(I2C_FLT_STOPIE_MASK | I2C_FLT_STOPF_MASK), I2C_FLT_STOPIE(value)))
#define I2C_BWR_FLT_STOPIE(base, value) (BME_BFI8(&I2C_FLT_REG(base),
((uint8_t)(value) << I2C_FLT_STOPIE_SHIFT), I2C_FLT_STOPIE_SHIFT,
I2C_FLT_STOPIE_WIDTH))
/*@}*/

/*!
 * @name Register I2C_FLT, field STOPF[6] (W1C)
 *
 * Hardware sets this bit when the I2C bus's stop status is detected. The
STOPF
 * bit must be cleared by writing 1 to it.
 *
 * Values:
 * - 0b0 - No stop happens on I2C bus
 * - 0b1 - Stop detected on I2C bus
 */
/*@*/
/*! @brief Read current value of the I2C_FLT_STOPF field. */
#define I2C_RD_FLT_STOPF(base) ((I2C_FLT_REG(base) & I2C_FLT_STOPF_MASK)
>> I2C_FLT_STOPF_SHIFT)
#define I2C_BRD_FLT_STOPF(base) (BME_UBFX8(&I2C_FLT_REG(base),
I2C_FLT_STOPF_SHIFT, I2C_FLT_STOPF_WIDTH))

/*! @brief Set the STOPF field to a new value. */
#define I2C_WR_FLT_STOPF(base, value) (I2C_RMW_FLT(base,
I2C_FLT_STOPF_MASK, I2C_FLT_STOPF(value)))
#define I2C_BWR_FLT_STOPF(base, value) (BME_BFI8(&I2C_FLT_REG(base),
((uint8_t)(value) << I2C_FLT_STOPF_SHIFT), I2C_FLT_STOPF_SHIFT,
I2C_FLT_STOPF_WIDTH))
/*@*/

/*!
 * @name Register I2C_FLT, field SHEN[7] (RW)
 *
 * Set this bit to hold off entry to stop mode when any data transmission
or
 * reception is occurring. The following scenario explains the holdoff
 * functionality: The I2C module is configured for a basic transfer, and
the SHEN bit is set
 * to 1. A transfer begins. The MCU signals the I2C module to enter stop
mode. The
 * byte currently being transferred, including both address and data,
completes
 * its transfer. The I2C slave or master acknowledges that the in-
transfer byte

```

```

    * completed its transfer and acknowledges the request to enter stop
mode. After
    * receiving the I2C module's acknowledgment of the request to enter stop
mode,
    * the MCU determines whether to shut off the I2C module's clock. If the
SHEN bit
    * is set to 1 and the I2C module is in an idle or disabled state when
the MCU
    * signals to enter stop mode, the module immediately acknowledges the
request to
    * enter stop mode. If SHEN is cleared to 0 and the overall data
transmission or
    * reception that was suspended by stop mode entry was incomplete: To
resume the
    * overall transmission or reception after the MCU exits stop mode,
software must
    * reinitialize the transfer by resending the address of the slave. If
the I2C
    * Control Register 1's IICIE bit was set to 1 before the MCU entered
stop mode,
    * system software will receive the interrupt triggered by the I2C Status
Register's
    * TCF bit after the MCU wakes from the stop mode.
    *
    * Values:
    * - 0b0 - Stop holdoff is disabled. The MCU's entry to stop mode is not
gated.
    * - 0b1 - Stop holdoff is enabled.
    */
/*@{*/
/*! @brief Read current value of the I2C_FLT_SHEN field. */
#define I2C_RD_FLT_SHEN(base) ((I2C_FLT_REG(base) & I2C_FLT_SHEN_MASK) >>
I2C_FLT_SHEN_SHIFT)
#define I2C_BRD_FLT_SHEN(base) (BME_UBFX8(&I2C_FLT_REG(base),
I2C_FLT_SHEN_SHIFT, I2C_FLT_SHEN_WIDTH))

/*! @brief Set the SHEN field to a new value. */
#define I2C_WR_FLT_SHEN(base, value) (I2C_RMW_FLT(base,
(I2C_FLT_SHEN_MASK | I2C_FLT_STOPF_MASK), I2C_FLT_SHEN(value)))
#define I2C_BWR_FLT_SHEN(base, value) (BME_BFI8(&I2C_FLT_REG(base),
((uint8_t)(value) << I2C_FLT_SHEN_SHIFT), I2C_FLT_SHEN_SHIFT,
I2C_FLT_SHEN_WIDTH))
/*}@*/
```

\*\*\*\*\*

    \* I2C\_RA - I2C Range Address register

\*\*\*\*\*/

/\*!

    \* @brief I2C\_RA - I2C Range Address register (RW)

    \*

```

 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire I2C_RA register
 */
/*@{*/
#define I2C_RD_RA(base)          (I2C_RA_REG(base))
#define I2C_WR_RA(base, value)    (I2C_RA_REG(base) = (value))
#define I2C_RMW_RA(base, mask, value) (I2C_WR_RA(base, (I2C_RD_RA(base) &
~(mask)) | (value)))
#define I2C_SET_RA(base, value)   (BME_OR8(&I2C_RA_REG(base),
(uint8_t)(value)))
#define I2C_CLR_RA(base, value)   (BME_AND8(&I2C_RA_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_RA(base, value)   (BME_XOR8(&I2C_RA_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual I2C_RA bitfields
 */
/*!
 * @name Register I2C_RA, field RAD[7:1] (RW)
 *
 * This field contains the slave address to be used by the I2C module.
The field
 * is used in the 7-bit address scheme. Any nonzero write enables this
register.
 * This register's use is similar to that of the A1 register, but in
addition
 * this register can be considered a maximum boundary in range matching
mode.
 */
/*@{*/
/*! @brief Read current value of the I2C_RA_RAD field. */
#define I2C_RD_RA_RAD(base) ((I2C_RA_REG(base) & I2C_RA_RAD_MASK) >>
I2C_RA_RAD_SHIFT)
#define I2C_BRD_RA_RAD(base) (BME_UBFX8(&I2C_RA_REG(base),
I2C_RA_RAD_SHIFT, I2C_RA_RAD_WIDTH))

/*! @brief Set the RAD field to a new value. */
#define I2C_WR_RA_RAD(base, value) (I2C_RMW_RA(base, I2C_RA_RAD_MASK,
I2C_RA_RAD(value)))
#define I2C_BWR_RA_RAD(base, value) (BME_BFI8(&I2C_RA_REG(base),
((uint8_t)(value) << I2C_RA_RAD_SHIFT), I2C_RA_RAD_SHIFT,
I2C_RA_RAD_WIDTH))
/*}@*/
*****  

* I2C_SMB - I2C SMBus Control and Status register

```

```
*****
 ****/
/*!
 * @brief I2C_SMB - I2C SMBus Control and Status register (RW)
 *
 * Reset value: 0x00U
 *
 * When the SCL and SDA signals are held high for a length of time
 * greater than
 * the high timeout period, the SHTF1 flag sets. Before reaching this
 * threshold,
 * while the system is detecting how long these signals are being held
 * high, a
 * master assumes that the bus is free. However, the SHTF1 bit rises in
 * the bus
 * transmission process with the idle bus state. When the TCKSEL bit is
 * set, there
 * is no need to monitor the SHTF1 bit because the bus speed is too high
 * to match
 * the protocol of SMBus.
 */
/*!
 * @name Constants and macros for entire I2C_SMB register
 */
/*@{*/
#define I2C_RD_SMB(base)          (I2C_SMB_REG(base))
#define I2C_WR_SMB(base, value)   (I2C_SMB_REG(base) = (value))
#define I2C_RMW_SMB(base, mask, value) (I2C_WR_SMB(base,
(I2C_RD_SMB(base) & ~mask) | (value)))
#define I2C_SET_SMB(base, value)  (BME_OR8(&I2C_SMB_REG(base),
(uint8_t)(value)))
#define I2C_CLR_SMB(base, value)  (BME_AND8(&I2C_SMB_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_SMB(base, value)  (BME_XOR8(&I2C_SMB_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual I2C_SMB bitfields
 */
/*!
 * @name Register I2C_SMB, field SHTF2IE[0] (RW)
 *
 * Enables SCL high and SDA low timeout interrupt.
 *
 * Values:
 * - 0b0 - SHTF2 interrupt is disabled
 * - 0b1 - SHTF2 interrupt is enabled
 */
/*@*/
/*! @brief Read current value of the I2C_SMB_SHTF2IE field. */

```

```

#define I2C_RD_SMB_SHTF2IE(base) ((I2C_SMB_REG(base) &
I2C_SMB_SHTF2IE_MASK) >> I2C_SMB_SHTF2IE_SHIFT)
#define I2C_BRD_SMB_SHTF2IE(base) (BME_UBFX8(&I2C_SMB_REG(base),
I2C_SMB_SHTF2IE_SHIFT, I2C_SMB_SHTF2IE_WIDTH))

/*! @brief Set the SHTF2IE field to a new value. */
#define I2C_WR_SMB_SHTF2IE(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_SHTF2IE_MASK | I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK),
I2C_SMB_SHTF2IE(value)))
#define I2C_BWR_SMB_SHTF2IE(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_SHTF2IE_SHIFT), I2C_SMB_SHTF2IE_SHIFT,
I2C_SMB_SHTF2IE_WIDTH))
/*@}*/

/*!
 * @name Register I2C_SMB, field SHTF2[1] (W1C)
 *
 * This bit sets when SCL is held high and SDA is held low more than
clock *
 * LoValue/512. Software clears this bit by writing a 1 to it.
*
* Values:
* - 0b0 - No SCL high and SDA low timeout occurs
* - 0b1 - SCL high and SDA low timeout occurs
*/
/*@{*/
/*! @brief Read current value of the I2C_SMB_SHTF2 field. */
#define I2C_RD_SMB_SHTF2(base) ((I2C_SMB_REG(base) & I2C_SMB_SHTF2_MASK)
>> I2C_SMB_SHTF2_SHIFT)
#define I2C_BRD_SMB_SHTF2(base) (BME_UBFX8(&I2C_SMB_REG(base),
I2C_SMB_SHTF2_SHIFT, I2C_SMB_SHTF2_WIDTH))

/*! @brief Set the SHTF2 field to a new value. */
#define I2C_WR_SMB_SHTF2(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK), I2C_SMB_SHTF2(value)))
#define I2C_BWR_SMB_SHTF2(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_SHTF2_SHIFT), I2C_SMB_SHTF2_SHIFT,
I2C_SMB_SHTF2_WIDTH))
/*@}*/

/*!
 * @name Register I2C_SMB, field SHTF1[2] (RO)
*
* This read-only bit sets when SCL and SDA are held high more than clock
*
* LoValue / 512, which indicates the bus is free. This bit is cleared
automatically.
*
* Values:
* - 0b0 - No SCL high and SDA high timeout occurs
* - 0b1 - SCL high and SDA high timeout occurs
*/
/*@{*/
/*! @brief Read current value of the I2C_SMB_SHTF1 field. */

```

```

#define I2C_RD_SMB_SHTF1(base) ((I2C_SMB_REG(base) & I2C_SMB_SHTF1_MASK) >> I2C_SMB_SHTF1_SHIFT)
#define I2C_BRD_SMB_SHTF1(base) (BME_UBFX8(&I2C_SMB_REG(base), I2C_SMB_SHTF1_SHIFT, I2C_SMB_SHTF1_WIDTH))
/*@}*/

/*!
 * @name Register I2C_SMB, field SLTF[3] (W1C)
 *
 * This bit is set when the SLT register (consisting of the SLTH and SLTL registers) is loaded with a non-zero value (LoValue) and an SCL low timeout occurs.
 * Software clears this bit by writing a logic 1 to it. The low timeout function
 * is disabled when the SLT register's value is zero.
 *
 * Values:
 * - 0b0 - No low timeout occurs
 * - 0b1 - Low timeout occurs
 */
/*@*/
/*! @brief Read current value of the I2C_SMB_SLTF field. */
#define I2C_RD_SMB_SLTF(base) ((I2C_SMB_REG(base) & I2C_SMB_SLTF_MASK) >> I2C_SMB_SLTF_SHIFT)
#define I2C_BRD_SMB_SLTF(base) (BME_UBFX8(&I2C_SMB_REG(base), I2C_SMB_SLTF_SHIFT, I2C_SMB_SLTF_WIDTH))

/*! @brief Set the SLTF field to a new value. */
#define I2C_WR_SMB_SLTF(base, value) (I2C_RMW_SMB(base, (I2C_SMB_SLTF_MASK | I2C_SMB_SHTF2_MASK), I2C_SMB_SLTF(value)))
#define I2C_BWR_SMB_SLTF(base, value) (BME_BFI8(&I2C_SMB_REG(base), ((uint8_t)(value) << I2C_SMB_SLTF_SHIFT), I2C_SMB_SLTF_SHIFT, I2C_SMB_SLTF_WIDTH))
/*@*/

/*!
 * @name Register I2C_SMB, field TCKSEL[4] (RW)
 *
 * Selects the clock source of the timeout counter.
 *
 * Values:
 * - 0b0 - Timeout counter counts at the frequency of the bus clock / 64
 * - 0b1 - Timeout counter counts at the frequency of the bus clock
 */
/*@*/
/*! @brief Read current value of the I2C_SMB_TCKSEL field. */
#define I2C_RD_SMB_TCKSEL(base) ((I2C_SMB_REG(base) & I2C_SMB_TCKSEL_MASK) >> I2C_SMB_TCKSEL_SHIFT)
#define I2C_BRD_SMB_TCKSEL(base) (BME_UBFX8(&I2C_SMB_REG(base), I2C_SMB_TCKSEL_SHIFT, I2C_SMB_TCKSEL_WIDTH))

/*! @brief Set the TCKSEL field to a new value. */

```

```

#define I2C_WR_SMB_TCKSEL(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_TCKSEL_MASK | I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK),
I2C_SMB_TCKSEL(value)))
#define I2C_BWR_SMB_TCKSEL(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_TCKSEL_SHIFT), I2C_SMB_TCKSEL_SHIFT,
I2C_SMB_TCKSEL_WIDTH))
/*@}*/

/*
 * @name Register I2C_SMB, field SIICAEN[5] (RW)
 *
 * Enables or disables SMBus device default address.
 *
 * Values:
 * - 0b0 - I2C address register 2 matching is disabled
 * - 0b1 - I2C address register 2 matching is enabled
 */
/*@{*/
/**! @brief Read current value of the I2C_SMB_SIICAEN field. */
#define I2C_RD_SMB_SIICAEN(base) ((I2C_SMB_REG(base) &
I2C_SMB_SIICAEN_MASK) >> I2C_SMB_SIICAEN_SHIFT)
#define I2C_BRD_SMB_SIICAEN(base) (BME_UBFX8(&I2C_SMB_REG(base),
I2C_SMB_SIICAEN_SHIFT, I2C_SMB_SIICAEN_WIDTH))

/**! @brief Set the SIICAEN field to a new value. */
#define I2C_WR_SMB_SIICAEN(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_SIICAEN_MASK | I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK),
I2C_SMB_SIICAEN(value)))
#define I2C_BWR_SMB_SIICAEN(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_SIICAEN_SHIFT), I2C_SMB_SIICAEN_SHIFT,
I2C_SMB_SIICAEN_WIDTH))
/*@}*/

/*
 * @name Register I2C_SMB, field ALERTEN[6] (RW)
 *
 * Enables or disables SMBus alert response address matching. After the
host
 * responds to a device that used the alert response address, you must
use software
 * to put the device's address on the bus. The alert protocol is
described in the
 * SMBus specification.
 *
 * Values:
 * - 0b0 - SMBus alert response address matching is disabled
 * - 0b1 - SMBus alert response address matching is enabled
 */
/*@{*/
/**! @brief Read current value of the I2C_SMB_ALERTEN field. */
#define I2C_RD_SMB_ALERTEN(base) ((I2C_SMB_REG(base) &
I2C_SMB_ALERTEN_MASK) >> I2C_SMB_ALERTEN_SHIFT)
#define I2C_BRD_SMB_ALERTEN(base) (BME_UBFX8(&I2C_SMB_REG(base),
I2C_SMB_ALERTEN_SHIFT, I2C_SMB_ALERTEN_WIDTH))

```

```

/*! @brief Set the ALERTEN field to a new value. */
#define I2C_WR_SMB_ALERTEN(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_ALERTEN_MASK | I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK),
I2C_SMB_ALERTEN(value)))
#define I2C_BWR_SMB_ALERTEN(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_ALERTEN_SHIFT), I2C_SMB_ALERTEN_SHIFT,
I2C_SMB_ALERTEN_WIDTH))
/*@}*/

/*!
 * @name Register I2C_SMB, field FACK[7] (RW)
 *
 * For SMBus packet error checking, the CPU must be able to issue an ACK
or NACK
 * according to the result of receiving data byte.
 *
 * Values:
 * - 0b0 - An ACK or NACK is sent on the following receiving data byte
 * - 0b1 - Writing 0 to TXAK after receiving a data byte generates an
ACK.
 *     Writing 1 to TXAK after receiving a data byte generates a NACK.
 */
/*@{*/
/*! @brief Read current value of the I2C_SMB_FACK field. */
#define I2C_RD_SMB_FACK(base) ((I2C_SMB_REG(base) & I2C_SMB_FACK_MASK) >>
I2C_SMB_FACK_SHIFT)
#define I2C_BRD_SMB_FACK(base) (BME_UBFX8(&I2C_SMB_REG(base),
I2C_SMB_FACK_SHIFT, I2C_SMB_FACK_WIDTH))

/*! @brief Set the FACK field to a new value. */
#define I2C_WR_SMB_FACK(base, value) (I2C_RMW_SMB(base,
(I2C_SMB_FACK_MASK | I2C_SMB_SHTF2_MASK | I2C_SMB_SLTF_MASK),
I2C_SMB_FACK(value)))
#define I2C_BWR_SMB_FACK(base, value) (BME_BFI8(&I2C_SMB_REG(base),
((uint8_t)(value) << I2C_SMB_FACK_SHIFT), I2C_SMB_FACK_SHIFT,
I2C_SMB_FACK_WIDTH))
/*@}*/

*****  

*****  

 * I2C_A2 - I2C Address Register 2  

*****  

*****  

/*!
 * @brief I2C_A2 - I2C Address Register 2 (RW)
 *
 * Reset value: 0xC2U
 */
/*!
 * @name Constants and macros for entire I2C_A2 register
 */

```

```

/*@{ */
#define I2C_RD_A2(base)          (I2C_A2_REG(base))
#define I2C_WR_A2(base, value)    (I2C_A2_REG(base) = (value))
#define I2C_RMW_A2(base, mask, value) (I2C_WR_A2(base, (I2C_RD_A2(base) &
~(mask)) | (value)))
#define I2C_SET_A2(base, value)   (BME_OR8(&I2C_A2_REG(base),
(uint8_t)(value)))
#define I2C_CLR_A2(base, value)   (BME_AND8(&I2C_A2_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_A2(base, value)   (BME_XOR8(&I2C_A2_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual I2C_A2 bitfields
 */

/*!
 * @name Register I2C_A2, field SAD[7:1] (RW)
 *
 * Contains the slave address used by the SMBus. This field is used on
the
 * device default address or other related addresses.
 */
/*@{ */
/*! @brief Read current value of the I2C_A2_SAD field. */
#define I2C_RD_A2_SAD(base) ((I2C_A2_REG(base) & I2C_A2_SAD_MASK) >>
I2C_A2_SAD_SHIFT)
#define I2C_BRD_A2_SAD(base) (BME_UBFX8(&I2C_A2_REG(base),
I2C_A2_SAD_SHIFT, I2C_A2_SAD_WIDTH))

/*! @brief Set the SAD field to a new value. */
#define I2C_WR_A2_SAD(base, value) (I2C_RMW_A2(base, I2C_A2_SAD_MASK,
I2C_A2_SAD(value)))
#define I2C_BWR_A2_SAD(base, value) (BME_BFI8(&I2C_A2_REG(base),
((uint8_t)(value) << I2C_A2_SAD_SHIFT), I2C_A2_SAD_SHIFT,
I2C_A2_SAD_WIDTH))
/*@} */

*****  

*****  

* I2C_SLTH - I2C SCL Low Timeout Register High  

*****  

*****  

/*!  

 * @brief I2C_SLTH - I2C SCL Low Timeout Register High (RW)
 *
 * Reset value: 0x00U
 */
/*!  

 * @name Constants and macros for entire I2C_SLTH register
 */

```

```

/*@{*/
#define I2C_RD_SLTH(base)          (I2C_SLTH_REG(base))
#define I2C_WR_SLTH(base, value)   (I2C_SLTH_REG(base) = (value))
#define I2C_RMW_SLTH(base, mask, value) (I2C_WR_SLTH(base,
(I2C_RD_SLTH(base) & ~mask) | (value)))
#define I2C_SET_SLTH(base, value)  (BME_OR8(&I2C_SLTH_REG(base),
(uint8_t)(value)))
#define I2C_CLR_SLTH(base, value)  (BME_AND8(&I2C_SLTH_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_SLTH(base, value)  (BME_XOR8(&I2C_SLTH_REG(base),
(uint8_t)(value)))
/*@}*/
/******
 * I2C_SLTL - I2C SCL Low Timeout Register Low
*****/
/*!
 * @brief I2C_SLTL - I2C SCL Low Timeout Register Low (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire I2C_SLTL register
 */
/*@*/
#define I2C_RD_SLTL(base)          (I2C_SLTL_REG(base))
#define I2C_WR_SLTL(base, value)   (I2C_SLTL_REG(base) = (value))
#define I2C_RMW_SLTL(base, mask, value) (I2C_WR_SLTL(base,
(I2C_RD_SLTL(base) & ~mask) | (value)))
#define I2C_SET_SLTL(base, value)  (BME_OR8(&I2C_SLTL_REG(base),
(uint8_t)(value)))
#define I2C_CLR_SLTL(base, value)  (BME_AND8(&I2C_SLTL_REG(base),
(uint8_t)(~(value))))
#define I2C_TOG_SLTL(base, value)  (BME_XOR8(&I2C_SLTL_REG(base),
(uint8_t)(value)))
/*@*/
/*
 * MKL25Z4 LLWU
 *
 * Low leakage wakeup unit
 *
 * Registers defined in this header file:
 * - LLWU_PE1 - LLWU Pin Enable 1 register
 * - LLWU_PE2 - LLWU Pin Enable 2 register
 * - LLWU_PE3 - LLWU Pin Enable 3 register
 * - LLWU_PE4 - LLWU Pin Enable 4 register
 * - LLWU_ME - LLWU Module Enable register
 * - LLWU_F1 - LLWU Flag 1 register
 * - LLWU_F2 - LLWU Flag 2 register

```

```

* - LLWU_F3 - LLWU Flag 3 register
* - LLWU_FILTER1 - LLWU Pin Filter 1 register
* - LLWU_FILTER2 - LLWU Pin Filter 2 register
*/
#define LLWU_INSTANCE_COUNT (1U) /*!< Number of instances of the LLWU
module. */
#define LLWU_IDX (0U) /*!< Instance number for LLWU. */

/*****
 * LLWU_PE1 - LLWU Pin Enable 1 register
****/

/*!
 * @brief LLWU_PE1 - LLWU Pin Enable 1 register (RW)
 *
 * Reset value: 0x00U
 *
 * LLWU_PE1 contains the field to enable and select the edge detect type
for the
 * external wakeup input pins LLWU_P3-LLWU_P0. This register is reset on
Chip
 * Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It
is
 * unaffected by reset types that do not trigger Chip Reset not VLLS. See
the
 * Introduction details for more information.
 */
/*!
 * @name Constants and macros for entire LLWU_PE1 register
 */
/*@{*/
#define LLWU_RD_PE1(base)          (LLWU_PE1_REG(base))
#define LLWU_WR_PE1(base, value)   (LLWU_PE1_REG(base) = (value))
#define LLWU_RMW_PE1(base, mask, value) (LLWU_WR_PE1(base,
(LLWU_RD_PE1(base) & ~mask) | (value)))
#define LLWU_SET_PE1(base, value)  (BME_OR8(&LLWU_PE1_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_PE1(base, value)  (BME_AND8(&LLWU_PE1_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_PE1(base, value)  (BME_XOR8(&LLWU_PE1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual LLWU_PE1 bitfields
 */

/*!
 * @name Register LLWU_PE1, field WUPE0[1:0] (RW)
 *

```

```

 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE1_WUPE0 field. */
#define LLWU_RD_PE1_WUPE0(base) ((LLWU_PE1_REG(base) &
LLWU_PE1_WUPE0_MASK) >> LLWU_PE1_WUPE0_SHIFT)
#define LLWU_BRD_PE1_WUPE0(base) (BME_UBFX8(&LLWU_PE1_REG(base),
LLWU_PE1_WUPE0_SHIFT, LLWU_PE1_WUPE0_WIDTH))

/*! @brief Set the WUPE0 field to a new value. */
#define LLWU_WR_PE1_WUPE0(base, value) (LLWU_RMW_PE1(base,
LLWU_PE1_WUPE0_MASK, LLWU_PE1_WUPE0(value)))
#define LLWU_BWR_PE1_WUPE0(base, value) (BME_BFI8(&LLWU_PE1_REG(base),
((uint8_t)(value) << LLWU_PE1_WUPE0_SHIFT), LLWU_PE1_WUPE0_SHIFT,
LLWU_PE1_WUPE0_WIDTH))
/*}@*/ */

/*!
 * @name Register LLWU_PE1, field WUPE1[3:2] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE1_WUPE1 field. */
#define LLWU_RD_PE1_WUPE1(base) ((LLWU_PE1_REG(base) &
LLWU_PE1_WUPE1_MASK) >> LLWU_PE1_WUPE1_SHIFT)
#define LLWU_BRD_PE1_WUPE1(base) (BME_UBFX8(&LLWU_PE1_REG(base),
LLWU_PE1_WUPE1_SHIFT, LLWU_PE1_WUPE1_WIDTH))

/*! @brief Set the WUPE1 field to a new value. */
#define LLWU_WR_PE1_WUPE1(base, value) (LLWU_RMW_PE1(base,
LLWU_PE1_WUPE1_MASK, LLWU_PE1_WUPE1(value)))
#define LLWU_BWR_PE1_WUPE1(base, value) (BME_BFI8(&LLWU_PE1_REG(base),
((uint8_t)(value) << LLWU_PE1_WUPE1_SHIFT), LLWU_PE1_WUPE1_SHIFT,
LLWU_PE1_WUPE1_WIDTH))
/*}@*/ */

/*!
 * @name Register LLWU_PE1, field WUPE2[5:4] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *

```

```

* Values:
* - 0b00 - External input pin disabled as wakeup input
* - 0b01 - External input pin enabled with rising edge detection
* - 0b10 - External input pin enabled with falling edge detection
* - 0b11 - External input pin enabled with any change detection
*/
/*@{ */
/*! @brief Read current value of the LLWU_PE1_WUPE2 field. */
#define LLWU_RD_PE1_WUPE2(base) ((LLWU_PE1_REG(base) &
LLWU_PE1_WUPE2_MASK) >> LLWU_PE1_WUPE2_SHIFT)
#define LLWU_BRD_PE1_WUPE2(base) (BME_UBFX8(&LLWU_PE1_REG(base),
LLWU_PE1_WUPE2_SHIFT, LLWU_PE1_WUPE2_WIDTH))

/*! @brief Set the WUPE2 field to a new value. */
#define LLWU_WR_PE1_WUPE2(base, value) (LLWU_RMW_PE1(base,
LLWU_PE1_WUPE2_MASK, LLWU_PE1_WUPE2(value)))
#define LLWU_BWR_PE1_WUPE2(base, value) (BME_BFI8(&LLWU_PE1_REG(base),
((uint8_t)(value) << LLWU_PE1_WUPE2_SHIFT), LLWU_PE1_WUPE2_SHIFT,
LLWU_PE1_WUPE2_WIDTH))
/*}@*/ */

/*!
* @name Register LLWU_PE1, field WUPE3[7:6] (RW)
*
* Enables and configures the edge detection for the wakeup pin.
*
* Values:
* - 0b00 - External input pin disabled as wakeup input
* - 0b01 - External input pin enabled with rising edge detection
* - 0b10 - External input pin enabled with falling edge detection
* - 0b11 - External input pin enabled with any change detection
*/
/*@{ */
/*! @brief Read current value of the LLWU_PE1_WUPE3 field. */
#define LLWU_RD_PE1_WUPE3(base) ((LLWU_PE1_REG(base) &
LLWU_PE1_WUPE3_MASK) >> LLWU_PE1_WUPE3_SHIFT)
#define LLWU_BRD_PE1_WUPE3(base) (BME_UBFX8(&LLWU_PE1_REG(base),
LLWU_PE1_WUPE3_SHIFT, LLWU_PE1_WUPE3_WIDTH))

/*! @brief Set the WUPE3 field to a new value. */
#define LLWU_WR_PE1_WUPE3(base, value) (LLWU_RMW_PE1(base,
LLWU_PE1_WUPE3_MASK, LLWU_PE1_WUPE3(value)))
#define LLWU_BWR_PE1_WUPE3(base, value) (BME_BFI8(&LLWU_PE1_REG(base),
((uint8_t)(value) << LLWU_PE1_WUPE3_SHIFT), LLWU_PE1_WUPE3_SHIFT,
LLWU_PE1_WUPE3_WIDTH))
/*}@*/ */

*****  

*****  

* LLWU_PE2 - LLWU Pin Enable 2 register  

*****  

*****  


```

```

/*!
 * @brief LLWU_PE2 - LLWU Pin Enable 2 register (RW)
 *
 * Reset value: 0x00U
 *
 * LLWU_PE2 contains the field to enable and select the edge detect type
 * for the
 * external wakeup input pins LLWU_P7-LLWU_P4. This register is reset on
 * Chip
 * Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It
 * is
 * unaffected by reset types that do not trigger Chip Reset not VLLS. See
 * the
 * Introduction details for more information.
 */
/*!
 * @name Constants and macros for entire LLWU_PE2 register
 */
/*@{ */
#define LLWU_RD_PE2(base)          (LLWU_PE2_REG(base))
#define LLWU_WR_PE2(base, value)   (LLWU_PE2_REG(base) = (value))
#define LLWU_RMW_PE2(base, mask, value) (LLWU_WR_PE2(base,
(LLWU_RD_PE2(base) & ~mask) | (value)))
#define LLWU_SET_PE2(base, value)  (BME_OR8(&LLWU_PE2_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_PE2(base, value)  (BME_AND8(&LLWU_PE2_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_PE2(base, value)  (BME_XOR8(&LLWU_PE2_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual LLWU_PE2 bitfields
 */

/*!
 * @name Register LLWU_PE2, field WUPE4[1:0] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{ */
/*! @brief Read current value of the LLWU_PE2_WUPE4 field. */
#define LLWU_RD_PE2_WUPE4(base) ((LLWU_PE2_REG(base) &
LLWU_PE2_WUPE4_MASK) >> LLWU_PE2_WUPE4_SHIFT)
#define LLWU_BRD_PE2_WUPE4(base) (BME_UBFX8(&LLWU_PE2_REG(base),
LLWU_PE2_WUPE4_SHIFT, LLWU_PE2_WUPE4_WIDTH))

/*! @brief Set the WUPE4 field to a new value. */

```

```

#define LLWU_WR_PE2_WUPE4(base, value) (LLWU_RMW_PE2(base,
LLWU_PE2_WUPE4_MASK, LLWU_PE2_WUPE4(value)))
#define LLWU_BWR_PE2_WUPE4(base, value) (BME_BFI8(&LLWU_PE2_REG(base),
((uint8_t)(value) << LLWU_PE2_WUPE4_SHIFT), LLWU_PE2_WUPE4_SHIFT,
LLWU_PE2_WUPE4_WIDTH))
/*@}*/

/*
 * @name Register LLWU_PE2, field WUPE5[3:2] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@*/
/*! @brief Read current value of the LLWU_PE2_WUPE5 field. */
#define LLWU_RD_PE2_WUPE5(base) ((LLWU_PE2_REG(base) &
LLWU_PE2_WUPE5_MASK) >> LLWU_PE2_WUPE5_SHIFT)
#define LLWU_BRD_PE2_WUPE5(base) (BME_UBFX8(&LLWU_PE2_REG(base),
LLWU_PE2_WUPE5_SHIFT, LLWU_PE2_WUPE5_WIDTH))

/*! @brief Set the WUPE5 field to a new value. */
#define LLWU_WR_PE2_WUPE5(base, value) (LLWU_RMW_PE2(base,
LLWU_PE2_WUPE5_MASK, LLWU_PE2_WUPE5(value)))
#define LLWU_BWR_PE2_WUPE5(base, value) (BME_BFI8(&LLWU_PE2_REG(base),
((uint8_t)(value) << LLWU_PE2_WUPE5_SHIFT), LLWU_PE2_WUPE5_SHIFT,
LLWU_PE2_WUPE5_WIDTH))
/*@*/

/*
 * @name Register LLWU_PE2, field WUPE6[5:4] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@*/
/*! @brief Read current value of the LLWU_PE2_WUPE6 field. */
#define LLWU_RD_PE2_WUPE6(base) ((LLWU_PE2_REG(base) &
LLWU_PE2_WUPE6_MASK) >> LLWU_PE2_WUPE6_SHIFT)
#define LLWU_BRD_PE2_WUPE6(base) (BME_UBFX8(&LLWU_PE2_REG(base),
LLWU_PE2_WUPE6_SHIFT, LLWU_PE2_WUPE6_WIDTH))

/*! @brief Set the WUPE6 field to a new value. */
#define LLWU_WR_PE2_WUPE6(base, value) (LLWU_RMW_PE2(base,
LLWU_PE2_WUPE6_MASK, LLWU_PE2_WUPE6(value)))

```

```

#define LLWU_BWR_PE2_WUPE6(base, value) (BME_BFI8(&LLWU_PE2_REG(base),  

((uint8_t)(value) << LLWU_PE2_WUPE6_SHIFT), LLWU_PE2_WUPE6_SHIFT,  

LLWU_PE2_WUPE6_WIDTH))  

/*@}*/

/*!  

 * @name Register LLWU_PE2, field WUPE7[7:6] (RW)  

 *  

 * Enables and configures the edge detection for the wakeup pin.  

 *  

 * Values:  

 * - 0b00 - External input pin disabled as wakeup input  

 * - 0b01 - External input pin enabled with rising edge detection  

 * - 0b10 - External input pin enabled with falling edge detection  

 * - 0b11 - External input pin enabled with any change detection  

 */  

/*@{*/  

/*! @brief Read current value of the LLWU_PE2_WUPE7 field. */  

#define LLWU_RD_PE2_WUPE7(base) ((LLWU_PE2_REG(base) &  

LLWU_PE2_WUPE7_MASK) >> LLWU_PE2_WUPE7_SHIFT)  

#define LLWU_BRD_PE2_WUPE7(base) (BME_UBFX8(&LLWU_PE2_REG(base),  

LLWU_PE2_WUPE7_SHIFT, LLWU_PE2_WUPE7_WIDTH))

/*! @brief Set the WUPE7 field to a new value. */  

#define LLWU_WR_PE2_WUPE7(base, value) (LLWU_RMW_PE2(base,  

LLWU_PE2_WUPE7_MASK, LLWU_PE2_WUPE7(value)))  

#define LLWU_BWR_PE2_WUPE7(base, value) (BME_BFI8(&LLWU_PE2_REG(base),  

((uint8_t)(value) << LLWU_PE2_WUPE7_SHIFT), LLWU_PE2_WUPE7_SHIFT,  

LLWU_PE2_WUPE7_WIDTH))  

/*@}*/

*****  

*****  

 * LLWU_PE3 - LLWU Pin Enable 3 register  

*****  

*****  

*****  

/*!  

 * @brief LLWU_PE3 - LLWU Pin Enable 3 register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * LLWU_PE3 contains the field to enable and select the edge detect type  

for the  

 * external wakeup input pins LLWU_P11-LLWU_P8. This register is reset on  

Chip  

 * Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It  

is  

 * unaffected by reset types that do not trigger Chip Reset not VLLS. See  

the  

 * Introduction details for more information.  

 */  

/*!
```

```

/* @name Constants and macros for entire LLWU_PE3 register
 */
/*@{*/
#define LLWU_RD_PE3(base)          (LLWU_PE3_REG(base))
#define LLWU_WR_PE3(base, value)   (LLWU_PE3_REG(base) = (value))
#define LLWU_RMW_PE3(base, mask, value) (LLWU_WR_PE3(base,
(LLWU_RD_PE3(base) & ~mask)) | (value)))
#define LLWU_SET_PE3(base, value)  (BME_OR8(&LLWU_PE3_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_PE3(base, value)  (BME_AND8(&LLWU_PE3_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_PE3(base, value)  (BME_XOR8(&LLWU_PE3_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual LLWU_PE3 bitfields
 */

/*!
 * @name Register LLWU_PE3, field WUPE8[1:0] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE3_WUPE8 field. */
#define LLWU_RD_PE3_WUPE8(base) ((LLWU_PE3_REG(base) &
LLWU_PE3_WUPE8_MASK) >> LLWU_PE3_WUPE8_SHIFT)
#define LLWU_BRD_PE3_WUPE8(base) (BME_UBFX8(&LLWU_PE3_REG(base),
LLWU_PE3_WUPE8_SHIFT, LLWU_PE3_WUPE8_WIDTH))

/*! @brief Set the WUPE8 field to a new value. */
#define LLWU_WR_PE3_WUPE8(base, value) (LLWU_RMW_PE3(base,
LLWU_PE3_WUPE8_MASK, LLWU_PE3_WUPE8(value)))
#define LLWU_BWR_PE3_WUPE8(base, value) (BME_BFI8(&LLWU_PE3_REG(base),
((uint8_t)(value) << LLWU_PE3_WUPE8_SHIFT), LLWU_PE3_WUPE8_SHIFT,
LLWU_PE3_WUPE8_WIDTH))
/*}@*/
/*
 * @name Register LLWU_PE3, field WUPE9[3:2] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection

```

```

* - 0b11 - External input pin enabled with any change detection
*/
/*@{ */
/*! @brief Read current value of the LLWU_PE3_WUPE9 field. */
#define LLWU_RD_PE3_WUPE9(base) ((LLWU_PE3_REG(base) &
LLWU_PE3_WUPE9_MASK) >> LLWU_PE3_WUPE9_SHIFT)
#define LLWU_BRD_PE3_WUPE9(base) (BME_UBFX8(&LLWU_PE3_REG(base),
LLWU_PE3_WUPE9_SHIFT, LLWU_PE3_WUPE9_WIDTH))

/*! @brief Set the WUPE9 field to a new value. */
#define LLWU_WR_PE3_WUPE9(base, value) (LLWU_RMW_PE3(base,
LLWU_PE3_WUPE9_MASK, LLWU_PE3_WUPE9(value)))
#define LLWU_BWR_PE3_WUPE9(base, value) (BME_BFI8(&LLWU_PE3_REG(base),
((uint8_t)(value) << LLWU_PE3_WUPE9_SHIFT), LLWU_PE3_WUPE9_SHIFT,
LLWU_PE3_WUPE9_WIDTH))
/*@} */

/*!
* @name Register LLWU_PE3, field WUPE10[5:4] (RW)
*
* Enables and configures the edge detection for the wakeup pin.
*
* Values:
* - 0b00 - External input pin disabled as wakeup input
* - 0b01 - External input pin enabled with rising edge detection
* - 0b10 - External input pin enabled with falling edge detection
* - 0b11 - External input pin enabled with any change detection
*/
/*@{ */
/*! @brief Read current value of the LLWU_PE3_WUPE10 field. */
#define LLWU_RD_PE3_WUPE10(base) ((LLWU_PE3_REG(base) &
LLWU_PE3_WUPE10_MASK) >> LLWU_PE3_WUPE10_SHIFT)
#define LLWU_BRD_PE3_WUPE10(base) (BME_UBFX8(&LLWU_PE3_REG(base),
LLWU_PE3_WUPE10_SHIFT, LLWU_PE3_WUPE10_WIDTH))

/*! @brief Set the WUPE10 field to a new value. */
#define LLWU_WR_PE3_WUPE10(base, value) (LLWU_RMW_PE3(base,
LLWU_PE3_WUPE10_MASK, LLWU_PE3_WUPE10(value)))
#define LLWU_BWR_PE3_WUPE10(base, value) (BME_BFI8(&LLWU_PE3_REG(base),
((uint8_t)(value) << LLWU_PE3_WUPE10_SHIFT), LLWU_PE3_WUPE10_SHIFT,
LLWU_PE3_WUPE10_WIDTH))
/*@} */

/*!
* @name Register LLWU_PE3, field WUPE11[7:6] (RW)
*
* Enables and configures the edge detection for the wakeup pin.
*
* Values:
* - 0b00 - External input pin disabled as wakeup input
* - 0b01 - External input pin enabled with rising edge detection
* - 0b10 - External input pin enabled with falling edge detection
* - 0b11 - External input pin enabled with any change detection
*/

```

```

/*@{*/
/*! @brief Read current value of the LLWU_PE3_WUPE11 field. */
#define LLWU_RD_PE3_WUPE11(base) ((LLWU_PE3_REG(base) &
LLWU_PE3_WUPE11_MASK) >> LLWU_PE3_WUPE11_SHIFT)
#define LLWU_BRD_PE3_WUPE11(base) (BME_UBFX8(&LLWU_PE3_REG(base),
LLWU_PE3_WUPE11_SHIFT, LLWU_PE3_WUPE11_WIDTH))

/*! @brief Set the WUPE11 field to a new value. */
#define LLWU_WR_PE3_WUPE11(base, value) (LLWU_RMW_PE3(base,
LLWU_PE3_WUPE11_MASK, LLWU_PE3_WUPE11(value)))
#define LLWU_BWR_PE3_WUPE11(base, value) (BME_BFI8(&LLWU_PE3_REG(base),
((uint8_t)(value)) << LLWU_PE3_WUPE11_SHIFT), LLWU_PE3_WUPE11_SHIFT,
LLWU_PE3_WUPE11_WIDTH))
/*@}*/

/*****
 * LLWU_PE4 - LLWU Pin Enable 4 register
*****/

/*!
 * @brief LLWU_PE4 - LLWU Pin Enable 4 register (RW)
 *
 * Reset value: 0x00U
 *
 * LLWU_PE4 contains the field to enable and select the edge detect type
for the
 * external wakeup input pins LLWU_P15-LLWU_P12. This register is reset
on Chip
 * Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It
is
 * unaffected by reset types that do not trigger Chip Reset not VLLS. See
the
 * Introduction details for more information.
 */
/*!
 * @name Constants and macros for entire LLWU_PE4 register
*/
/*@*/
#define LLWU_RD_PE4(base) (LLWU_PE4_REG(base))
#define LLWU_WR_PE4(base, value) (LLWU_PE4_REG(base) = (value))
#define LLWU_RMW_PE4(base, mask, value) (LLWU_WR_PE4(base,
(LLWU_RD_PE4(base) & ~mask) | value))
#define LLWU_SET_PE4(base, value) (BME_OR8(&LLWU_PE4_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_PE4(base, value) (BME_AND8(&LLWU_PE4_REG(base),
(uint8_t)(~value)))
#define LLWU_TOG_PE4(base, value) (BME_XOR8(&LLWU_PE4_REG(base),
(uint8_t)(value)))
/*@*/
/*

```

```

 * Constants & macros for individual LLWU_PE4 bitfields
 */

/*!
 * @name Register LLWU_PE4, field WUPE12[1:0] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE4_WUPE12 field. */
#define LLWU_RD_PE4_WUPE12(base) ((LLWU_PE4_REG(base) &
LLWU_PE4_WUPE12_MASK) >> LLWU_PE4_WUPE12_SHIFT)
#define LLWU_BRD_PE4_WUPE12(base) (BME_UBFX8(&LLWU_PE4_REG(base),
LLWU_PE4_WUPE12_SHIFT, LLWU_PE4_WUPE12_WIDTH))

/*! @brief Set the WUPE12 field to a new value. */
#define LLWU_WR_PE4_WUPE12(base, value) (LLWU_RMW_PE4(base,
LLWU_PE4_WUPE12_MASK, LLWU_PE4_WUPE12(value)))
#define LLWU_BWR_PE4_WUPE12(base, value) (BME_BFI8(&LLWU_PE4_REG(base),
((uint8_t)(value) << LLWU_PE4_WUPE12_SHIFT), LLWU_PE4_WUPE12_SHIFT,
LLWU_PE4_WUPE12_WIDTH))
/*}@*/ */

/*!
 * @name Register LLWU_PE4, field WUPE13[3:2] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE4_WUPE13 field. */
#define LLWU_RD_PE4_WUPE13(base) ((LLWU_PE4_REG(base) &
LLWU_PE4_WUPE13_MASK) >> LLWU_PE4_WUPE13_SHIFT)
#define LLWU_BRD_PE4_WUPE13(base) (BME_UBFX8(&LLWU_PE4_REG(base),
LLWU_PE4_WUPE13_SHIFT, LLWU_PE4_WUPE13_WIDTH))

/*! @brief Set the WUPE13 field to a new value. */
#define LLWU_WR_PE4_WUPE13(base, value) (LLWU_RMW_PE4(base,
LLWU_PE4_WUPE13_MASK, LLWU_PE4_WUPE13(value)))
#define LLWU_BWR_PE4_WUPE13(base, value) (BME_BFI8(&LLWU_PE4_REG(base),
((uint8_t)(value) << LLWU_PE4_WUPE13_SHIFT), LLWU_PE4_WUPE13_SHIFT,
LLWU_PE4_WUPE13_WIDTH))
/*}@*/ 
```

```

/*!!
 * @name Register LLWU_PE4, field WUPE14[5:4] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE4_WUPE14 field. */
#define LLWU_RD_PE4_WUPE14(base) ((LLWU_PE4_REG(base) &
LLWU_PE4_WUPE14_MASK) >> LLWU_PE4_WUPE14_SHIFT)
#define LLWU_BRD_PE4_WUPE14(base) (BME_UBFX8(&LLWU_PE4_REG(base),
LLWU_PE4_WUPE14_SHIFT, LLWU_PE4_WUPE14_WIDTH))

/*! @brief Set the WUPE14 field to a new value. */
#define LLWU_WR_PE4_WUPE14(base, value) (LLWU_RMW_PE4(base,
LLWU_PE4_WUPE14_MASK, LLWU_PE4_WUPE14(value)))
#define LLWU_BWR_PE4_WUPE14(base, value) (BME_BFI8(&LLWU_PE4_REG(base),
((uint8_t)(value) << LLWU_PE4_WUPE14_SHIFT), LLWU_PE4_WUPE14_SHIFT,
LLWU_PE4_WUPE14_WIDTH))
/*@}*/

/*!!
 * @name Register LLWU_PE4, field WUPE15[7:6] (RW)
 *
 * Enables and configures the edge detection for the wakeup pin.
 *
 * Values:
 * - 0b00 - External input pin disabled as wakeup input
 * - 0b01 - External input pin enabled with rising edge detection
 * - 0b10 - External input pin enabled with falling edge detection
 * - 0b11 - External input pin enabled with any change detection
 */
/*@{*/
/*! @brief Read current value of the LLWU_PE4_WUPE15 field. */
#define LLWU_RD_PE4_WUPE15(base) ((LLWU_PE4_REG(base) &
LLWU_PE4_WUPE15_MASK) >> LLWU_PE4_WUPE15_SHIFT)
#define LLWU_BRD_PE4_WUPE15(base) (BME_UBFX8(&LLWU_PE4_REG(base),
LLWU_PE4_WUPE15_SHIFT, LLWU_PE4_WUPE15_WIDTH))

/*! @brief Set the WUPE15 field to a new value. */
#define LLWU_WR_PE4_WUPE15(base, value) (LLWU_RMW_PE4(base,
LLWU_PE4_WUPE15_MASK, LLWU_PE4_WUPE15(value)))
#define LLWU_BWR_PE4_WUPE15(base, value) (BME_BFI8(&LLWU_PE4_REG(base),
((uint8_t)(value) << LLWU_PE4_WUPE15_SHIFT), LLWU_PE4_WUPE15_SHIFT,
LLWU_PE4_WUPE15_WIDTH))
/*@}*/

```

```

/*
*****LLWU_ME - LLWU Module Enable register
*****/
/*!
 * @brief LLWU_ME - LLWU Module Enable register (RW)
 *
 * Reset value: 0x00U
 *
 * LLWU_ME contains the bits to enable the internal module flag as a
 * wakeup
 * input source for inputs MWUF7-MWUF0. This register is reset on Chip
 * Reset not VLLS
 * and by reset types that trigger Chip Reset not VLLS. It is unaffected
 * by
 * reset types that do not trigger Chip Reset not VLLS. See the
 * Introduction details
 * for more information.
 */
/*!
 * @name Constants and macros for entire LLWU_ME register
 */
/*@{*/
#define LLWU_RD_ME(base)          (LLWU_ME_REG(base))
#define LLWU_WR_ME(base, value)   (LLWU_ME_REG(base) = (value))
#define LLWU_RMW_ME(base, mask, value) (LLWU_WR_ME(base,
(LLWU_RD_ME(base) & ~mask) | (value)))
#define LLWU_SET_ME(base, value)  (BME_OR8(&LLWU_ME_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_ME(base, value)  (BME_AND8(&LLWU_ME_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_ME(base, value)  (BME_XOR8(&LLWU_ME_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual LLWU_ME bitfields
 */
/*!
 * @name Register LLWU_ME, field WUME0[0] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@*/
/*! @brief Read current value of the LLWU_ME_WUME0 field. */

```

```

#define LLWU_RD_ME_WUME0(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME0_MASK)
>> LLWU_ME_WUME0_SHIFT)
#define LLWU_BRD_ME_WUME0(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME0_SHIFT, LLWU_ME_WUME0_WIDTH))

/*! @brief Set the WUME0 field to a new value. */
#define LLWU_WR_ME_WUME0(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME0_MASK, LLWU_ME_WUME0(value)))
#define LLWU_BWR_ME_WUME0(base, value) (BME_BFI8(&LLWU_ME_REG(base),
((uint8_t)(value) << LLWU_ME_WUME0_SHIFT), LLWU_ME_WUME0_SHIFT,
LLWU_ME_WUME0_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_ME, field WUME1[1] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_ME_WUME1 field. */
#define LLWU_RD_ME_WUME1(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME1_MASK)
>> LLWU_ME_WUME1_SHIFT)
#define LLWU_BRD_ME_WUME1(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME1_SHIFT, LLWU_ME_WUME1_WIDTH))

/*! @brief Set the WUME1 field to a new value. */
#define LLWU_WR_ME_WUME1(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME1_MASK, LLWU_ME_WUME1(value)))
#define LLWU_BWR_ME_WUME1(base, value) (BME_BFI8(&LLWU_ME_REG(base),
((uint8_t)(value) << LLWU_ME_WUME1_SHIFT), LLWU_ME_WUME1_SHIFT,
LLWU_ME_WUME1_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_ME, field WUME2[2] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_ME_WUME2 field. */
#define LLWU_RD_ME_WUME2(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME2_MASK)
>> LLWU_ME_WUME2_SHIFT)
#define LLWU_BRD_ME_WUME2(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME2_SHIFT, LLWU_ME_WUME2_WIDTH))

/*! @brief Set the WUME2 field to a new value. */

```

```

#define LLWU_WR_ME_WUME2(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME2_MASK, LLWU_ME_WUME2(value)))
#define LLWU_BWR_ME_WUME2(base, value) (BME_BFI8(&LLWU_ME_REG(base),
((uint8_t)(value) << LLWU_ME_WUME2_SHIFT), LLWU_ME_WUME2_SHIFT,
LLWU_ME_WUME2_WIDTH))
/*@}*/

/*
 * @name Register LLWU_ME, field WUME3[3] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@*/
/*! @brief Read current value of the LLWU_ME_WUME3 field. */
#define LLWU_RD_ME_WUME3(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME3_MASK)
>> LLWU_ME_WUME3_SHIFT)
#define LLWU_BRD_ME_WUME3(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME3_SHIFT, LLWU_ME_WUME3_WIDTH))

/*! @brief Set the WUME3 field to a new value. */
#define LLWU_WR_ME_WUME3(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME3_MASK, LLWU_ME_WUME3(value)))
#define LLWU_BWR_ME_WUME3(base, value) (BME_BFI8(&LLWU_ME_REG(base),
((uint8_t)(value) << LLWU_ME_WUME3_SHIFT), LLWU_ME_WUME3_SHIFT,
LLWU_ME_WUME3_WIDTH))
/*@*/

/*
 * @name Register LLWU_ME, field WUME4[4] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@*/
/*! @brief Read current value of the LLWU_ME_WUME4 field. */
#define LLWU_RD_ME_WUME4(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME4_MASK)
>> LLWU_ME_WUME4_SHIFT)
#define LLWU_BRD_ME_WUME4(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME4_SHIFT, LLWU_ME_WUME4_WIDTH))

/*! @brief Set the WUME4 field to a new value. */
#define LLWU_WR_ME_WUME4(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME4_MASK, LLWU_ME_WUME4(value)))
#define LLWU_BWR_ME_WUME4(base, value) (BME_BFI8(&LLWU_ME_REG(base),
((uint8_t)(value) << LLWU_ME_WUME4_SHIFT), LLWU_ME_WUME4_SHIFT,
LLWU_ME_WUME4_WIDTH))
/*@*/

```

```

/*!
 * @name Register LLWU_ME, field WUME5[5] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_ME_WUME5 field. */
#define LLWU_RD_ME_WUME5(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME5_MASK) \
>> LLWU_ME_WUME5_SHIFT)
#define LLWU_BRD_ME_WUME5(base) (BME_UBFX8(&LLWU_ME_REG(base), \
LLWU_ME_WUME5_SHIFT, LLWU_ME_WUME5_WIDTH))

/*! @brief Set the WUME5 field to a new value. */
#define LLWU_WR_ME_WUME5(base, value) (LLWU_RMW_ME(base, \
LLWU_ME_WUME5_MASK, LLWU_ME_WUME5(value)))
#define LLWU_BWR_ME_WUME5(base, value) (BME_BFI8(&LLWU_ME_REG(base), \
((uint8_t)(value) << LLWU_ME_WUME5_SHIFT), LLWU_ME_WUME5_SHIFT, \
LLWU_ME_WUME5_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register LLWU_ME, field WUME6[6] (RW)
 *
 * Enables an internal module as a wakeup source input.
 *
 * Values:
 * - 0b0 - Internal module flag not used as wakeup source
 * - 0b1 - Internal module flag used as wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_ME_WUME6 field. */
#define LLWU_RD_ME_WUME6(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME6_MASK) \
>> LLWU_ME_WUME6_SHIFT)
#define LLWU_BRD_ME_WUME6(base) (BME_UBFX8(&LLWU_ME_REG(base), \
LLWU_ME_WUME6_SHIFT, LLWU_ME_WUME6_WIDTH))

/*! @brief Set the WUME6 field to a new value. */
#define LLWU_WR_ME_WUME6(base, value) (LLWU_RMW_ME(base, \
LLWU_ME_WUME6_MASK, LLWU_ME_WUME6(value)))
#define LLWU_BWR_ME_WUME6(base, value) (BME_BFI8(&LLWU_ME_REG(base), \
((uint8_t)(value) << LLWU_ME_WUME6_SHIFT), LLWU_ME_WUME6_SHIFT, \
LLWU_ME_WUME6_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register LLWU_ME, field WUME7[7] (RW)
 *
 * Enables an internal module as a wakeup source input.
 */

```

```

* Values:
* - 0b0 - Internal module flag not used as wakeup source
* - 0b1 - Internal module flag used as wakeup source
*/
/*@{/*/
/*! @brief Read current value of the LLWU_ME_WUME7 field. */
#define LLWU_RD_ME_WUME7(base) ((LLWU_ME_REG(base) & LLWU_ME_WUME7_MASK)
>> LLWU_ME_WUME7_SHIFT)
#define LLWU_BRD_ME_WUME7(base) (BME_UBFX8(&LLWU_ME_REG(base),
LLWU_ME_WUME7_SHIFT, LLWU_ME_WUME7_WIDTH))

/*! @brief Set the WUME7 field to a new value. */
#define LLWU_WR_ME_WUME7(base, value) (LLWU_RMW_ME(base,
LLWU_ME_WUME7_MASK, LLWU_ME_WUME7(value)))
#define LLWU_BWR_ME_WUME7(base, value) (BME_BFI8(&LLWU_ME_REG(base),
(uint8_t)(value) << LLWU_ME_WUME7_SHIFT), LLWU_ME_WUME7_SHIFT,
LLWU_ME_WUME7_WIDTH)
/*@{/*/

/******************
*****
* LLWU_F1 - LLWU Flag 1 register
*****
******************/

/*!!
* @brief LLWU_F1 - LLWU Flag 1 register (W1C)
*
* Reset value: 0x00U
*
* LLWU_F1 contains the wakeup flags indicating which wakeup source
caused the
* MCU to exit LLS or VLLS mode. For LLS, this is the source causing the
CPU
* interrupt flow. For VLLS, this is the source causing the MCU reset
flow. The
* external wakeup flags are read-only and clearing a flag is
accomplished by a write
* of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set,
will
* remain set if the associated WUPEx bit is cleared. This register is
reset on Chip
* Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It
is
* unaffected by reset types that do not trigger Chip Reset not VLLS. See
the
* Introduction details for more information.
*/
/*!
* @name Constants and macros for entire LLWU_F1 register
*/
/*@{/*/
#define LLWU_RD_F1(base) (LLWU_F1_REG(base))

```

```

#define LLWU_WR_F1(base, value)  (LLWU_F1_REG(base) = (value))
#define LLWU_RMW_F1(base, mask, value) (LLWU_WR_F1(base,
(LLWU_RD_F1(base) & ~mask) | (value)))
#define LLWU_SET_F1(base, value)  (BME_OR8(&LLWU_F1_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_F1(base, value)  (BME_AND8(&LLWU_F1_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_F1(base, value)  (BME_XOR8(&LLWU_F1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual LLWU_F1 bitfields
 */

/*!
 * @name Register LLWU_F1, field WUF0[0] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * a low-leakage power mode. To clear the flag write a one to WUF0.
 *
 * Values:
 * - 0b0 - LLWU_P0 input was not a wakeup source
 * - 0b1 - LLWU_P0 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF0 field. */
#define LLWU_RD_F1_WUF0(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF0_MASK) >>
LLWU_F1_WUF0_SHIFT)
#define LLWU_BRD_F1_WUF0(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF0_SHIFT, LLWU_F1_WUF0_WIDTH))

/*! @brief Set the WUF0 field to a new value. */
#define LLWU_WR_F1_WUF0(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF0_MASK | LLWU_F1_WUF1_MASK | LLWU_F1_WUF2_MASK |
LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK | LLWU_F1_WUF5_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF0(value)))
#define LLWU_BWR_F1_WUF0(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF0_SHIFT), LLWU_F1_WUF0_SHIFT,
LLWU_F1_WUF0_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_F1, field WUF1[1] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * a low-leakage power mode. To clear the flag write a one to WUF1.
 *
 * Values:
 * - 0b0 - LLWU_P1 input was not a wakeup source
 * - 0b1 - LLWU_P1 input was a wakeup source
 */

```

```

/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF1 field. */
#define LLWU_RD_F1_WUF1(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF1_MASK) >>
LLWU_F1_WUF1_SHIFT)
#define LLWU_BRD_F1_WUF1(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF1_SHIFT, LLWU_F1_WUF1_WIDTH))

/*! @brief Set the WUF1 field to a new value. */
#define LLWU_WR_F1_WUF1(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF1_MASK | LLWU_F1_WUFO_MASK | LLWU_F1_WUF2_MASK |
LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK | LLWU_F1_WUF5_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF1(value)))
#define LLWU_BWR_F1_WUF1(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF1_SHIFT), LLWU_F1_WUF1_SHIFT,
LLWU_F1_WUF1_WIDTH))
/*@}*/

/*
 * @name Register LLWU_F1, field WUF2[2] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * a low-leakage power mode. To clear the flag write a one to WUF2.
 *
 * Values:
 * - 0b0 - LLWU_P2 input was not a wakeup source
 * - 0b1 - LLWU_P2 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF2 field. */
#define LLWU_RD_F1_WUF2(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF2_MASK) >>
LLWU_F1_WUF2_SHIFT)
#define LLWU_BRD_F1_WUF2(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF2_SHIFT, LLWU_F1_WUF2_WIDTH))

/*! @brief Set the WUF2 field to a new value. */
#define LLWU_WR_F1_WUF2(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF2_MASK | LLWU_F1_WUFO_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK | LLWU_F1_WUF5_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF2(value)))
#define LLWU_BWR_F1_WUF2(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF2_SHIFT), LLWU_F1_WUF2_SHIFT,
LLWU_F1_WUF2_WIDTH))
/*@}*/

/*
 * @name Register LLWU_F1, field WUF3[3] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * a low-leakage power mode. To clear the flag write a one to WUF3.
 *
 * Values:
 * - 0b0 - LLWU_P3 input was not a wakeup source
 */

```

```

* - 0b1 - LLWU_P3 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF3 field. */
#define LLWU_RD_F1_WUF3(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF3_MASK) >>
LLWU_F1_WUF3_SHIFT)
#define LLWU_BRD_F1_WUF3(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF3_SHIFT, LLWU_F1_WUF3_WIDTH))

/*! @brief Set the WUF3 field to a new value. */
#define LLWU_WR_F1_WUF3(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF3_MASK | LLWU_F1_WUF0_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF2_MASK | LLWU_F1_WUF4_MASK | LLWU_F1_WUF5_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF3(value)))
#define LLWU_BWR_F1_WUF3(base, value) (BME_BFI8(&LLWU_F1_REG(base),
(uint8_t)(value) << LLWU_F1_WUF3_SHIFT), LLWU_F1_WUF3_SHIFT,
LLWU_F1_WUF3_WIDTH)
/*}@*/
```

```

/*!
* @name Register LLWU_F1, field WUF4[4] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
* low-leakage power mode. To clear the flag write a one to WUF4.
*
* Values:
* - 0b0 - LLWU_P4 input was not a wakeup source
* - 0b1 - LLWU_P4 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF4 field. */
#define LLWU_RD_F1_WUF4(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF4_MASK) >>
LLWU_F1_WUF4_SHIFT)
#define LLWU_BRD_F1_WUF4(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF4_SHIFT, LLWU_F1_WUF4_WIDTH))

/*! @brief Set the WUF4 field to a new value. */
#define LLWU_WR_F1_WUF4(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF4_MASK | LLWU_F1_WUF0_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF2_MASK | LLWU_F1_WUF3_MASK | LLWU_F1_WUF5_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF4(value)))
#define LLWU_BWR_F1_WUF4(base, value) (BME_BFI8(&LLWU_F1_REG(base),
(uint8_t)(value) << LLWU_F1_WUF4_SHIFT), LLWU_F1_WUF4_SHIFT,
LLWU_F1_WUF4_WIDTH)
/*}@*/
```

```

/*!
* @name Register LLWU_F1, field WUF5[5] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
* low-leakage power mode. To clear the flag write a one to WUF5.
*
```

```

* Values:
* - 0b0 - LLWU_P5 input was not a wakeup source
* - 0b1 - LLWU_P5 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF5 field. */
#define LLWU_RD_F1_WUF5(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF5_MASK) >>
LLWU_F1_WUF5_SHIFT)
#define LLWU_BRD_F1_WUF5(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF5_SHIFT, LLWU_F1_WUF5_WIDTH))

/*! @brief Set the WUF5 field to a new value. */
#define LLWU_WR_F1_WUF5(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF5_MASK | LLWU_F1_WUFO_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF2_MASK | LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK |
LLWU_F1_WUF6_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF5(value)))
#define LLWU_BWR_F1_WUF5(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF5_SHIFT), LLWU_F1_WUF5_SHIFT,
LLWU_F1_WUF5_WIDTH))
/*}@*/
```

/\*!

\* @name Register LLWU\_F1, field WUF6[6] (W1C)

\*

\* Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF6.

\*

\* Values:

\* - 0b0 - LLWU\_P6 input was not a wakeup source

\* - 0b1 - LLWU\_P6 input was a wakeup source

\*/

```
/*@{*/
/*! @brief Read current value of the LLWU_F1_WUF6 field. */
#define LLWU_RD_F1_WUF6(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF6_MASK) >>
LLWU_F1_WUF6_SHIFT)
#define LLWU_BRD_F1_WUF6(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF6_SHIFT, LLWU_F1_WUF6_WIDTH))

/*! @brief Set the WUF6 field to a new value. */
#define LLWU_WR_F1_WUF6(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF6_MASK | LLWU_F1_WUFO_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF2_MASK | LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK |
LLWU_F1_WUF5_MASK | LLWU_F1_WUF7_MASK), LLWU_F1_WUF6(value)))
#define LLWU_BWR_F1_WUF6(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF6_SHIFT), LLWU_F1_WUF6_SHIFT,
LLWU_F1_WUF6_WIDTH))
/*}@*/
```

/\*!

\* @name Register LLWU\_F1, field WUF7[7] (W1C)

\*

\* Indicates that an enabled external wakeup pin was a source of exiting a

```

* low-leakage power mode. To clear the flag write a one to WUF7.
*
* Values:
* - 0b0 - LLWU_P7 input was not a wakeup source
* - 0b1 - LLWU_P7 input was a wakeup source
*/
/*@{ */
/*! @brief Read current value of the LLWU_F1_WUF7 field. */
#define LLWU_RD_F1_WUF7(base) ((LLWU_F1_REG(base) & LLWU_F1_WUF7_MASK) >>
LLWU_F1_WUF7_SHIFT)
#define LLWU_BRD_F1_WUF7(base) (BME_UBFX8(&LLWU_F1_REG(base),
LLWU_F1_WUF7_SHIFT, LLWU_F1_WUF7_WIDTH))

/*! @brief Set the WUF7 field to a new value. */
#define LLWU_WR_F1_WUF7(base, value) (LLWU_RMW_F1(base,
(LLWU_F1_WUF7_MASK | LLWU_F1_WUF0_MASK | LLWU_F1_WUF1_MASK |
LLWU_F1_WUF2_MASK | LLWU_F1_WUF3_MASK | LLWU_F1_WUF4_MASK |
LLWU_F1_WUF5_MASK | LLWU_F1_WUF6_MASK), LLWU_F1_WUF7(value)))
#define LLWU_BWR_F1_WUF7(base, value) (BME_BFI8(&LLWU_F1_REG(base),
((uint8_t)(value) << LLWU_F1_WUF7_SHIFT), LLWU_F1_WUF7_SHIFT,
LLWU_F1_WUF7_WIDTH))
/*}@*/ */

/********************* LLWU Flag 2 register ********************
*****
* LLWU_F2 - LLWU Flag 2 register
*****
/********************* LLWU Flag 2 register ********************
****

/*!
* @brief LLWU_F2 - LLWU Flag 2 register (W1C)
*
* Reset value: 0x00U
*
* LLWU_F2 contains the wakeup flags indicating which wakeup source
caused the
* MCU to exit LLS or VLLS mode. For LLS, this is the source causing the
CPU
* interrupt flow. For VLLS, this is the source causing the MCU reset
flow. The
* external wakeup flags are read-only and clearing a flag is
accomplished by a write
* of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set,
will
* remain set if the associated WUPEx bit is cleared. This register is
reset on Chip
* Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It
is
* unaffected by reset types that do not trigger Chip Reset not VLLS. See
the
* Introduction details for more information.
*/
/*!

```

```

/* @name Constants and macros for entire LLWU_F2 register
 */
/*@{ */
#define LLWU_RD_F2(base)          (LLWU_F2_REG(base))
#define LLWU_WR_F2(base, value)   (LLWU_F2_REG(base) = (value))
#define LLWU_RMW_F2(base, mask, value) (LLWU_WR_F2(base,
(LLWU_RD_F2(base) & ~mask) | (value)))
#define LLWU_SET_F2(base, value)  (BME_OR8(&LLWU_F2_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_F2(base, value)  (BME_AND8(&LLWU_F2_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_F2(base, value)  (BME_XOR8(&LLWU_F2_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual LLWU_F2 bitfields
 */

/*!
 * @name Register LLWU_F2, field WUF8[0] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * low-leakage power mode. To clear the flag write a one to WUF8.
 *
 * Values:
 * - 0b0 - LLWU_P8 input was not a wakeup source
 * - 0b1 - LLWU_P8 input was a wakeup source
 */
/*@{ */
/*! @brief Read current value of the LLWU_F2_WUF8 field. */
#define LLWU_RD_F2_WUF8(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF8_MASK) >>
LLWU_F2_WUF8_SHIFT)
#define LLWU_BRD_F2_WUF8(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF8_SHIFT, LLWU_F2_WUF8_WIDTH))

/*! @brief Set the WUF8 field to a new value. */
#define LLWU_WR_F2_WUF8(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK | LLWU_F2_WUF10_MASK |
LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK | LLWU_F2_WUF13_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF8(value)))
#define LLWU_BWR_F2_WUF8(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF8_SHIFT), LLWU_F2_WUF8_SHIFT,
LLWU_F2_WUF8_WIDTH))
/*@} */

/*!
 * @name Register LLWU_F2, field WUF9[1] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
 * low-leakage power mode. To clear the flag write a one to WUF9.
 */

```

```

* Values:
* - 0b0 - LLWU_P9 input was not a wakeup source
* - 0b1 - LLWU_P9 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F2_WUF9 field. */
#define LLWU_RD_F2_WUF9(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF9_MASK) >>
LLWU_F2_WUF9_SHIFT)
#define LLWU_BRD_F2_WUF9(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF9_SHIFT, LLWU_F2_WUF9_WIDTH))

/*! @brief Set the WUF9 field to a new value. */
#define LLWU_WR_F2_WUF9(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF9_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF10_MASK |
LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK | LLWU_F2_WUF13_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF9(value)))
#define LLWU_BWR_F2_WUF9(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF9_SHIFT), LLWU_F2_WUF9_SHIFT,
LLWU_F2_WUF9_WIDTH))
/*@}*/

/*!
* @name Register LLWU_F2, field WUF10[2] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
* a low-leakage power mode. To clear the flag write a one to WUF10.
*
* Values:
* - 0b0 - LLWU_P10 input was not a wakeup source
* - 0b1 - LLWU_P10 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F2_WUF10 field. */
#define LLWU_RD_F2_WUF10(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF10_MASK)
>> LLWU_F2_WUF10_SHIFT)
#define LLWU_BRD_F2_WUF10(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF10_SHIFT, LLWU_F2_WUF10_WIDTH))

/*! @brief Set the WUF10 field to a new value. */
#define LLWU_WR_F2_WUF10(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF10_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK | LLWU_F2_WUF13_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF10(value)))
#define LLWU_BWR_F2_WUF10(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF10_SHIFT), LLWU_F2_WUF10_SHIFT,
LLWU_F2_WUF10_WIDTH))
/*@}*/

/*!
* @name Register LLWU_F2, field WUF11[3] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
* a

```

```

* low-leakage power mode. To clear the flag write a one to WUF11.
*
* Values:
* - 0b0 - LLWU_P11 input was not a wakeup source
* - 0b1 - LLWU_P11 input was a wakeup source
*/
/*@{ */
/*! @brief Read current value of the LLWU_F2_WUF11 field. */
#define LLWU_RD_F2_WUF11(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF11_MASK)
>> LLWU_F2_WUF11_SHIFT)
#define LLWU_BRD_F2_WUF11(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF11_SHIFT, LLWU_F2_WUF11_WIDTH))

/*! @brief Set the WUF11 field to a new value. */
#define LLWU_WR_F2_WUF11(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF11_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF10_MASK | LLWU_F2_WUF12_MASK | LLWU_F2_WUF13_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF11(value)))
#define LLWU_BWR_F2_WUF11(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF11_SHIFT), LLWU_F2_WUF11_SHIFT,
LLWU_F2_WUF11_WIDTH))
/*}@*/ */

/*!
* @name Register LLWU_F2, field WUF12[4] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
a
* low-leakage power mode. To clear the flag write a one to WUF12.
*
* Values:
* - 0b0 - LLWU_P12 input was not a wakeup source
* - 0b1 - LLWU_P12 input was a wakeup source
*/
/*@{ */
/*! @brief Read current value of the LLWU_F2_WUF12 field. */
#define LLWU_RD_F2_WUF12(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF12_MASK)
>> LLWU_F2_WUF12_SHIFT)
#define LLWU_BRD_F2_WUF12(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF12_SHIFT, LLWU_F2_WUF12_WIDTH))

/*! @brief Set the WUF12 field to a new value. */
#define LLWU_WR_F2_WUF12(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF12_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF10_MASK | LLWU_F2_WUF11_MASK | LLWU_F2_WUF13_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF12(value)))
#define LLWU_BWR_F2_WUF12(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF12_SHIFT), LLWU_F2_WUF12_SHIFT,
LLWU_F2_WUF12_WIDTH))
/*}@*/ */

/*!
* @name Register LLWU_F2, field WUF13[5] (W1C)
*

```

```

 * Indicates that an enabled external wakeup pin was a source of exiting
a
 * low-leakage power mode. To clear the flag write a one to WUF13.
 *
 * Values:
 * - 0b0 - LLWU_P13 input was not a wakeup source
 * - 0b1 - LLWU_P13 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F2_WUF13 field. */
#define LLWU_RD_F2_WUF13(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF13_MASK)
>> LLWU_F2_WUF13_SHIFT)
#define LLWU_BRD_F2_WUF13(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF13_SHIFT, LLWU_F2_WUF13_WIDTH))

/*! @brief Set the WUF13 field to a new value. */
#define LLWU_WR_F2_WUF13(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF13_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF10_MASK | LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK |
LLWU_F2_WUF14_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF13(value)))
#define LLWU_BWR_F2_WUF13(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF13_SHIFT), LLWU_F2_WUF13_SHIFT,
LLWU_F2_WUF13_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register LLWU_F2, field WUF14[6] (W1C)
 *
 * Indicates that an enabled external wakeup pin was a source of exiting
a
 * low-leakage power mode. To clear the flag write a one to WUF14.
 *
 * Values:
 * - 0b0 - LLWU_P14 input was not a wakeup source
 * - 0b1 - LLWU_P14 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F2_WUF14 field. */
#define LLWU_RD_F2_WUF14(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF14_MASK)
>> LLWU_F2_WUF14_SHIFT)
#define LLWU_BRD_F2_WUF14(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF14_SHIFT, LLWU_F2_WUF14_WIDTH))

/*! @brief Set the WUF14 field to a new value. */
#define LLWU_WR_F2_WUF14(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF14_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF10_MASK | LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK |
LLWU_F2_WUF13_MASK | LLWU_F2_WUF15_MASK), LLWU_F2_WUF14(value)))
#define LLWU_BWR_F2_WUF14(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF14_SHIFT), LLWU_F2_WUF14_SHIFT,
LLWU_F2_WUF14_WIDTH))
/*}@*/
```

  

```

/*!
```

```

* @name Register LLWU_F2, field WUF15[7] (W1C)
*
* Indicates that an enabled external wakeup pin was a source of exiting
a
* low-leakage power mode. To clear the flag write a one to WUF15.
*
* Values:
* - 0b0 - LLWU_P15 input was not a wakeup source
* - 0b1 - LLWU_P15 input was a wakeup source
*/
/*@{*/
/*! @brief Read current value of the LLWU_F2_WUF15 field. */
#define LLWU_RD_F2_WUF15(base) ((LLWU_F2_REG(base) & LLWU_F2_WUF15_MASK)
>> LLWU_F2_WUF15_SHIFT)
#define LLWU_BRD_F2_WUF15(base) (BME_UBFX8(&LLWU_F2_REG(base),
LLWU_F2_WUF15_SHIFT, LLWU_F2_WUF15_WIDTH))

/*! @brief Set the WUF15 field to a new value. */
#define LLWU_WR_F2_WUF15(base, value) (LLWU_RMW_F2(base,
(LLWU_F2_WUF15_MASK | LLWU_F2_WUF8_MASK | LLWU_F2_WUF9_MASK |
LLWU_F2_WUF10_MASK | LLWU_F2_WUF11_MASK | LLWU_F2_WUF12_MASK |
LLWU_F2_WUF13_MASK | LLWU_F2_WUF14_MASK), LLWU_F2_WUF15(value)))
#define LLWU_BWR_F2_WUF15(base, value) (BME_BFI8(&LLWU_F2_REG(base),
((uint8_t)(value) << LLWU_F2_WUF15_SHIFT), LLWU_F2_WUF15_SHIFT,
LLWU_F2_WUF15_WIDTH))
/*@}*/

/********************* */
***** */
* LLWU_F3 - LLWU Flag 3 register
***** */

/*! !
* @brief LLWU_F3 - LLWU Flag 3 register (RO)
*
* Reset value: 0x00U
*
* LLWU_F3 contains the wakeup flags indicating which internal wakeup
source
* caused the MCU to exit LLS or VLLS mode. For LLS, this is the source
causing the
* CPU interrupt flow. For VLLS, this is the source causing the MCU reset
flow.
* For internal peripherals that are capable of running in a low-leakage
power
* mode, such as iRTC or CMP modules, the flag from the associated
peripheral is
* accessible as the MWUFx bit. The flag will need to be cleared in the
peripheral
* instead of writing a 1 to the MWUFx bit. This register is reset on
Chip Reset

```

```

    * not VLLS and by reset types that trigger Chip Reset not VLLS. It is
unaffected
    * by reset types that do not trigger Chip Reset not VLLS. See the
Introduction
    * details for more information.
*/
/*!
 * @name Constants and macros for entire LLWU_F3 register
 */
/*@{*/
#define LLWU_RD_F3(base)          (LLWU_F3_REG(base))
/*}@*/



/*
 * Constants & macros for individual LLWU_F3 bitfields
 */

/*!
 * @name Register LLWU_F3, field MWUF0[0] (RO)
 *
 * Indicates that an enabled internal peripheral was a source of exiting
a
 * low-leakage power mode. To clear the flag, follow the internal
peripheral flag
 * clearing mechanism.
 *
 * Values:
 * - 0b0 - Module 0 input was not a wakeup source
 * - 0b1 - Module 0 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF0 field. */
#define LLWU_RD_F3_MWUF0(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF0_MASK)
>> LLWU_F3_MWUF0_SHIFT)
#define LLWU_BRD_F3_MWUF0(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF0_SHIFT, LLWU_F3_MWUF0_WIDTH))
/*}@*/



/*!
 * @name Register LLWU_F3, field MWUF1[1] (RO)
 *
 * Indicates that an enabled internal peripheral was a source of exiting
a
 * low-leakage power mode. To clear the flag, follow the internal
peripheral flag
 * clearing mechanism.
 *
 * Values:
 * - 0b0 - Module 1 input was not a wakeup source
 * - 0b1 - Module 1 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF1 field. */

```

```

#define LLWU_RD_F3_MWUF1(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF1_MASK)
>> LLWU_F3_MWUF1_SHIFT)
#define LLWU_BRD_F3_MWUF1(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF1_SHIFT, LLWU_F3_MWUF1_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_F3, field MWUF2[2] (RO)
 *
 * Indicates that an enabled internal peripheral was a source of exiting
 * a low-leakage power mode. To clear the flag, follow the internal
 * peripheral flag clearing mechanism.
 *
 * Values:
 * - 0b0 - Module 2 input was not a wakeup source
 * - 0b1 - Module 2 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF2 field. */
#define LLWU_RD_F3_MWUF2(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF2_MASK)
>> LLWU_F3_MWUF2_SHIFT)
#define LLWU_BRD_F3_MWUF2(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF2_SHIFT, LLWU_F3_MWUF2_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_F3, field MWUF3[3] (RO)
 *
 * Indicates that an enabled internal peripheral was a source of exiting
 * a low-leakage power mode. To clear the flag, follow the internal
 * peripheral flag clearing mechanism.
 *
 * Values:
 * - 0b0 - Module 3 input was not a wakeup source
 * - 0b1 - Module 3 input was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF3 field. */
#define LLWU_RD_F3_MWUF3(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF3_MASK)
>> LLWU_F3_MWUF3_SHIFT)
#define LLWU_BRD_F3_MWUF3(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF3_SHIFT, LLWU_F3_MWUF3_WIDTH))
/*@}*/

/*!
 * @name Register LLWU_F3, field MWUF4[4] (RO)
 *
 * Indicates that an enabled internal peripheral was a source of exiting
 * a

```

```

    * low-leakage power mode. To clear the flag, follow the internal
    peripheral flag
    * clearing mechanism.
    *
    * Values:
    * - 0b0 - Module 4 input was not a wakeup source
    * - 0b1 - Module 4 input was a wakeup source
    */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF4 field. */
#define LLWU_RD_F3_MWUF4(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF4_MASK)
>> LLWU_F3_MWUF4_SHIFT)
#define LLWU_BRD_F3_MWUF4(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF4_SHIFT, LLWU_F3_MWUF4_WIDTH))
/*@}*/
/*!
    * @name Register LLWU_F3, field MWUF5[5] (RO)
    *
    * Indicates that an enabled internal peripheral was a source of exiting
    a
    * low-leakage power mode. To clear the flag, follow the internal
    peripheral flag
    * clearing mechanism.
    *
    * Values:
    * - 0b0 - Module 5 input was not a wakeup source
    * - 0b1 - Module 5 input was a wakeup source
    */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF5 field. */
#define LLWU_RD_F3_MWUF5(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF5_MASK)
>> LLWU_F3_MWUF5_SHIFT)
#define LLWU_BRD_F3_MWUF5(base) (BME_UBFX8(&LLWU_F3_REG(base),
LLWU_F3_MWUF5_SHIFT, LLWU_F3_MWUF5_WIDTH))
/*@}*/
/*!
    * @name Register LLWU_F3, field MWUF6[6] (RO)
    *
    * Indicates that an enabled internal peripheral was a source of exiting
    a
    * low-leakage power mode. To clear the flag, follow the internal
    peripheral flag
    * clearing mechanism.
    *
    * Values:
    * - 0b0 - Module 6 input was not a wakeup source
    * - 0b1 - Module 6 input was a wakeup source
    */
/*@{*/
/*! @brief Read current value of the LLWU_F3_MWUF6 field. */
#define LLWU_RD_F3_MWUF6(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF6_MASK)
>> LLWU_F3_MWUF6_SHIFT)

```

```

#define LLWU_BRD_F3_MWUF6(base) (BME_UBFX8(&LLWU_F3_REG(base),  

LLWU_F3_MWUF6_SHIFT, LLWU_F3_MWUF6_WIDTH))  

/*@}*/

/*!  

 * @name Register LLWU_F3, field MWUF7[7] (RO)  

 *  

 * Indicates that an enabled internal peripheral was a source of exiting  

 * a  

 * low-leakage power mode. To clear the flag, follow the internal  

 * peripheral flag  

 * clearing mechanism.  

 *  

 * Values:  

 * - 0b0 - Module 7 input was not a wakeup source  

 * - 0b1 - Module 7 input was a wakeup source  

 */  

/*@{*/  

/*! @brief Read current value of the LLWU_F3_MWUF7 field. */  

#define LLWU_RD_F3_MWUF7(base) ((LLWU_F3_REG(base) & LLWU_F3_MWUF7_MASK)  

>> LLWU_F3_MWUF7_SHIFT)  

#define LLWU_BRD_F3_MWUF7(base) (BME_UBFX8(&LLWU_F3_REG(base),  

LLWU_F3_MWUF7_SHIFT, LLWU_F3_MWUF7_WIDTH))  

/*@}/

*****  

* LLWU_FILT1 - LLWU Pin Filter 1 register  

*****  

/*!  

 * @brief LLWU_FILT1 - LLWU Pin Filter 1 register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * LLWU_FILT1 is a control and status register that is used to  

enable/disable  

* the digital filter 1 features for an external pin. This register is  

reset on  

* Chip Reset not VLLS and by reset types that trigger Chip Reset not  

VLLS. It is  

* unaffected by reset types that do not trigger Chip Reset not VLLS. See  

the  

* Introduction details for more information.  

*/  

/*!  

 * @name Constants and macros for entire LLWU_FILT1 register  

*/  

/*@{*/  

#define LLWU_RD_FILT1(base) (LLWU_FILT1_REG(base))  

#define LLWU_WR_FILT1(base, value) (LLWU_FILT1_REG(base) = (value))

```

```

#define LLWU_RMW_FILT1(base, mask, value) (LLWU_WR_FILT1(base,
(LLWU_RD_FILT1(base) & ~(mask)) | (value)))
#define LLWU_SET_FILT1(base, value) (BME_OR8(&LLWU_FILT1_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_FILT1(base, value) (BME_AND8(&LLWU_FILT1_REG(base),
(uint8_t)(~(value))))
#define LLWU_TOG_FILT1(base, value) (BME_XOR8(&LLWU_FILT1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual LLWU_FILT1 bitfields
 */

/*!!
 * @name Register LLWU_FILT1, field FILTSEL[3:0] (RW)
 *
 * Selects 1 out of the 16 wakeup pins to be muxed into the filter.
 *
 * Values:
 * - 0b0000 - Select LLWU_P0 for filter
 * - 0b1111 - Select LLWU_P15 for filter
 */
/*@{*/
/*!! @brief Read current value of the LLWU_FILT1_FILTSEL field. */
#define LLWU_RD_FILT1_FILTSEL(base) ((LLWU_FILT1_REG(base) &
LLWU_FILT1_FILTSEL_MASK) >> LLWU_FILT1_FILTSEL_SHIFT)
#define LLWU_BRD_FILT1_FILTSEL(base) (BME_UBFX8(&LLWU_FILT1_REG(base),
LLWU_FILT1_FILTSEL_SHIFT, LLWU_FILT1_FILTSEL_WIDTH))

/*!! @brief Set the FILTSEL field to a new value. */
#define LLWU_WR_FILT1_FILTSEL(base, value) (LLWU_RMW_FILT1(base,
(LLWU_FILT1_FILTSEL_MASK | LLWU_FILT1_FILTSEL_MASK),
LLWU_FILT1_FILTSEL(value)))
#define LLWU_BWR_FILT1_FILTSEL(base, value)
(BME_BFI8(&LLWU_FILT1_REG(base), ((uint8_t)(value) <<
LLWU_FILT1_FILTSEL_SHIFT), LLWU_FILT1_FILTSEL_SHIFT,
LLWU_FILT1_FILTSEL_WIDTH))
/*@}*/

/*!!
 * @name Register LLWU_FILT1, field FILTE[6:5] (RW)
 *
 * Controls the digital filter options for the external pin detect.
 *
 * Values:
 * - 0b00 - Filter disabled
 * - 0b01 - Filter posedge detect enabled
 * - 0b10 - Filter negedge detect enabled
 * - 0b11 - Filter any edge detect enabled
 */
/*@{*/
/*!! @brief Read current value of the LLWU_FILT1_FILTE field. */

```

```

#define LLWU_RD_FILT1_FILTE(base) ((LLWU_FILT1_REG(base) &
LLWU_FILT1_FILTE_MASK) >> LLWU_FILT1_FILTE_SHIFT)
#define LLWU_BRD_FILT1_FILTE(base) (BME_UBFX8(&LLWU_FILT1_REG(base),
LLWU_FILT1_FILTE_SHIFT, LLWU_FILT1_FILTE_WIDTH))

/*! @brief Set the FILTE field to a new value. */
#define LLWU_WR_FILT1_FILTE(base, value) (LLWU_RMW_FILT1(base,
(LLWU_FILT1_FILTE_MASK | LLWU_FILT1_FILTF_MASK),
LLWU_FILT1_FILTE(value)))
#define LLWU_BWR_FILT1_FILTE(base, value)
(BME_BFI8(&LLWU_FILT1_REG(base), ((uint8_t)(value) <<
LLWU_FILT1_FILTE_SHIFT), LLWU_FILT1_FILTE_SHIFT, LLWU_FILT1_FILTE_WIDTH))
/*@}*/

/*
 * @name Register LLWU_FILT1, field FILTF[7] (W1C)
 *
 * Indicates that the filtered external wakeup pin, selected by FILTSEL,
was a
 * source of exiting a low-leakage power mode. To clear the flag write a
one to
 * FILTF.
 *
 * Values:
 * - 0b0 - Pin Filter 1 was not a wakeup source
 * - 0b1 - Pin Filter 1 was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_FILT1_FILTF field. */
#define LLWU_RD_FILT1_FILTF(base) ((LLWU_FILT1_REG(base) &
LLWU_FILT1_FILTF_MASK) >> LLWU_FILT1_FILTF_SHIFT)
#define LLWU_BRD_FILT1_FILTF(base) (BME_UBFX8(&LLWU_FILT1_REG(base),
LLWU_FILT1_FILTF_SHIFT, LLWU_FILT1_FILTF_WIDTH))

/*! @brief Set the FILTF field to a new value. */
#define LLWU_WR_FILT1_FILTF(base, value) (LLWU_RMW_FILT1(base,
LLWU_FILT1_FILTF_MASK, LLWU_FILT1_FILTF(value)))
#define LLWU_BWR_FILT1_FILTF(base, value)
(BME_BFI8(&LLWU_FILT1_REG(base), ((uint8_t)(value) <<
LLWU_FILT1_FILTF_SHIFT), LLWU_FILT1_FILTF_SHIFT, LLWU_FILT1_FILTF_WIDTH))
/*@}*/

*****
*****  

* LLWU_FILT2 - LLWU Pin Filter 2 register  

*****  

*****  

/*!
 * @brief LLWU_FILT2 - LLWU Pin Filter 2 register (RW)
 *
 * Reset value: 0x00U
 *

```

```

    * LLWU_FILT2 is a control and status register that is used to
enable/disable
    * the digital filter 2 features for an external pin. This register is
reset on
    * Chip Reset not VLLS and by reset types that trigger Chip Reset not
VLLS. It is
    * unaffected by reset types that do not trigger Chip Reset not VLLS. See
the
    * Introduction details for more information.
*/
/*!
 * @name Constants and macros for entire LLWU_FILT2 register
 */
/*@{*/
#define LLWU_RD_FILT2(base)      (LLWU_FILT2_REG(base))
#define LLWU_WR_FILT2(base, value) (LLWU_FILT2_REG(base) = (value))
#define LLWU_RMW_FILT2(base, mask, value) (LLWU_WR_FILT2(base,
(LLWU_RD_FILT2(base) & ~mask)) | (value)))
#define LLWU_SET_FILT2(base, value) (BME_OR8(&LLWU_FILT2_REG(base),
(uint8_t)(value)))
#define LLWU_CLR_FILT2(base, value) (BME_AND8(&LLWU_FILT2_REG(base),
(uint8_t)(~value))))
#define LLWU_TOG_FILT2(base, value) (BME_XOR8(&LLWU_FILT2_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual LLWU_FILT2 bitfields
 */
/*!
 * @name Register LLWU_FILT2, field FILTSEL[3:0] (RW)
 *
 * Selects 1 out of the 16 wakeup pins to be muxed into the filter.
 *
 * Values:
 * - 0b0000 - Select LLWU_P0 for filter
 * - 0b1111 - Select LLWU_P15 for filter
 */
/*@{*/
/*! @brief Read current value of the LLWU_FILT2_FILTSEL field. */
#define LLWU_RD_FILT2_FILTSEL(base) ((LLWU_FILT2_REG(base) &
LLWU_FILT2_FILTSEL_MASK) >> LLWU_FILT2_FILTSEL_SHIFT)
#define LLWU_BRD_FILT2_FILTSEL(base) (BME_UBFX8(&LLWU_FILT2_REG(base),
LLWU_FILT2_FILTSEL_SHIFT, LLWU_FILT2_FILTSEL_WIDTH))

/*! @brief Set the FILTSEL field to a new value. */
#define LLWU_WR_FILT2_FILTSEL(base, value) (LLWU_RMW_FILT2(base,
(LLWU_FILT2_FILTSEL_MASK | LLWU_FILT2_FILTSEL(value)),
LLWU_FILT2_FILTSEL(value)))
#define LLWU_BWR_FILT2_FILTSEL(base, value)
(BME_BFI8(&LLWU_FILT2_REG(base), ((uint8_t)(value) <<
LLWU_FILT2_FILTSEL_SHIFT), LLWU_FILT2_FILTSEL_SHIFT,
LLWU_FILT2_FILTSEL_WIDTH))

```

```

/*@}*/

/*
 * @name Register LLWU_FILT2, field FILTE[6:5] (RW)
 *
 * Controls the digital filter options for the external pin detect.
 *
 * Values:
 * - 0b00 - Filter disabled
 * - 0b01 - Filter posedge detect enabled
 * - 0b10 - Filter negedge detect enabled
 * - 0b11 - Filter any edge detect enabled
 */
/*@{*/
/*! @brief Read current value of the LLWU_FILT2_FILTE field. */
#define LLWU_RD_FILT2_FILTE(base) ((LLWU_FILT2_REG(base) &
LLWU_FILT2_FILTE_MASK) >> LLWU_FILT2_FILTE_SHIFT)
#define LLWU_BRD_FILT2_FILTE(base) (BME_UBFX8(&LLWU_FILT2_REG(base),
LLWU_FILT2_FILTE_SHIFT, LLWU_FILT2_FILTE_WIDTH))

/*! @brief Set the FILTE field to a new value. */
#define LLWU_WR_FILT2_FILTE(base, value) (LLWU_RMW_FILT2(base,
(LLWU_FILT2_FILTE_MASK | LLWU_FILT2_FILTF_MASK),
LLWU_FILT2_FILTE(value)))
#define LLWU_BWR_FILT2_FILTE(base, value)
(BME_BFI8(&LLWU_FILT2_REG(base), ((uint8_t)(value) <<
LLWU_FILT2_FILTE_SHIFT), LLWU_FILT2_FILTE_SHIFT, LLWU_FILT2_FILTE_WIDTH))
/*@*/ */

/*
 * @name Register LLWU_FILT2, field FILTF[7] (W1C)
 *
 * Indicates that the filtered external wakeup pin, selected by FILTSEL,
was a
 * source of exiting a low-leakage power mode. To clear the flag write a
one to
 * FILTF.
 *
 * Values:
 * - 0b0 - Pin Filter 2 was not a wakeup source
 * - 0b1 - Pin Filter 2 was a wakeup source
 */
/*@{*/
/*! @brief Read current value of the LLWU_FILT2_FILTF field. */
#define LLWU_RD_FILT2_FILTF(base) ((LLWU_FILT2_REG(base) &
LLWU_FILT2_FILTF_MASK) >> LLWU_FILT2_FILTF_SHIFT)
#define LLWU_BRD_FILT2_FILTF(base) (BME_UBFX8(&LLWU_FILT2_REG(base),
LLWU_FILT2_FILTF_SHIFT, LLWU_FILT2_FILTF_WIDTH))

/*! @brief Set the FILTF field to a new value. */
#define LLWU_WR_FILT2_FILTF(base, value) (LLWU_RMW_FILT2(base,
LLWU_FILT2_FILTF_MASK, LLWU_FILT2_FILTF(value)))

```

```

#define LLWU_BWR_FILT2_FILTF(base, value)
(BME_BFI8(&LLWU_FILT2_REG(base), ((uint8_t)(value) <<
LLWU_FILT2_FILTF_SHIFT), LLWU_FILT2_FILTF_SHIFT, LLWU_FILT2_FILTF_WIDTH))
/*@}*/

/*
 * MKL25Z4 LPTMR
 *
 * Low Power Timer
 *
 * Registers defined in this header file:
 * - LPTMR_CSR - Low Power Timer Control Status Register
 * - LPTMR_PSR - Low Power Timer Prescale Register
 * - LPTMR_CMR - Low Power Timer Compare Register
 * - LPTMR_CNR - Low Power Timer Counter Register
 */

```

```

#define LPTMR_INSTANCE_COUNT (1U) /*!< Number of instances of the LPTMR
module. */
#define LPTMR0_IDX (0U) /*!< Instance number for LPTMR0. */

/***** LPTMR_CSR - Low Power Timer Control Status Register *****/
***** /
```

```

/*!
 * @brief LPTMR_CSR - Low Power Timer Control Status Register (RW)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire LPTMR_CSR register
 */
/*@{ */

#define LPTMR_RD_CSR(base) (LPTMR_CSR_REG(base))
#define LPTMR_WR_CSR(base, value) (LPTMR_CSR_REG(base) = (value))
#define LPTMR_RMW_CSR(base, mask, value) (LPTMR_WR_CSR(base,
(LPTMR_RD_CSR(base) & ~mask) | value)))
#define LPTMR_SET_CSR(base, value) (BME_OR32(&LPTMR_CSR_REG(base),
(uint32_t)(value)))
#define LPTMR_CLR_CSR(base, value) (BME_AND32(&LPTMR_CSR_REG(base),
(uint32_t)(~value))))
#define LPTMR_TOG_CSR(base, value) (BME_XOR32(&LPTMR_CSR_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual LPTMR_CSR bitfields
 */
/*!
```

```

* @name Register LPTMR_CSR, field TEN[0] (RW)
*
* When TEN is clear, it resets the LPTMR internal logic, including the
CNR and
* TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this
field,
* CSR[5:1] must not be altered.
*
* Values:
* - 0b0 - LPTMR is disabled and internal logic is reset.
* - 0b1 - LPTMR is enabled.
*/
/*@{*/
/*! @brief Read current value of the LPTMR_CSR_TEN field. */
#define LPTMR_RD_CSR_TEN(base) ((LPTMR_CSR_REG(base) &
LPTMR_CSR_TEN_MASK) >> LPTMR_CSR_TEN_SHIFT)
#define LPTMR_BRD_CSR_TEN(base) (BME_UBFX32(&LPTMR_CSR_REG(base),
LPTMR_CSR_TEN_SHIFT, LPTMR_CSR_TEN_WIDTH))

/*! @brief Set the TEN field to a new value. */
#define LPTMR_WR_CSR_TEN(base, value) (LPTMR_RMW_CSR(base,
(LPTMR_CSR_TEN_MASK | LPTMR_CSR_TCF_MASK), LPTMR_CSR_TEN(value)))
#define LPTMR_BWR_CSR_TEN(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),
((uint32_t)(value) << LPTMR_CSR_TEN_SHIFT), LPTMR_CSR_TEN_SHIFT,
LPTMR_CSR_TEN_WIDTH))
/*@}*/

/*
* @name Register LPTMR_CSR, field TMS[1] (RW)
*
* Configures the mode of the LPTMR. TMS must be altered only when the
LPTMR is
* disabled.
*
* Values:
* - 0b0 - Time Counter mode.
* - 0b1 - Pulse Counter mode.
*/
/*@{*/
/*! @brief Read current value of the LPTMR_CSR_TMS field. */
#define LPTMR_RD_CSR_TMS(base) ((LPTMR_CSR_REG(base) &
LPTMR_CSR_TMS_MASK) >> LPTMR_CSR_TMS_SHIFT)
#define LPTMR_BRD_CSR_TMS(base) (BME_UBFX32(&LPTMR_CSR_REG(base),
LPTMR_CSR_TMS_SHIFT, LPTMR_CSR_TMS_WIDTH))

/*! @brief Set the TMS field to a new value. */
#define LPTMR_WR_CSR_TMS(base, value) (LPTMR_RMW_CSR(base,
(LPTMR_CSR_TMS_MASK | LPTMR_CSR_TCF_MASK), LPTMR_CSR_TMS(value)))
#define LPTMR_BWR_CSR_TMS(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),
((uint32_t)(value) << LPTMR_CSR_TMS_SHIFT), LPTMR_CSR_TMS_SHIFT,
LPTMR_CSR_TMS_WIDTH))
/*@}*/

/*

```

```

* @name Register LPTMR_CSR, field TFC[2] (RW)
*
* When clear, TFC configures the CNR to reset whenever TCF is set. When
set,
* TFC configures the CNR to reset on overflow. TFC must be altered only
when the
* LPTMR is disabled.
*
* Values:
* - 0b0 - CNR is reset whenever TCF is set.
* - 0b1 - CNR is reset on overflow.
*/
/*@{*/
/*! @brief Read current value of the LPTMR_CSR_TFC field. */
#define LPTMR_RD_CSR_TFC(base) ((LPTMR_CSR_REG(base) &
LPTMR_CSR_TFC_MASK) >> LPTMR_CSR_TFC_SHIFT)
#define LPTMR_BRD_CSR_TFC(base) (BME_UBFX32(&LPTMR_CSR_REG(base),
LPTMR_CSR_TFC_SHIFT, LPTMR_CSR_TFC_WIDTH))

/*! @brief Set the TFC field to a new value. */
#define LPTMR_WR_CSR_TFC(base, value) (LPTMR_RMW_CSR(base,
(LPTMR_CSR_TFC_MASK | LPTMR_CSR_TCF_MASK), LPTMR_CSR_TFC(value)))
#define LPTMR_BWR_CSR_TFC(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),
((uint32_t)(value) << LPTMR_CSR_TFC_SHIFT), LPTMR_CSR_TFC_SHIFT,
LPTMR_CSR_TFC_WIDTH))
/*}@*/
```

```

/*!
* @name Register LPTMR_CSR, field TPP[3] (RW)
*
* Configures the polarity of the input source in Pulse Counter mode. TPP
must
* be changed only when the LPTMR is disabled.
*
* Values:
* - 0b0 - Pulse Counter input source is active-high, and the CNR will
increment
*   on the rising-edge.
* - 0b1 - Pulse Counter input source is active-low, and the CNR will
increment
*   on the falling-edge.
*/
/*@{*/
/*! @brief Read current value of the LPTMR_CSR TPP field. */
#define LPTMR_RD_CSR TPP(base) ((LPTMR_CSR_REG(base) &
LPTMR_CSR TPP_MASK) >> LPTMR_CSR TPP SHIFT)
#define LPTMR_BRD_CSR TPP(base) (BME_UBFX32(&LPTMR_CSR_REG(base),
LPTMR_CSR TPP SHIFT, LPTMR_CSR TPP WIDTH))

/*! @brief Set the TPP field to a new value. */
#define LPTMR_WR_CSR TPP(base, value) (LPTMR_RMW_CSR(base,
(LPTMR_CSR TPP MASK | LPTMR_CSR TCF MASK), LPTMR_CSR TPP(value)))
```

```

#define LPTMR_BWR_CSR TPP(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),  

((uint32_t)(value) << LPTMR_CSR_TPP_SHIFT), LPTMR_CSR_TPP_SHIFT,  

LPTMR_CSR_TPP_WIDTH))  

/*@}*/

/*!  

 * @name Register LPTMR_CSR, field TPS[5:4] (RW)  

 *  

 * Configures the input source to be used in Pulse Counter mode. TPS must  

be  

* altered only when the LPTMR is disabled. The input connections vary by  

device.  

* See the chip configuration details for information on the connections  

to these  

* inputs.  

*  

* Values:  

* - 0b00 - Pulse counter input 0 is selected.  

* - 0b01 - Pulse counter input 1 is selected.  

* - 0b10 - Pulse counter input 2 is selected.  

* - 0b11 - Pulse counter input 3 is selected.  

*/  

/*@{*/  

/*! @brief Read current value of the LPTMR_CSR_TPS field. */  

#define LPTMR_RD_CSR_TPS(base) ((LPTMR_CSR_REG(base) &  

LPTMR_CSR_TPS_MASK) >> LPTMR_CSR_TPS_SHIFT)  

#define LPTMR_BRD_CSR_TPS(base) (BME_UBFX32(&LPTMR_CSR_REG(base),  

LPTMR_CSR_TPS_SHIFT, LPTMR_CSR_TPS_WIDTH))

/*! @brief Set the TPS field to a new value. */  

#define LPTMR_WR_CSR_TPS(base, value) (LPTMR_RMW_CSR(base,  

(LPTMR_CSR_TPS_MASK | LPTMR_CSR_TCF_MASK), LPTMR_CSR_TPS(value)))  

#define LPTMR_BWR_CSR_TPS(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),  

((uint32_t)(value) << LPTMR_CSR_TPS_SHIFT), LPTMR_CSR_TPS_SHIFT,  

LPTMR_CSR_TPS_WIDTH))  

/*@}*/

/*!  

 * @name Register LPTMR_CSR, field TIE[6] (RW)  

 *  

 * When TIE is set, the LPTMR Interrupt is generated whenever TCF is also  

set.  

*  

* Values:  

* - 0b0 - Timer interrupt disabled.  

* - 0b1 - Timer interrupt enabled.  

*/  

/*@{*/  

/*! @brief Read current value of the LPTMR_CSR_TIE field. */  

#define LPTMR_RD_CSR_TIE(base) ((LPTMR_CSR_REG(base) &  

LPTMR_CSR_TIE_MASK) >> LPTMR_CSR_TIE_SHIFT)  

#define LPTMR_BRD_CSR_TIE(base) (BME_UBFX32(&LPTMR_CSR_REG(base),  

LPTMR_CSR_TIE_SHIFT, LPTMR_CSR_TIE_WIDTH))

```

```

/*! @brief Set the TIE field to a new value. */
#define LPTMR_WR_CSR_TIE(base, value) (LPTMR_RMW_CSR(base,
(LPTMR_CSR_TIE_MASK | LPTMR_CSR_TCF_MASK), LPTMR_CSR_TIE(value)))
#define LPTMR_BWR_CSR_TIE(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),
((uint32_t)(value) << LPTMR_CSR_TIE_SHIFT), LPTMR_CSR_TIE_SHIFT,
LPTMR_CSR_TIE_WIDTH))
/*@}*/

/*!!
 * @name Register LPTMR_CSR, field TCF[7] (W1C)
 *
 * TCF is set when the LPTMR is enabled and the CNR equals the CMR and
 * increments. TCF is cleared when the LPTMR is disabled or a logic 1 is
written to it.
 *
 * Values:
 * - 0b0 - The value of CNR is not equal to CMR and increments.
 * - 0b1 - The value of CNR is equal to CMR and increments.
 */
/*@{*/
/*! @brief Read current value of the LPTMR_CSR_TCF field. */
#define LPTMR_RD_CSR_TCF(base) ((LPTMR_CSR_REG(base) &
LPTMR_CSR_TCF_MASK) >> LPTMR_CSR_TCF_SHIFT)
#define LPTMR_BRD_CSR_TCF(base) (BME_UBFX32(&LPTMR_CSR_REG(base),
LPTMR_CSR_TCF_SHIFT, LPTMR_CSR_TCF_WIDTH))

/*! @brief Set the TCF field to a new value. */
#define LPTMR_WR_CSR_TCF(base, value) (LPTMR_RMW_CSR(base,
LPTMR_CSR_TCF_MASK, LPTMR_CSR_TCF(value)))
#define LPTMR_BWR_CSR_TCF(base, value) (BME_BFI32(&LPTMR_CSR_REG(base),
((uint32_t)(value) << LPTMR_CSR_TCF_SHIFT), LPTMR_CSR_TCF_SHIFT,
LPTMR_CSR_TCF_WIDTH))
/*@}*/

*****  

*****  

 * LPTMR_PSR - Low Power Timer Prescale Register  

*****  

*****  

/*!  

 * @brief LPTMR_PSR - Low Power Timer Prescale Register (RW)  

 *  

 * Reset value: 0x00000000U  

 */
/*!  

 * @name Constants and macros for entire LPTMR_PSR register  

 */
/*@{*/
#define LPTMR_RD_PSR(base) (LPTMR_PSR_REG(base))  

#define LPTMR_WR_PSR(base, value) (LPTMR_PSR_REG(base) = (value))  

#define LPTMR_RMW_PSR(base, mask, value) (LPTMR_WR_PSR(base,
(LPTMR_RD_PSR(base) & ~mask) | (value)))

```

```

#define LPTMR_SET_PSR(base, value) (BME_OR32(&LPTMR_PSR_REG(base),  

(uint32_t)(value)))  

#define LPTMR_CLR_PSR(base, value) (BME_AND32(&LPTMR_PSR_REG(base),  

(uint32_t)(~(value))))  

#define LPTMR_TOG_PSR(base, value) (BME_XOR32(&LPTMR_PSR_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual LPTMR_PSR bitfields
 */

/*!  

 * @name Register LPTMR_PSR, field PCS[1:0] (RW)  

 *  

 * Selects the clock to be used by the LPTMR prescaler/glitch filter. PCS  

must  

* be altered only when the LPTMR is disabled. The clock connections vary  

by  

* device. See the chip configuration details for information on the  

connections to  

* these inputs.  

*  

* Values:  

* - 0b00 - Prescaler/glitch filter clock 0 selected.  

* - 0b01 - Prescaler/glitch filter clock 1 selected.  

* - 0b10 - Prescaler/glitch filter clock 2 selected.  

* - 0b11 - Prescaler/glitch filter clock 3 selected.  

*/
/*@{*/  

/*! @brief Read current value of the LPTMR_PSR_PCS field. */  

#define LPTMR_RD_PSR_PCS(base) ((LPTMR_PSR_REG(base) &  

LPTMR_PSR_PCS_MASK) >> LPTMR_PSR_PCS_SHIFT)  

#define LPTMR_BRD_PSR_PCS(base) (BME_UBFX32(&LPTMR_PSR_REG(base),  

LPTMR_PSR_PCS_SHIFT, LPTMR_PSR_PCS_WIDTH))  

/*! @brief Set the PCS field to a new value. */  

#define LPTMR_WR_PSR_PCS(base, value) (LPTMR_RMW_PSR(base,  

LPTMR_PSR_PCS_MASK, LPTMR_PSR_PCS(value)))  

#define LPTMR_BWR_PSR_PCS(base, value) (BME_BFI32(&LPTMR_PSR_REG(base),  

((uint32_t)(value) << LPTMR_PSR_PCS_SHIFT), LPTMR_PSR_PCS_SHIFT,  

LPTMR_PSR_PCS_WIDTH))  

/*@}/
```

/\*!  
\* @name Register LPTMR\_PSR, field PBYP[2] (RW)  
\*  
\* When PBYP is set, the selected prescaler clock in Time Counter mode or  
\* selected input source in Pulse Counter mode directly clocks the CNR.  
When PBYP is  
\* clear, the CNR is clocked by the output of the prescaler/glitch  
filter. PBYP  
\* must be altered only when the LPTMR is disabled.  
\*

```

* Values:
* - 0b0 - Prescaler/glitch filter is enabled.
* - 0b1 - Prescaler/glitch filter is bypassed.
*/
/*@{/ */
/*! @brief Read current value of the LPTMR_PSR_PBYP field. */
#define LPTMR_RD_PSR_PBYP(base) ((LPTMR_PSR_REG(base) &
LPTMR_PSR_PBYP_MASK) >> LPTMR_PSR_PBYP_SHIFT)
#define LPTMR_BRD_PSR_PBYP(base) (BME_UBFX32(&LPTMR_PSR_REG(base),
LPTMR_PSR_PBYP_SHIFT, LPTMR_PSR_PBYP_WIDTH))

/*! @brief Set the PBYP field to a new value. */
#define LPTMR_WR_PSR_PBYP(base, value) (LPTMR_RMW_PSR(base,
LPTMR_PSR_PBYP_MASK, LPTMR_PSR_PBYP(value)))
#define LPTMR_BWR_PSR_PBYP(base, value) (BME_BFI32(&LPTMR_PSR_REG(base),
(uint32_t)(value) << LPTMR_PSR_PBYP_SHIFT), LPTMR_PSR_PBYP_SHIFT,
LPTMR_PSR_PBYP_WIDTH)
/*@{/ */

/*!
* @name Register LPTMR_PSR, field PRESCALE[6:3] (RW)
*
* Configures the size of the Prescaler in Time Counter mode or width of
the
* glitch filter in Pulse Counter mode. PRESCALE must be altered only
when the LPTMR
* is disabled.
*
* Values:
* - 0b0000 - Prescaler divides the prescaler clock by 2; glitch filter
does not
* support this configuration.
* - 0b0001 - Prescaler divides the prescaler clock by 4; glitch filter
recognizes change on input pin after 2 rising clock edges.
* - 0b0010 - Prescaler divides the prescaler clock by 8; glitch filter
recognizes change on input pin after 4 rising clock edges.
* - 0b0011 - Prescaler divides the prescaler clock by 16; glitch filter
recognizes change on input pin after 8 rising clock edges.
* - 0b0100 - Prescaler divides the prescaler clock by 32; glitch filter
recognizes change on input pin after 16 rising clock edges.
* - 0b0101 - Prescaler divides the prescaler clock by 64; glitch filter
recognizes change on input pin after 32 rising clock edges.
* - 0b0110 - Prescaler divides the prescaler clock by 128; glitch filter
recognizes change on input pin after 64 rising clock edges.
* - 0b0111 - Prescaler divides the prescaler clock by 256; glitch filter
recognizes change on input pin after 128 rising clock edges.
* - 0b1000 - Prescaler divides the prescaler clock by 512; glitch filter
recognizes change on input pin after 256 rising clock edges.
* - 0b1001 - Prescaler divides the prescaler clock by 1024; glitch
filter
* recognizes change on input pin after 512 rising clock edges.
* - 0b1010 - Prescaler divides the prescaler clock by 2048; glitch
filter
* recognizes change on input pin after 1024 rising clock edges.

```

```

* - 0b1011 - Prescaler divides the prescaler clock by 4096; glitch
filter
*      recognizes change on input pin after 2048 rising clock edges.
* - 0b1100 - Prescaler divides the prescaler clock by 8192; glitch
filter
*      recognizes change on input pin after 4096 rising clock edges.
* - 0b1101 - Prescaler divides the prescaler clock by 16,384; glitch
filter
*      recognizes change on input pin after 8192 rising clock edges.
* - 0b1110 - Prescaler divides the prescaler clock by 32,768; glitch
filter
*      recognizes change on input pin after 16,384 rising clock edges.
* - 0b1111 - Prescaler divides the prescaler clock by 65,536; glitch
filter
*      recognizes change on input pin after 32,768 rising clock edges.
*/
/*@{*/
/*! @brief Read current value of the LPTMR_PSR_PRESCALE field. */
#define LPTMR_RD_PSR_PRESCALE(base) ((LPTMR_PSR_REG(base) &
LPTMR_PSR_PRESCALE_MASK) >> LPTMR_PSR_PRESCALE_SHIFT)
#define LPTMR_BRD_PSR_PRESCALE(base) (BME_UBFX32(&LPTMR_PSR_REG(base),
LPTMR_PSR_PRESCALE_SHIFT, LPTMR_PSR_PRESCALE_WIDTH))

/*! @brief Set the PRESCALE field to a new value. */
#define LPTMR_WR_PSR_PRESCALE(base, value) (LPTMR_RMW_PSR(base,
LPTMR_PSR_PRESCALE_MASK, LPTMR_PSR_PRESCALE(value)))
#define LPTMR_BWR_PSR_PRESCALE(base, value)
(BME_BFI32(&LPTMR_PSR_REG(base), ((uint32_t)(value) <<
LPTMR_PSR_PRESCALE_SHIFT), LPTMR_PSR_PRESCALE_SHIFT,
LPTMR_PSR_PRESCALE_WIDTH))
/*@}*/
*****  

*****  

* LPTMR_CMR - Low Power Timer Compare Register  

*****  

*****  

/*!  

* @brief LPTMR_CMR - Low Power Timer Compare Register (RW)  

*  

* Reset value: 0x00000000U  

*/  

/*!  

* @name Constants and macros for entire LPTMR_CMR register  

*/  

/*@{*/
#define LPTMR_RD_CMR(base) (LPTMR_CMR_REG(base))
#define LPTMR_WR_CMR(base, value) (LPTMR_CMR_REG(base) = (value))
#define LPTMR_RMW_CMR(base, mask, value) (LPTMR_WR_CMR(base,
(LPTMR_RD_CMR(base) & ~mask) | value)))
#define LPTMR_SET_CMR(base, value) (BME_OR32(&LPTMR_CMR_REG(base),
(uint32_t)(value)))

```

```

#define LPTMR_CLR_CMR(base, value) (BME_AND32(&LPTMR_CMREG(base),  

(uint32_t)(~(value))))  

#define LPTMR_TOG_CMREG(base, value) (BME_XOR32(&LPTMR_CMREG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual LPTMR_CMREG bitfields
 */

/*!  

 * @name Register LPTMR_CMREG, field COMPARE[15:0] (RW)  

 *  

 * When the LPTMR is enabled and the CNR equals the value in the CMR and  

 * increments, TCF is set and the hardware trigger asserts until the next  

time the CNR  

 * increments. If the CMR is 0, the hardware trigger will remain asserted  

until  

 * the LPTMR is disabled. If the LPTMR is enabled, the CMR must be  

altered only  

 * when TCF is set.  

 */
/*@{*/  

/*! @brief Read current value of the LPTMR_CMREG_COMPARE field. */  

#define LPTMR_RD_CMREG_COMPARE(base) ((LPTMR_CMREG(base) &  

LPTMR_CMREG_COMPARE_MASK) >> LPTMR_CMREG_COMPARE_SHIFT)  

#define LPTMR_BRD_CMREG_COMPARE(base) (BME_UBFX32(&LPTMR_CMREG(base),  

LPTMR_CMREG_COMPARE_SHIFT, LPTMR_CMREG_COMPARE_WIDTH))  

/*! @brief Set the COMPARE field to a new value. */  

#define LPTMR_WR_CMREG_COMPARE(base, value) (LPTMR_RMW_CMREG(base,  

LPTMR_CMREG_COMPARE_MASK, LPTMR_CMREG_COMPARE(value)))  

#define LPTMR_BWR_CMREG_COMPARE(base, value)  

(BME_BFI32(&LPTMR_CMREG(base), ((uint32_t)(value) <<  

LPTMR_CMREG_COMPARE_SHIFT), LPTMR_CMREG_COMPARE_SHIFT,  

LPTMR_CMREG_COMPARE_WIDTH))  

/*@}/
```

\*\*\*\*\*  
\*\*\*\*\*  
\* LPTMR\_CNR - Low Power Timer Counter Register  
\*\*\*\*\*  
\*\*\*\*\*/

```

/*!  

 * @brief LPTMR_CNR - Low Power Timer Counter Register (RW)  

 *  

 * Reset value: 0x00000000U  

 */
/*!  

 * @name Constants and macros for entire LPTMR_CNR register  

 */
/*@{*/
```

```

#define LPTMR_RD_CNR(base)          (LPTMR_CNR_REG(base))
#define LPTMR_WR_CNR(base, value)   (LPTMR_CNR_REG(base) = (value))
#define LPTMR_RMW_CNR(base, mask, value) (LPTMR_WR_CNR(base,
(LPTMR_RD_CNR(base) & ~mask) | (value)))
#define LPTMR_SET_CNR(base, value)  (BME_OR32(&LPTMR_CNR_REG(base),
(uint32_t)(value)))
#define LPTMR_CLR_CNR(base, value)  (BME_AND32(&LPTMR_CNR_REG(base),
(uint32_t)(~(value))))
#define LPTMR_TOG_CNR(base, value)  (BME_XOR32(&LPTMR_CNR_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual LPTMR_CNR bitfields
 */

/*!!
 * @name Register LPTMR_CNR, field COUNTER[15:0] (RW)
 */
/*@{*/
/*! @brief Read current value of the LPTMR_CNR_COUNTER field. */
#define LPTMR_RD_CNR_COUNTER(base) ((LPTMR_CNR_REG(base) &
LPTMR_CNR_COUNTER_MASK) >> LPTMR_CNR_COUNTER_SHIFT)
#define LPTMR_BRD_CNR_COUNTER(base) (BME_UBFX32(&LPTMR_CNR_REG(base),
LPTMR_CNR_COUNTER_SHIFT, LPTMR_CNR_COUNTER_WIDTH))

/*! @brief Set the COUNTER field to a new value. */
#define LPTMR_WR_CNR_COUNTER(base, value) (LPTMR_RMW_CNR(base,
LPTMR_CNR_COUNTER_MASK, LPTMR_CNR_COUNTER(value)))
#define LPTMR_BWR_CNR_COUNTER(base, value)
(BME_BFI32(&LPTMR_CNR_REG(base), ((uint32_t)(value) <<
LPTMR_CNR_COUNTER_SHIFT), LPTMR_CNR_COUNTER_SHIFT,
LPTMR_CNR_COUNTER_WIDTH))
/*@}*/

/*
 * MKL25Z4 MCG
 *
 * Multipurpose Clock Generator module
 *
 * Registers defined in this header file:
 * - MCG_C1 - MCG Control 1 Register
 * - MCG_C2 - MCG Control 2 Register
 * - MCG_C3 - MCG Control 3 Register
 * - MCG_C4 - MCG Control 4 Register
 * - MCG_C5 - MCG Control 5 Register
 * - MCG_C6 - MCG Control 6 Register
 * - MCG_S - MCG Status Register
 * - MCG_SC - MCG Status and Control Register
 * - MCG_ATCVH - MCG Auto Trim Compare Value High Register
 * - MCG_ATCVL - MCG Auto Trim Compare Value Low Register
 * - MCG_C7 - MCG Control 7 Register
 * - MCG_C8 - MCG Control 8 Register
 * - MCG_C9 - MCG Control 9 Register

```

```

* - MCG_C10 - MCG Control 10 Register
*/
#define MCG_INSTANCE_COUNT (1U) /*!< Number of instances of the MCG
module. */
#define MCG_IDX (0U) /*!< Instance number for MCG. */

*****  

*****  

* MCG_C1 - MCG Control 1 Register  

*****  

*****  

/*!  

* @brief MCG_C1 - MCG Control 1 Register (RW)  

*  

* Reset value: 0x04U  

*/  

/*!  

* @name Constants and macros for entire MCG_C1 register  

*/  

/*@{ */  

#define MCG_RD_C1(base) (MCG_C1_REG(base))  

#define MCG_WR_C1(base, value) (MCG_C1_REG(base) = (value))  

#define MCG_RMW_C1(base, mask, value) (MCG_WR_C1(base, (MCG_RD_C1(base) &  

~(mask)) | (value)))  

#define MCG_SET_C1(base, value) (BME_OR8(&MCG_C1_REG(base),  

(uint8_t)(value)))  

#define MCG_CLR_C1(base, value) (BME_AND8(&MCG_C1_REG(base),  

(uint8_t)(~(value))))  

#define MCG_TOG_C1(base, value) (BME_XOR8(&MCG_C1_REG(base),  

(uint8_t)(value)))  

/*}@*/  

  

/*  

* Constants & macros for individual MCG_C1 bitfields  

*/  

  

/*!  

* @name Register MCG_C1, field IREFSTEN[0] (RW)  

*  

* Controls whether or not the internal reference clock remains enabled  

when the  

* MCG enters Stop mode.  

*  

* Values:  

* - 0b0 - Internal reference clock is disabled in Stop mode.  

* - 0b1 - Internal reference clock is enabled in Stop mode if IRCLKEN is  

set or  

*      if MCG is in FEI, FBI, or BLPI modes before entering Stop mode.  

*/  

/*@{ */  

/*! @brief Read current value of the MCG_C1_IREFSTEN field. */

```

```

#define MCG_RD_C1_IREFSTEN(base) ((MCG_C1_REG(base) &
MCG_C1_IREFSTEN_MASK) >> MCG_C1_IREFSTEN_SHIFT)
#define MCG_BRD_C1_IREFSTEN(base) (BME_UBFX8(&MCG_C1_REG(base),
MCG_C1_IREFSTEN_SHIFT, MCG_C1_IREFSTEN_WIDTH))

/*! @brief Set the IREFSTEN field to a new value. */
#define MCG_WR_C1_IREFSTEN(base, value) (MCG_RMW_C1(base,
MCG_C1_IREFSTEN_MASK, MCG_C1_IREFSTEN(value)))
#define MCG_BWR_C1_IREFSTEN(base, value) (BME_BFI8(&MCG_C1_REG(base),
((uint8_t)(value) << MCG_C1_IREFSTEN_SHIFT), MCG_C1_IREFSTEN_SHIFT,
MCG_C1_IREFSTEN_WIDTH))
/*@}*/

/*!
 * @name Register MCG_C1, field IRCLKEN[1] (RW)
 *
 * Enables the internal reference clock for use as MCGIRCLK.
 *
 * Values:
 * - 0b0 - MCGIRCLK inactive.
 * - 0b1 - MCGIRCLK active.
 */
/*@{*/
/*! @brief Read current value of the MCG_C1_IRCLKEN field. */
#define MCG_RD_C1_IRCLKEN(base) ((MCG_C1_REG(base) & MCG_C1_IRCLKEN_MASK)
>> MCG_C1_IRCLKEN_SHIFT)
#define MCG_BRD_C1_IRCLKEN(base) (BME_UBFX8(&MCG_C1_REG(base),
MCG_C1_IRCLKEN_SHIFT, MCG_C1_IRCLKEN_WIDTH))

/*! @brief Set the IRCLKEN field to a new value. */
#define MCG_WR_C1_IRCLKEN(base, value) (MCG_RMW_C1(base,
MCG_C1_IRCLKEN_MASK, MCG_C1_IRCLKEN(value)))
#define MCG_BWR_C1_IRCLKEN(base, value) (BME_BFI8(&MCG_C1_REG(base),
((uint8_t)(value) << MCG_C1_IRCLKEN_SHIFT), MCG_C1_IRCLKEN_SHIFT,
MCG_C1_IRCLKEN_WIDTH))
/*@}*/

/*!
 * @name Register MCG_C1, field IREFS[2] (RW)
 *
 * Selects the reference clock source for the FLL.
 *
 * Values:
 * - 0b0 - External reference clock is selected.
 * - 0b1 - The slow internal reference clock is selected.
 */
/*@{*/
/*! @brief Read current value of the MCG_C1_IREFS field. */
#define MCG_RD_C1_IREFS(base) ((MCG_C1_REG(base) & MCG_C1_IREFS_MASK) >>
MCG_C1_IREFS_SHIFT)
#define MCG_BRD_C1_IREFS(base) (BME_UBFX8(&MCG_C1_REG(base),
MCG_C1_IREFS_SHIFT, MCG_C1_IREFS_WIDTH))

/*! @brief Set the IREFS field to a new value. */

```

```

#define MCG_WR_C1_IREFS(base, value)  (MCG_RMW_C1(base, MCG_C1_IREFS_MASK, \
MCG_C1_IREFS(value)))
#define MCG_BWR_C1_IREFS(base, value)  (BME_BFI8(&MCG_C1_REG(base), \
((uint8_t)(value) << MCG_C1_IREFS_SHIFT), MCG_C1_IREFS_SHIFT, \
MCG_C1_IREFS_WIDTH))
/*@}*/

/*
 * @name Register MCG_C1, field FRDIV[5:3] (RW)
 *
 * Selects the amount to divide down the external reference clock for the
FLL.
 * The resulting frequency must be in the range 31.25 kHz to 39.0625 kHz
(This is
 * required when FLL/DCO is the clock source for MCGOUTCLK . In FBE mode,
it is
 * not required to meet this range, but it is recommended in the cases
when trying
 * to enter a FLL mode from FBE).
 *
 * Values:
 * - 0b000 - If RANGE 0 = 0 , Divide Factor is 1; for all other RANGE 0
values,
 *     Divide Factor is 32.
 * - 0b001 - If RANGE 0 = 0 , Divide Factor is 2; for all other RANGE 0
values,
 *     Divide Factor is 64.
 * - 0b010 - If RANGE 0 = 0 , Divide Factor is 4; for all other RANGE 0
values,
 *     Divide Factor is 128.
 * - 0b011 - If RANGE 0 = 0 , Divide Factor is 8; for all other RANGE 0
values,
 *     Divide Factor is 256.
 * - 0b100 - If RANGE 0 = 0 , Divide Factor is 16; for all other RANGE 0
values,
 *     Divide Factor is 512.
 * - 0b101 - If RANGE 0 = 0 , Divide Factor is 32; for all other RANGE 0
values,
 *     Divide Factor is 1024.
 * - 0b110 - If RANGE 0 = 0 , Divide Factor is 64; for all other RANGE 0
values,
 *     Divide Factor is 1280 .
 * - 0b111 - If RANGE 0 = 0 , Divide Factor is 128; for all other RANGE 0
values, Divide Factor is 1536 .
*/
/*@{*/
/*! @brief Read current value of the MCG_C1_FRDIV field. */
#define MCG_RD_C1_FRDIV(base)  ((MCG_C1_REG(base) & MCG_C1_FRDIV_MASK) >>
MCG_C1_FRDIV_SHIFT)
#define MCG_BRD_C1_FRDIV(base)  (BME_UBFX8(&MCG_C1_REG(base),
MCG_C1_FRDIV_SHIFT, MCG_C1_FRDIV_WIDTH))

/*! @brief Set the FRDIV field to a new value. */

```

```

#define MCG_WR_C1_FRDIV(base, value)  (MCG_RMW_C1(base, MCG_C1_FRDIV_MASK,
MCG_C1_FRDIV(value)))
#define MCG_BWR_C1_FRDIV(base, value)  (BME_BFI8(&MCG_C1_REG(base),
((uint8_t)(value) << MCG_C1_FRDIV_SHIFT), MCG_C1_FRDIV_SHIFT,
MCG_C1_FRDIV_WIDTH))
/*@}*/

/*
 * @name Register MCG_C1, field CLKS[7:6] (RW)
 *
 * Selects the clock source for MCGOUTCLK .
 *
 * Values:
 * - 0b00 - Encoding 0 - Output of FLL or PLL is selected (depends on
PLLS
 *         control bit).
 * - 0b01 - Encoding 1 - Internal reference clock is selected.
 * - 0b10 - Encoding 2 - External reference clock is selected.
 * - 0b11 - Encoding 3 - Reserved.
 */
/*@*/
/*! @brief Read current value of the MCG_C1_CLKS field. */
#define MCG_RD_C1_CLKS(base) ((MCG_C1_REG(base) & MCG_C1_CLKS_MASK) >>
MCG_C1_CLKS_SHIFT)
#define MCG_BRD_C1_CLKS(base) (BME_UBFX8(&MCG_C1_REG(base),
MCG_C1_CLKS_SHIFT, MCG_C1_CLKS_WIDTH))

/*! @brief Set the CLKS field to a new value. */
#define MCG_WR_C1_CLKS(base, value) (MCG_RMW_C1(base, MCG_C1_CLKS_MASK,
MCG_C1_CLKS(value)))
#define MCG_BWR_C1_CLKS(base, value) (BME_BFI8(&MCG_C1_REG(base),
((uint8_t)(value) << MCG_C1_CLKS_SHIFT), MCG_C1_CLKS_SHIFT,
MCG_C1_CLKS_WIDTH))
/*@*/

*****  

****  

 * MCG_C2 - MCG Control 2 Register  

*****  

****/  

  

/*
 * @brief MCG_C2 - MCG Control 2 Register (RW)
 *
 * Reset value: 0x80U
 */
/*!  

 * @name Constants and macros for entire MCG_C2 register
 */
/*@*/
#define MCG_RD_C2(base)          (MCG_C2_REG(base))
#define MCG_WR_C2(base, value)    (MCG_C2_REG(base) = (value))

```

```

#define MCG_RMW_C2(base, mask, value)  (MCG_WR_C2(base, (MCG_RD_C2(base) &
~(mask)) | (value)))
#define MCG_SET_C2(base, value)      (BME_OR8(&MCG_C2_REG(base),
(uint8_t)(value)))
#define MCG_CLR_C2(base, value)     (BME_AND8(&MCG_C2_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_C2(base, value)     (BME_XOR8(&MCG_C2_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual MCG_C2 bitfields
 */

/*!!
 * @name Register MCG_C2, field IRCS[0] (RW)
 *
 * Selects between the fast or slow internal reference clock source.
 *
 * Values:
 * - 0b0 - Slow internal reference clock selected.
 * - 0b1 - Fast internal reference clock selected.
 */
/*@{*/
/*! @brief Read current value of the MCG_C2_IRCS field. */
#define MCG_RD_C2_IRCS(base) ((MCG_C2_REG(base) & MCG_C2_IRCS_MASK) >>
MCG_C2_IRCS_SHIFT)
#define MCG_BRD_C2_IRCS(base) (BME_UBFX8(&MCG_C2_REG(base),
MCG_C2_IRCS_SHIFT, MCG_C2_IRCS_WIDTH))

/*! @brief Set the IRCS field to a new value. */
#define MCG_WR_C2_IRCS(base, value) (MCG_RMW_C2(base, MCG_C2_IRCS_MASK,
MCG_C2_IRCS(value)))
#define MCG_BWR_C2_IRCS(base, value) (BME_BFI8(&MCG_C2_REG(base),
((uint8_t)(value) << MCG_C2_IRCS_SHIFT), MCG_C2_IRCS_SHIFT,
MCG_C2_IRCS_WIDTH))
/*@}*/

/*
 * @name Register MCG_C2, field LP[1] (RW)
 *
 * Controls whether the FLL or PLL is disabled in BLPI and BLPE modes. In
FBE or
 * PBE modes, setting this bit to 1 will transition the MCG into BLPE
mode; in
 * FBI mode, setting this bit to 1 will transition the MCG into BLPI
mode. In any
 * other MCG mode, LP bit has no affect.
 *
 * Values:
 * - 0b0 - FLL or PLL is not disabled in bypass modes.
 * - 0b1 - FLL or PLL is disabled in bypass modes (lower power)
 */
/*@{*/

```

```

/*! @brief Read current value of the MCG_C2_LP field. */
#define MCG_RD_C2_LP(base) ((MCG_C2_REG(base) & MCG_C2_LP_MASK) >>
MCG_C2_LP_SHIFT)
#define MCG_BRD_C2_LP(base) (BME_UBFX8(&MCG_C2_REG(base),
MCG_C2_LP_SHIFT, MCG_C2_LP_WIDTH))

/*! @brief Set the LP field to a new value. */
#define MCG_WR_C2_LP(base, value) (MCG_RMW_C2(base, MCG_C2_LP_MASK,
MCG_C2_LP(value)))
#define MCG_BWR_C2_LP(base, value) (BME_BFI8(&MCG_C2_REG(base),
((uint8_t)(value) << MCG_C2_LP_SHIFT), MCG_C2_LP_SHIFT, MCG_C2_LP_WIDTH))
/*@}*/

/*!
 * @name Register MCG_C2, field EREFS0[2] (RW)
 *
 * Selects the source for the external reference clock. See the
Oscillator (OSC)
 * chapter for more details.
 *
 * Values:
 * - 0b0 - External reference clock requested.
 * - 0b1 - Oscillator requested.
 */
/*@{*/
/*! @brief Read current value of the MCG_C2_EREFS0 field. */
#define MCG_RD_C2_EREFS0(base) ((MCG_C2_REG(base) & MCG_C2_EREFS0_MASK)
>> MCG_C2_EREFS0_SHIFT)
#define MCG_BRD_C2_EREFS0(base) (BME_UBFX8(&MCG_C2_REG(base),
MCG_C2_EREFS0_SHIFT, MCG_C2_EREFS0_WIDTH))

/*! @brief Set the EREFS0 field to a new value. */
#define MCG_WR_C2_EREFS0(base, value) (MCG_RMW_C2(base,
MCG_C2_EREFS0_MASK, MCG_C2_EREFS0(value)))
#define MCG_BWR_C2_EREFS0(base, value) (BME_BFI8(&MCG_C2_REG(base),
((uint8_t)(value) << MCG_C2_EREFS0_SHIFT), MCG_C2_EREFS0_SHIFT,
MCG_C2_EREFS0_WIDTH))
/*@}*/

/*!
 * @name Register MCG_C2, field HGO0[3] (RW)
 *
 * Controls the crystal oscillator mode of operation. See the Oscillator
(OSC)
 * chapter for more details.
 *
 * Values:
 * - 0b0 - Configure crystal oscillator for low-power operation.
 * - 0b1 - Configure crystal oscillator for high-gain operation.
 */
/*@{*/
/*! @brief Read current value of the MCG_C2_HGO0 field. */
#define MCG_RD_C2_HGO0(base) ((MCG_C2_REG(base) & MCG_C2_HGO0_MASK) >>
MCG_C2_HGO0_SHIFT)

```

```

#define MCG_BRD_C2_HGO0(base)  (BME_UBFX8(&MCG_C2_REG(base),  

MCG_C2_HGO0_SHIFT, MCG_C2_HGO0_WIDTH))

/*! @brief Set the HGO0 field to a new value. */  

#define MCG_WR_C2_HGO0(base, value)  (MCG_RMW_C2(base, MCG_C2_HGO0_MASK,  

MCG_C2_HGO0(value)))  

#define MCG_BWR_C2_HGO0(base, value)  (BME_BFI8(&MCG_C2_REG(base),  

(uint8_t)(value) << MCG_C2_HGO0_SHIFT), MCG_C2_HGO0_SHIFT,  

MCG_C2_HGO0_WIDTH))  

/*@}*/

/*!  

 * @name Register MCG_C2, field RANGE0[5:4] (RW)  

 *  

 * Selects the frequency range for the crystal oscillator or external  

clock  

 * source. See the Oscillator (OSC) chapter for more details and the  

device data  

 * sheet for the frequency ranges used.  

 *  

 * Values:  

 * - 0b00 - Encoding 0 - Low frequency range selected for the crystal  

oscillator  

 * .  

 * - 0b01 - Encoding 1 - High frequency range selected for the crystal  

oscillator .  

 */  

/*@*/  

/*! @brief Read current value of the MCG_C2_RANGE0 field. */  

#define MCG_RD_C2_RANGE0(base)  ((MCG_C2_REG(base) & MCG_C2_RANGE0_MASK)  

>> MCG_C2_RANGE0_SHIFT)  

#define MCG_BRD_C2_RANGE0(base)  (BME_UBFX8(&MCG_C2_REG(base),  

MCG_C2_RANGE0_SHIFT, MCG_C2_RANGE0_WIDTH))

/*! @brief Set the RANGE0 field to a new value. */  

#define MCG_WR_C2_RANGE0(base, value)  (MCG_RMW_C2(base,  

MCG_C2_RANGE0_MASK, MCG_C2_RANGE0(value)))  

#define MCG_BWR_C2_RANGE0(base, value)  (BME_BFI8(&MCG_C2_REG(base),  

(uint8_t)(value) << MCG_C2_RANGE0_SHIFT), MCG_C2_RANGE0_SHIFT,  

MCG_C2_RANGE0_WIDTH))  

/*@*/

/*!  

 * @name Register MCG_C2, field LOCRE0[7] (RW)  

 *  

 * Determines whether an interrupt or a reset request is made following a  

loss  

 * of OSC0 external reference clock. The LOCRE0 only has an affect when  

CME0 is  

 * set.  

 *  

 * Values:  

 * - 0b0 - Interrupt request is generated on a loss of OSC0 external  

reference

```

```

*      clock.
* - 0b1 - Generate a reset request on a loss of OSC0 external reference
clock.
*/
/*@{*/
/*! @brief Read current value of the MCG_C2_LOCRE0 field. */
#define MCG_RD_C2_LOCRE0(base) ((MCG_C2_REG(base) & MCG_C2_LOCRE0_MASK)
>> MCG_C2_LOCRE0_SHIFT)
#define MCG_BRD_C2_LOCRE0(base) (BME_UBFX8(&MCG_C2_REG(base),
MCG_C2_LOCRE0_SHIFT, MCG_C2_LOCRE0_WIDTH))

/*! @brief Set the LOCRE0 field to a new value. */
#define MCG_WR_C2_LOCRE0(base, value) (MCG_RMW_C2(base,
MCG_C2_LOCRE0_MASK, MCG_C2_LOCRE0(value)))
#define MCG_BWR_C2_LOCRE0(base, value) (BME_BFI8(&MCG_C2_REG(base),
(uint8_t)(value) << MCG_C2_LOCRE0_SHIFT), MCG_C2_LOCRE0_SHIFT,
MCG_C2_LOCRE0_WIDTH)
/*}@*/



/********************* MCG_C3 - MCG Control 3 Register ********************
****

* MCG_C3 - MCG Control 3 Register

*****/


/*!
* @brief MCG_C3 - MCG Control 3 Register (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire MCG_C3 register
*/
/*@{*/
#define MCG_RD_C3(base)          (MCG_C3_REG(base))
#define MCG_WR_C3(base, value)    (MCG_C3_REG(base) = (value))
#define MCG_RMW_C3(base, mask, value) (MCG_WR_C3(base, (MCG_RD_C3(base) &
~(mask)) | (value)))
#define MCG_SET_C3(base, value)   (BME_OR8(&MCG_C3_REG(base),
(uint8_t)(value)))
#define MCG_CLR_C3(base, value)   (BME_AND8(&MCG_C3_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_C3(base, value)   (BME_XOR8(&MCG_C3_REG(base),
(uint8_t)(value)))
/*}@*/



/********************* MCG_C4 - MCG Control 4 Register ********************
****

* MCG_C4 - MCG Control 4 Register

*****/

```

```

/*!
 * @brief MCG_C4 - MCG Control 4 Register (RW)
 *
 * Reset value: 0x00U
 *
 * Reset values for DRST and DMX32 bits are 0.
 */
/*!
 * @name Constants and macros for entire MCG_C4 register
 */
/*@{*/
#define MCG_RD_C4(base)          (MCG_C4_REG(base))
#define MCG_WR_C4(base, value)    (MCG_C4_REG(base) = (value))
#define MCG_RMW_C4(base, mask, value) (MCG_WR_C4(base, (MCG_RD_C4(base) & ~mask) | (value)))
#define MCG_SET_C4(base, value)   (BME_OR8(&MCG_C4_REG(base), (uint8_t)(value)))
#define MCG_CLR_C4(base, value)   (BME_AND8(&MCG_C4_REG(base), (uint8_t)(~(value))))
#define MCG_TOG_C4(base, value)   (BME_XOR8(&MCG_C4_REG(base), (uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual MCG_C4 bitfields
 */

/*!
 * @name Register MCG_C4, field SCFTRIM[0] (RW)
 *
 * SCFTRIM A value for SCFTRIM is loaded during reset from a factory programmed
 * location . controls the smallest adjustment of the slow internal reference
 * clock frequency. Setting SCFTRIM increases the period and clearing SCFTRIM
 * decreases the period by the smallest amount possible. If an SCFTRIM value stored in
 * nonvolatile memory is to be used, it is your responsibility to copy that value
 * from the nonvolatile memory location to this bit.
 */
/*@{*/
/*! @brief Read current value of the MCG_C4_SCFTRIM field. */
#define MCG_RD_C4_SCFTRIM(base) ((MCG_C4_REG(base) & MCG_C4_SCFTRIM_MASK) >> MCG_C4_SCFTRIM_SHIFT)
#define MCG_BRD_C4_SCFTRIM(base) (BME_UBFX8(&MCG_C4_REG(base), MCG_C4_SCFTRIM_SHIFT, MCG_C4_SCFTRIM_WIDTH))

/*! @brief Set the SCFTRIM field to a new value. */
#define MCG_WR_C4_SCFTRIM(base, value) (MCG_RMW_C4(base, MCG_C4_SCFTRIM_MASK, MCG_C4_SCFTRIM(value)))

```

```

#define MCG_BWR_C4_SCFTRIM(base, value) (BME_BFI8(&MCG_C4_REG(base),  

((uint8_t)(value) << MCG_C4_SCFTRIM_SHIFT), MCG_C4_SCFTRIM_SHIFT,  

MCG_C4_SCFTRIM_WIDTH))  

/*@}*/

/*!  

 * @name Register MCG_C4, field FCTRIM[4:1] (RW)  

 *  

 * FCTRIM A value for FCTRIM is loaded during reset from a factory  

programmed  

 * location . controls the fast internal reference clock frequency by  

controlling  

 * the fast internal reference clock period. The FCTRIM bits are binary  

weighted,  

 * that is, bit 1 adjusts twice as much as bit 0. Increasing the binary  

value  

 * increases the period, and decreasing the value decreases the period.  

If an  

 * FCTRIM[3:0] value stored in nonvolatile memory is to be used, it is  

your  

 * responsibility to copy that value from the nonvolatile memory location  

to this register.  

 */  

/*@{*/  

/*! @brief Read current value of the MCG_C4_FCTRIM field. */  

#define MCG_RD_C4_FCTRIM(base) ((MCG_C4_REG(base) & MCG_C4_FCTRIM_MASK)  

>> MCG_C4_FCTRIM_SHIFT)  

#define MCG_BRD_C4_FCTRIM(base) (BME_UBFX8(&MCG_C4_REG(base),  

MCG_C4_FCTRIM_SHIFT, MCG_C4_FCTRIM_WIDTH))

/*! @brief Set the FCTRIM field to a new value. */  

#define MCG_WR_C4_FCTRIM(base, value) (MCG_RMW_C4(base,  

MCG_C4_FCTRIM_MASK, MCG_C4_FCTRIM(value)))  

#define MCG_BWR_C4_FCTRIM(base, value) (BME_BFI8(&MCG_C4_REG(base),  

((uint8_t)(value) << MCG_C4_FCTRIM_SHIFT), MCG_C4_FCTRIM_SHIFT,  

MCG_C4_FCTRIM_WIDTH))  

/*@}*/

/*!  

 * @name Register MCG_C4, field DRST_DRS[6:5] (RW)  

 *  

 * The DRS bits select the frequency range for the FLL output, DCOOUT.  

When the  

 * LP bit is set, writes to the DRS bits are ignored. The DRST read field  

 * indicates the current frequency range for DCOOUT. The DRST field does  

not update  

 * immediately after a write to the DRS field due to internal  

synchronization between  

 * clock domains. See the DCO Frequency Range table for more details.  

 *  

 * Values:  

 * - 0b00 - Encoding 0 - Low range (reset default).  

 * - 0b01 - Encoding 1 - Mid range.  

 * - 0b10 - Encoding 2 - Mid-high range.

```

```

* - 0b11 - Encoding 3 - High range.
*/
/*@{*/
/*! @brief Read current value of the MCG_C4_DRST_DRS field. */
#define MCG_RD_C4_DRST_DRS(base) ((MCG_C4_REG(base) &
MCG_C4_DRST_DRS_MASK) >> MCG_C4_DRST_DRS_SHIFT)
#define MCG_BRD_C4_DRST_DRS(base) (BME_UBFX8(&MCG_C4_REG(base),
MCG_C4_DRST_DRS_SHIFT, MCG_C4_DRST_DRS_WIDTH))

/*! @brief Set the DRST_DRS field to a new value. */
#define MCG_WR_C4_DRST_DRS(base, value) (MCG_RMW_C4(base,
MCG_C4_DRST_DRS_MASK, MCG_C4_DRST_DRS(value)))
#define MCG_BWR_C4_DRST_DRS(base, value) (BME_BFI8(&MCG_C4_REG(base),
((uint8_t)(value) << MCG_C4_DRST_DRS_SHIFT), MCG_C4_DRST_DRS_SHIFT,
MCG_C4_DRST_DRS_WIDTH))
/*}@*/
```

/\*
 \* @name Register MCG\_C4, field DMX32[7] (RW)
 \*
 \* The DMX32 bit controls whether the DCO frequency range is narrowed to its
 \* maximum frequency with a 32.768 kHz reference. The following table identifies
 \* settings for the DCO frequency range. The system clocks derived from this source
 \* should not exceed their specified maximums. DRST\_DRS DMX32 Reference Range FLL
 \* Factor DCO Range 00 0 31.25-39.0625 kHz 640 20-25 MHz 1 32.768 kHz 732
24 MHz
 \* 01 0 31.25-39.0625 kHz 1280 40-50 MHz 1 32.768 kHz 1464 48 MHz 10 0
 \* 31.25-39.0625 kHz 1920 60-75 MHz 1 32.768 kHz 2197 72 MHz 11 0 31.25-
39.0625 kHz 2560
 \* 80-100 MHz 1 32.768 kHz 2929 96 MHz
 \*
 \* Values:
 \* - 0b0 - DCO has a default range of 25%.
 \* - 0b1 - DCO is fine-tuned for maximum frequency with 32.768 kHz reference.
\*/
/\*@\*/
/\*! @brief Read current value of the MCG\_C4\_DMX32 field. \*/
#define MCG\_RD\_C4\_DMX32(base) ((MCG\_C4\_REG(base) & MCG\_C4\_DMX32\_MASK) >>
MCG\_C4\_DMX32\_SHIFT)
#define MCG\_BRD\_C4\_DMX32(base) (BME\_UBFX8(&MCG\_C4\_REG(base),
MCG\_C4\_DMX32\_SHIFT, MCG\_C4\_DMX32\_WIDTH))

/\*! @brief Set the DMX32 field to a new value. \*/
#define MCG\_WR\_C4\_DMX32(base, value) (MCG\_RMW\_C4(base, MCG\_C4\_DMX32\_MASK,
MCG\_C4\_DMX32(value)))
#define MCG\_BWR\_C4\_DMX32(base, value) (BME\_BFI8(&MCG\_C4\_REG(base),
((uint8\_t)(value) << MCG\_C4\_DMX32\_SHIFT), MCG\_C4\_DMX32\_SHIFT,
MCG\_C4\_DMX32\_WIDTH))
/\*}@\*/

```

/*****
 * MCG_C5 - MCG Control 5 Register
 *****/
/*!
 * @brief MCG_C5 - MCG Control 5 Register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire MCG_C5 register
 */
/*@{*/
#define MCG_RD_C5(base)          (MCG_C5_REG(base))
#define MCG_WR_C5(base, value)    (MCG_C5_REG(base) = (value))
#define MCG_RMW_C5(base, mask, value) (MCG_WR_C5(base, (MCG_RD_C5(base) & ~mask) | (value)))
#define MCG_SET_C5(base, value)   (BME_OR8(&MCG_C5_REG(base), (uint8_t)(value)))
#define MCG_CLR_C5(base, value)   (BME_AND8(&MCG_C5_REG(base), (uint8_t)(~(value))))
#define MCG_TOG_C5(base, value)   (BME_XOR8(&MCG_C5_REG(base), (uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual MCG_C5 bitfields
 */
/*!
 * @name Register MCG_C5, field PRDIV0[4:0] (RW)
 *
 * Selects the amount to divide down the external reference clock for the
 * PLL.
 * The resulting frequency must be in the range of 2 MHz to 4 MHz. After
 * the PLL
 * is enabled (by setting either PLLCLKEN 0 or PLLS), the PRDIV 0 value
 * must not
 * be changed when LOCK0 is zero. PLL External Reference Divide Factor
 * PRDIV 0
 * Divide Factor PRDIV 0 Divide Factor PRDIV 0 Divide Factor PRDIV 0
 * Divide Factor
 * 00000 1 01000 9 10000 17 11000 25 00001 2 01001 10 10001 18 11001
 * Reserved
 * 00010 3 01010 11 10010 19 11010 Reserved 00011 4 01011 12 10011 20
 * 11011 Reserved
 * 00100 5 01100 13 10100 21 11100 Reserved 00101 6 01101 14 10101 22
 * 11101
 * Reserved 00110 7 01110 15 10110 23 11110 Reserved 00111 8 01111 16
 * 10111 24 11111

```

```

    * Reserved
    */
/*@{ */
/*! @brief Read current value of the MCG_C5_PRDIV0 field. */
#define MCG_RD_C5_PRDIV0(base) ((MCG_C5_REG(base) & MCG_C5_PRDIV0_MASK)
>> MCG_C5_PRDIV0_SHIFT)
#define MCG_BRD_C5_PRDIV0(base) (BME_UBFX8(&MCG_C5_REG(base),
MCG_C5_PRDIV0_SHIFT, MCG_C5_PRDIV0_WIDTH))

/*! @brief Set the PRDIV0 field to a new value. */
#define MCG_WR_C5_PRDIV0(base, value) (MCG_RMW_C5(base,
MCG_C5_PRDIV0_MASK, MCG_C5_PRDIV0(value)))
#define MCG_BWR_C5_PRDIV0(base, value) (BME_BFI8(&MCG_C5_REG(base),
((uint8_t)(value) << MCG_C5_PRDIV0_SHIFT), MCG_C5_PRDIV0_SHIFT,
MCG_C5_PRDIV0_WIDTH))
/*@} */

/*!
 * @name Register MCG_C5, field PLLSTEN0[5] (RW)
 *
 * Enables the PLL Clock during Normal Stop. In Low Power Stop mode, the
PLL
 * clock gets disabled even if PLLSTEN 0 =1. All other power modes,
PLLSTEN 0 bit
 * has no affect and does not enable the PLL Clock to run if it is
written to 1.
 *
 * Values:
 * - 0b0 - MCGPLLCLK is disabled in any of the Stop modes.
 * - 0b1 - MCGPLLCLK is enabled if system is in Normal Stop mode.
 */
/*@{ */
/*! @brief Read current value of the MCG_C5_PLLSTEN0 field. */
#define MCG_RD_C5_PLLSTEN0(base) ((MCG_C5_REG(base) &
MCG_C5_PLLSTEN0_MASK) >> MCG_C5_PLLSTEN0_SHIFT)
#define MCG_BRD_C5_PLLSTEN0(base) (BME_UBFX8(&MCG_C5_REG(base),
MCG_C5_PLLSTEN0_SHIFT, MCG_C5_PLLSTEN0_WIDTH))

/*! @brief Set the PLLSTEN0 field to a new value. */
#define MCG_WR_C5_PLLSTEN0(base, value) (MCG_RMW_C5(base,
MCG_C5_PLLSTEN0_MASK, MCG_C5_PLLSTEN0(value)))
#define MCG_BWR_C5_PLLSTEN0(base, value) (BME_BFI8(&MCG_C5_REG(base),
((uint8_t)(value) << MCG_C5_PLLSTEN0_SHIFT), MCG_C5_PLLSTEN0_SHIFT,
MCG_C5_PLLSTEN0_WIDTH))
/*@} */

/*!
 * @name Register MCG_C5, field PLLCLKEN0[6] (RW)
 *
 * Enables the PLL independent of PLLS and enables the PLL clock for use
as
 * MCGPLLCLK. (PRDIV 0 needs to be programmed to the correct divider to
generate a

```

```

 * PLL reference clock in the range of 2 - 4 MHz range prior to setting
the
 * PLLCLKEN 0 bit). Setting PLLCLKEN 0 will enable the external
oscillator if not
 * already enabled. Whenever the PLL is being enabled by means of the
PLLCLKEN 0 bit,
 * and the external oscillator is being used as the reference clock, the
OSCINIT 0
 * bit should be checked to make sure it is set.
*
* Values:
* - 0b0 - MCGPLLCLK is inactive.
* - 0b1 - MCGPLLCLK is active.
*/
/*@{*/
/*! @brief Read current value of the MCG_C5_PLLCLKEN0 field. */
#define MCG_RD_C5_PLLCLKEN0(base) ((MCG_C5_REG(base) &
MCG_C5_PLLCLKENO_MASK) >> MCG_C5_PLLCLKENO_SHIFT)
#define MCG_BRD_C5_PLLCLKEN0(base) (BME_UBFX8(&MCG_C5_REG(base),
MCG_C5_PLLCLKENO_SHIFT, MCG_C5_PLLCLKENO_WIDTH))

/*! @brief Set the PLLCLKEN0 field to a new value. */
#define MCG_WR_C5_PLLCLKEN0(base, value) (MCG_RMW_C5(base,
MCG_C5_PLLCLKENO_MASK, MCG_C5_PLLCLKENO(value)))
#define MCG_BWR_C5_PLLCLKEN0(base, value) (BME_BFI8(&MCG_C5_REG(base),
((uint8_t)(value) << MCG_C5_PLLCLKENO_SHIFT), MCG_C5_PLLCLKENO_SHIFT,
MCG_C5_PLLCLKENO_WIDTH))
/*@}*/
/*
*****
* MCG_C6 - MCG Control 6 Register
*****
*/
/*!
* @brief MCG_C6 - MCG Control 6 Register (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire MCG_C6 register
*/
/*@{*/
#define MCG_RD_C6(base) (MCG_C6_REG(base))
#define MCG_WR_C6(base, value) (MCG_C6_REG(base) = (value))
#define MCG_RMW_C6(base, mask, value) (MCG_WR_C6(base, (MCG_RD_C6(base) &
~(mask)) | (value)))
#define MCG_SET_C6(base, value) (BME_OR8(&MCG_C6_REG(base),
(uint8_t)(value)))
#define MCG_CLR_C6(base, value) (BME_AND8(&MCG_C6_REG(base),
(uint8_t)(~(value))))

```

```

#define MCG_TOG_C6(base, value)  (BME_XOR8 (&MCG_C6_REG(base),  

(uint8_t) (value)))  

/*@}*/

/*
 * Constants & macros for individual MCG_C6 bitfields
 */

/*!!
 * @name Register MCG_C6, field VDIV0[4:0] (RW)
 *
 * Selects the amount to divide the VCO output of the PLL. The VDIV 0 bits
 * establish the multiplication factor (M) applied to the reference clock frequency.
 * After the PLL is enabled (by setting either PLLCLKEN 0 or PLLS), the VDIV 0
 * value must not be changed when LOCK 0 is zero. PLL VCO Divide Factor VDIV 0
 * Multiply Factor VDIV 0 Multiply Factor VDIV 0 Multiply Factor VDIV 0
 * Factor 00000 24 01000 32 10000 40 11000 48 00001 25 01001 33 10001 41
11001 49
 * 00010 26 01010 34 10010 42 11010 50 00011 27 01011 35 10011 43 11011
51 00100 28
 * 01100 36 10100 44 11100 52 00101 29 01101 37 10101 45 11101 53 00110
30 01110
 * 38 10110 46 11110 54 00111 31 01111 39 10111 47 11111 55
*/
/*@{*/
/*!! @brief Read current value of the MCG_C6_VDIV0 field. */
#define MCG_RD_C6_VDIV0(base) ((MCG_C6_REG(base) & MCG_C6_VDIV0_MASK) >>
MCG_C6_VDIV0_SHIFT)
#define MCG_BRD_C6_VDIV0(base) (BME_UBFX8 (&MCG_C6_REG(base),  

MCG_C6_VDIV0_SHIFT, MCG_C6_VDIV0_WIDTH))

/*!! @brief Set the VDIV0 field to a new value. */
#define MCG_WR_C6_VDIV0(base, value) (MCG_RMW_C6(base, MCG_C6_VDIV0_MASK,  

MCG_C6_VDIV0(value)))
#define MCG_BWR_C6_VDIV0(base, value) (BME_BFI8 (&MCG_C6_REG(base),  

((uint8_t)(value) << MCG_C6_VDIV0_SHIFT), MCG_C6_VDIV0_SHIFT,  

MCG_C6_VDIV0_WIDTH))
/*@}*/

/*!!
 * @name Register MCG_C6, field CME0[5] (RW)
 *
 * Enables the loss of clock monitoring circuit for the OSC0 external reference
 * mux select. The LOCRE0 bit will determine if a interrupt or a reset request is
 * generated following a loss of OSC0 indication. The CME0 bit should only be

```

```

    * set to a logic 1 when the MCG is in an operational mode that uses the
    external
    * clock (FEE, FBE, PEE, PBE, or BLPE) . Whenever the CME0 bit is set to
    a logic
    * 1, the value of the RANGE0 bits in the C2 register should not be
    changed. CME0
    * bit should be set to a logic 0 before the MCG enters any Stop mode.
    Otherwise,
    * a reset request may occur while in Stop mode. CME0 should also be set
    to a
    * logic 0 before entering VLPR or VLPW power modes if the MCG is in BLPE
    mode.
    *
    * Values:
    * - 0b0 - External clock monitor is disabled for OSC0.
    * - 0b1 - External clock monitor is enabled for OSC0.
    */
/*@{*/
/*! @brief Read current value of the MCG_C6_CME0 field. */
#define MCG_RD_C6_CME0(base) ((MCG_C6_REG(base) & MCG_C6_CME0_MASK) >>
MCG_C6_CME0_SHIFT)
#define MCG_BRD_C6_CME0(base) (BME_UBFX8(&MCG_C6_REG(base),
MCG_C6_CME0_SHIFT, MCG_C6_CME0_WIDTH))

/*! @brief Set the CME0 field to a new value. */
#define MCG_WR_C6_CME0(base, value) (MCG_RMW_C6(base, MCG_C6_CME0_MASK,
MCG_C6_CME0(value)))
#define MCG_BWR_C6_CME0(base, value) (BME_BFI8(&MCG_C6_REG(base),
((uint8_t)(value) << MCG_C6_CME0_SHIFT), MCG_C6_CME0_SHIFT,
MCG_C6_CME0_WIDTH))
/*@}*/

/*
 * @name Register MCG_C6, field PLLS[6] (RW)
 *
 * Controls whether the PLL or FLL output is selected as the MCG source
when
 * CLKS[1:0]=00. If the PLLS bit is cleared and PLLCLKEN 0 is not set,
the PLL is
 * disabled in all modes. If the PLLS is set, the FLL is disabled in all
modes.
*
* Values:
* - 0b0 - FLL is selected.
* - 0b1 - PLL is selected (PRDIV 0 need to be programmed to the correct
divider
*         to generate a PLL reference clock in the range of 2-4 MHz prior to
*         setting the PLLS bit).
*/
/*@*/
/*! @brief Read current value of the MCG_C6_PLLS field. */
#define MCG_RD_C6_PLLS(base) ((MCG_C6_REG(base) & MCG_C6_PLLS_MASK) >>
MCG_C6_PLLS_SHIFT)

```

```

#define MCG_BRD_C6_PLLS(base)  (BME_UBFX8(&MCG_C6_REG(base),  

MCG_C6_PLLS_SHIFT, MCG_C6_PLLS_WIDTH))

/*! @brief Set the PLLS field to a new value. */  

#define MCG_WR_C6_PLLS(base, value)  (MCG_RMW_C6(base, MCG_C6_PLLS_MASK,  

MCG_C6_PLLS(value)))  

#define MCG_BWR_C6_PLLS(base, value)  (BME_BFI8(&MCG_C6_REG(base),  

(uint8_t)(value) << MCG_C6_PLLS_SHIFT), MCG_C6_PLLS_SHIFT,  

MCG_C6_PLLS_WIDTH)  

/*@}*/

/*!  

 * @name Register MCG_C6, field LOLIE0[7] (RW)  

 *  

 * Determines if an interrupt request is made following a loss of lock  

 * indication. This bit only has an effect when LOLS 0 is set.  

 *  

 * Values:  

 * - 0b0 - No interrupt request is generated on loss of lock.  

 * - 0b1 - Generate an interrupt request on loss of lock.  

 */  

/*@*/  

/*! @brief Read current value of the MCG_C6_LOLIE0 field. */  

#define MCG_RD_C6_LOLIE0(base)  ((MCG_C6_REG(base) & MCG_C6_LOLIE0_MASK)  

>> MCG_C6_LOLIE0_SHIFT)  

#define MCG_BRD_C6_LOLIE0(base)  (BME_UBFX8(&MCG_C6_REG(base),  

MCG_C6_LOLIE0_SHIFT, MCG_C6_LOLIE0_WIDTH))

/*! @brief Set the LOLIE0 field to a new value. */  

#define MCG_WR_C6_LOLIE0(base, value)  (MCG_RMW_C6(base,  

MCG_C6_LOLIE0_MASK, MCG_C6_LOLIE0(value)))  

#define MCG_BWR_C6_LOLIE0(base, value)  (BME_BFI8(&MCG_C6_REG(base),  

(uint8_t)(value) << MCG_C6_LOLIE0_SHIFT), MCG_C6_LOLIE0_SHIFT,  

MCG_C6_LOLIE0_WIDTH)  

/*@*/

*****  

*****  

* MCG_S - MCG Status Register  

*****  

*****  

/*!  

 * @brief MCG_S - MCG Status Register (RW)  

 *  

 * Reset value: 0x10U  

 */  

/*!  

 * @name Constants and macros for entire MCG_S register  

 */  

/*@*/  

#define MCG_RD_S(base)          (MCG_S_REG(base))  

#define MCG_WR_S(base, value)    (MCG_S_REG(base) = (value))

```

```

#define MCG_RMW_S(base, mask, value)  (MCG_WR_S(base, (MCG_RD_S(base) &
~(mask)) | (value)))
#define MCG_SET_S(base, value)      (BME_OR8(&MCG_S_REG(base),
(uint8_t)(value)))
#define MCG_CLR_S(base, value)     (BME_AND8(&MCG_S_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_S(base, value)     (BME_XOR8(&MCG_S_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual MCG_S bitfields
 */

/*!!
 * @name Register MCG_S, field IRCST[0] (RO)
 *
 * The IRCST bit indicates the current source for the internal reference
 * clock
 * select clock (IRCSCLK). The IRCST bit does not update immediately
 * after a write
 * to the IRCS bit due to internal synchronization between clock domains.
 * The
 * * IRCST bit will only be updated if the internal reference clock is
 * enabled,
 * * either by the MCG being in a mode that uses the IRC or by setting the
 * C1[IRCLKEN]
 * * bit .
 *
 * Values:
 * - 0b0 - Source of internal reference clock is the slow clock (32 kHz
 * IRC) .
 * - 0b1 - Source of internal reference clock is the fast clock (4 MHz
 * IRC) .
 */
/*@{*/
/*!! @brief Read current value of the MCG_S_IRCST field. */
#define MCG_RD_S_IRCST(base) ((MCG_S_REG(base) & MCG_S_IRCST_MASK) >>
MCG_S_IRCST_SHIFT)
#define MCG_BRD_S_IRCST(base) (BME_UBFX8(&MCG_S_REG(base),
MCG_S_IRCST_SHIFT, MCG_S_IRCST_WIDTH))
/*@}*/

/*!!
 * @name Register MCG_S, field OSCINIT0[1] (RO)
 *
 * This bit, which resets to 0, is set to 1 after the initialization
 * cycles of
 * the crystal oscillator clock have completed. After being set, the bit
 * is
 * cleared to 0 if the OSC is subsequently disabled. See the OSC module's
 * detailed
 * description for more information.
 */

```

```

/*@{*/
/*! @brief Read current value of the MCG_S_OSCINIT0 field. */
#define MCG_RD_S_OSCINIT0(base) ((MCG_S_REG(base) & MCG_S_OSCINIT0_MASK) >> MCG_S_OSCINIT0_SHIFT)
#define MCG_BRD_S_OSCINIT0(base) (BME_UBXF8(&MCG_S_REG(base), MCG_S_OSCINIT0_SHIFT, MCG_S_OSCINIT0_WIDTH))
/*}@*/}

/*
 * @name Register MCG_S, field CLKST[3:2] (RO)
 *
 * These bits indicate the current clock mode. The CLKST bits do not update
 * immediately after a write to the CLKS bits due to internal synchronization between
 * clock domains.
 *
 * Values:
 * - 0b00 - Encoding 0 - Output of the FLL is selected (reset default).
 * - 0b01 - Encoding 1 - Internal reference clock is selected.
 * - 0b10 - Encoding 2 - External reference clock is selected.
 * - 0b11 - Encoding 3 - Output of the PLL is selected.
 */
/*@{*/
/*! @brief Read current value of the MCG_S_CLKST field. */
#define MCG_RD_S_CLKST(base) ((MCG_S_REG(base) & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT)
#define MCG_BRD_S_CLKST(base) (BME_UBXF8(&MCG_S_REG(base), MCG_S_CLKST_SHIFT, MCG_S_CLKST_WIDTH))
/*}@*/}

/*
 * @name Register MCG_S, field IREFST[4] (RO)
 *
 * This bit indicates the current source for the FLL reference clock. The IREFST
 * bit does not update immediately after a write to the IREFS bit due to
 * internal synchronization between clock domains.
 *
 * Values:
 * - 0b0 - Source of FLL reference clock is the external reference clock.
 * - 0b1 - Source of FLL reference clock is the internal reference clock.
 */
/*@{*/
/*! @brief Read current value of the MCG_S_IREFST field. */
#define MCG_RD_S_IREFST(base) ((MCG_S_REG(base) & MCG_S_IREFST_MASK) >> MCG_S_IREFST_SHIFT)
#define MCG_BRD_S_IREFST(base) (BME_UBXF8(&MCG_S_REG(base), MCG_S_IREFST_SHIFT, MCG_S_IREFST_WIDTH))
/*}@*/}

/*
 * @name Register MCG_S, field PLLST[5] (RO)
 *

```

```

 * This bit indicates the clock source selected by PLLS . The PLLST bit
does not
 * update immediately after a write to the PLLS bit due to internal
 * synchronization between clock domains.
 *
 * Values:
 * - 0b0 - Source of PLLS clock is FLL clock.
 * - 0b1 - Source of PLLS clock is PLL output clock.
 */
/*@{*/
/*! @brief Read current value of the MCG_S_PLLST field. */
#define MCG_RD_S_PLLST(base) ((MCG_S_REG(base) & MCG_S_PLLST_MASK) >>
MCG_S_PLLST_SHIFT)
#define MCG_BRD_S_PLLST(base) (BME_UBFX8(&MCG_S_REG(base),
MCG_S_PLLST_SHIFT, MCG_S_PLLST_WIDTH))
/*@}*/
/*
 * @name Register MCG_S, field LOCK0[6] (RO)
 *
 * This bit indicates whether the PLL has acquired lock. Lock detection
is only
 * enabled when the PLL is enabled (either through clock mode selection
or
 * PLLCLKEN0=1 setting). While the PLL clock is locking to the desired
frequency, the
 * MCG PLL clock (MCGPLLCLK) will be gated off until the LOCK bit gets
asserted.
 * If the lock status bit is set, changing the value of the PRDIV0 [4:0]
bits in
 * the C5 register or the VDIV0[4:0] bits in the C6 register causes the
lock
 * status bit to clear and stay cleared until the PLL has reacquired
lock. Loss of PLL
 * reference clock will also cause the LOCK0 bit to clear until the PLL
has
 * reacquired lock. Entry into LLS, VLPS, or regular Stop with PLLSTEN=0
also causes
 * the lock status bit to clear and stay cleared until the Stop mode is
exited
 * and the PLL has reacquired lock. Any time the PLL is enabled and the
LOCK0 bit
 * is cleared, the MCGPLLCLK will be gated off until the LOCK0 bit is
asserted
 * again.
 *
 * Values:
 * - 0b0 - PLL is currently unlocked.
 * - 0b1 - PLL is currently locked.
 */
/*@*/
/*! @brief Read current value of the MCG_S_LOCK0 field. */
#define MCG_RD_S_LOCK0(base) ((MCG_S_REG(base) & MCG_S_LOCK0_MASK) >>
MCG_S_LOCK0_SHIFT)

```

```

#define MCG_BRD_S_LOCK0(base)  (BME_UBFX8(&MCG_S_REG(base),  

MCG_S_LOCK0_SHIFT, MCG_S_LOCK0_WIDTH))  

/*@}*/

/*!  

 * @name Register MCG_S, field LOLS0[7] (W1C)  

 *  

 * This bit is a sticky bit indicating the lock status for the PLL. LOLS  

is set  

 * if after acquiring lock, the PLL output frequency has fallen outside  

the lock  

 * exit frequency tolerance, D unl . LOLIE determines whether an  

interrupt  

 * request is made when LOLS is set. LOLRE determines whether a reset  

request is made  

 * when LOLS is set. This bit is cleared by reset or by writing a logic 1  

to it  

 * when set. Writing a logic 0 to this bit has no effect.  

 *  

 * Values:  

 * - 0b0 - PLL has not lost lock since LOLS 0 was last cleared.  

 * - 0b1 - PLL has lost lock since LOLS 0 was last cleared.  

 */  

/*@{*/  

/*! @brief Read current value of the MCG_S_LOLS0 field. */  

#define MCG_RD_S_LOLS0(base) ((MCG_S_REG(base) & MCG_S_LOLS0_MASK) >>  

MCG_S_LOLS0_SHIFT)  

#define MCG_BRD_S_LOLS0(base) (BME_UBFX8(&MCG_S_REG(base),  

MCG_S_LOLS0_SHIFT, MCG_S_LOLS0_WIDTH))

/*! @brief Set the LOLS0 field to a new value. */  

#define MCG_WR_S_LOLS0(base, value) (MCG_RMW_S(base, MCG_S_LOLS0_MASK,  

MCG_S_LOLS0(value)))  

#define MCG_BWR_S_LOLS0(base, value) (BME_BFI8(&MCG_S_REG(base),  

((uint8_t)(value) << MCG_S_LOLS0_SHIFT), MCG_S_LOLS0_SHIFT,  

MCG_S_LOLS0_WIDTH))  

/*@}/
```

\*\*\*\*\*  
\*\*\*\*\*  
\* MCG\_SC - MCG Status and Control Register  
\*\*\*\*\*  
\*\*\*\*\* /

```

/*!  

 * @brief MCG_SC - MCG Status and Control Register (RW)  

 *  

 * Reset value: 0x02U  

 */  

/*!  

 * @name Constants and macros for entire MCG_SC register  

 */  

/*@{*/
```

```

#define MCG_RD_SC(base)           (MCG_SC_REG(base))
#define MCG_WR_SC(base, value)    (MCG_SC_REG(base) = (value))
#define MCG_RMW_SC(base, mask, value) (MCG_WR_SC(base, (MCG_RD_SC(base) &
~(mask)) | (value)))
#define MCG_SET_SC(base, value)   (BME_OR8(&MCG_SC_REG(base),
(uint8_t)(value)))
#define MCG_CLR_SC(base, value)   (BME_AND8(&MCG_SC_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_SC(base, value)   (BME_XOR8(&MCG_SC_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual MCG_SC bitfields
 */

/*!
 * @name Register MCG_SC, field LOCS0[0] (W1C)
 *
 * The LOCS0 indicates when a loss of OSC0 reference clock has occurred.
 * The
 * * LOCS0 bit only has an effect when CME0 is set. This bit is cleared by
 * writing a
 * * logic 1 to it when set.
 *
 * Values:
 * * - 0b0 - Loss of OSC0 has not occurred.
 * * - 0b1 - Loss of OSC0 has occurred.
 */
/*@{*/
/*! @brief Read current value of the MCG_SC_LOCS0 field. */
#define MCG_RD_SC_LOCS0(base) ((MCG_SC_REG(base) & MCG_SC_LOCS0_MASK) >>
MCG_SC_LOCS0_SHIFT)
#define MCG_BRD_SC_LOCS0(base) (BME_UBFX8(&MCG_SC_REG(base),
MCG_SC_LOCS0_SHIFT, MCG_SC_LOCS0_WIDTH))

/*! @brief Set the LOCS0 field to a new value. */
#define MCG_WR_SC_LOCS0(base, value) (MCG_RMW_SC(base, MCG_SC_LOCS0_MASK,
MCG_SC_LOCS0(value)))
#define MCG_BWR_SC_LOCS0(base, value) (BME_BFI8(&MCG_SC_REG(base),
((uint8_t)(value) << MCG_SC_LOCS0_SHIFT), MCG_SC_LOCS0_SHIFT,
MCG_SC_LOCS0_WIDTH))
/*@}*/

/*
 * @name Register MCG_SC, field FCRDIV[3:1] (RW)
 *
 * Selects the amount to divide down the fast internal reference clock.
 * The
 * * resulting frequency will be in the range 31.25 kHz to 4 MHz (Note:
 * Changing the
 * * divider when the Fast IRC is enabled is not supported).
 *
 * Values:

```

```

* - 0b000 - Divide Factor is 1
* - 0b001 - Divide Factor is 2.
* - 0b010 - Divide Factor is 4.
* - 0b011 - Divide Factor is 8.
* - 0b100 - Divide Factor is 16
* - 0b101 - Divide Factor is 32
* - 0b110 - Divide Factor is 64
* - 0b111 - Divide Factor is 128.
*/
/*@{*/
/*! @brief Read current value of the MCG_SC_FCRDIV field. */
#define MCG_RD_SC_FCRDIV(base) ((MCG_SC_REG(base) & MCG_SC_FCRDIV_MASK)
>> MCG_SC_FCRDIV_SHIFT)
#define MCG_BRD_SC_FCRDIV(base) (BME_UBXF8(&MCG_SC_REG(base),
MCG_SC_FCRDIV_SHIFT, MCG_SC_FCRDIV_WIDTH))

/*! @brief Set the FCRDIV field to a new value. */
#define MCG_WR_SC_FCRDIV(base, value) (MCG_RMW_SC(base,
(MCG_SC_FCRDIV_MASK | MCG_SC_LOCS0_MASK), MCG_SC_FCRDIV(value)))
#define MCG_BWR_SC_FCRDIV(base, value) (BME_BFI8(&MCG_SC_REG(base),
((uint8_t)(value) << MCG_SC_FCRDIV_SHIFT), MCG_SC_FCRDIV_SHIFT,
MCG_SC_FCRDIV_WIDTH))
/*}@*/
```

/\*!

- \* @name Register MCG\_SC, field FLTPRSRV[4] (RW)
- \*
- \* This bit will prevent the FLL filter values from resetting allowing the FLL
- \* output frequency to remain the same during clock mode changes where the FLL/DCO
- \* output is still valid. (Note: This requires that the FLL reference frequency
- \* to remain the same as what it was prior to the new clock mode switch.
- \* Otherwise FLL filter and frequency values will change.)
- \*
- \* Values:
- \* - 0b0 - FLL filter and FLL frequency will reset on changes to current clock
- \* mode.
- \* - 0b1 - Fll filter and FLL frequency retain their previous values during new
- \* clock mode change.

/\*}@\*/
/\*! @brief Read current value of the MCG\_SC\_FLTPRSRV field. \*/
#define MCG\_RD\_SC\_FLTPRSRV(base) ((MCG\_SC\_REG(base) &
MCG\_SC\_FLTPRSRV\_MASK) >> MCG\_SC\_FLTPRSRV\_SHIFT)
#define MCG\_BRD\_SC\_FLTPRSRV(base) (BME\_UBXF8(&MCG\_SC\_REG(base),
MCG\_SC\_FLTPRSRV\_SHIFT, MCG\_SC\_FLTPRSRV\_WIDTH))

/\*! @brief Set the FLTPRSRV field to a new value. \*/
#define MCG\_WR\_SC\_FLTPRSRV(base, value) (MCG\_RMW\_SC(base,
(MCG\_SC\_FLTPRSRV\_MASK | MCG\_SC\_LOCS0\_MASK), MCG\_SC\_FLTPRSRV(value)))

```

#define MCG_BWR_SC_FLTPRSRV(base, value) (BME_BFI8(&MCG_SC_REG(base), \
((uint8_t)(value) << MCG_SC_FLTPRSRV_SHIFT), MCG_SC_FLTPRSRV_SHIFT, \
MCG_SC_FLTPRSRV_WIDTH))
/*@}*/

/*
 * @name Register MCG_SC, field ATMF[5] (RW)
 *
 * Fail flag for the Automatic Trim Machine (ATM). This bit asserts when
the
 * Automatic Trim Machine is enabled, ATME=1, and a write to the C1, C3,
C4, and SC
 * registers is detected or the MCG enters into any Stop mode. A write to
ATMF
 * clears the flag.
*
* Values:
* - 0b0 - Automatic Trim Machine completed normally.
* - 0b1 - Automatic Trim Machine failed.
*/
/*@*/
/*! @brief Read current value of the MCG_SC_ATMF field. */
#define MCG_RD_SC_ATMF(base) ((MCG_SC_REG(base) & MCG_SC_ATMF_MASK) >>
MCG_SC_ATMF_SHIFT)
#define MCG_BRD_SC_ATMF(base) (BME_UBFX8(&MCG_SC_REG(base), \
MCG_SC_ATMF_SHIFT, MCG_SC_ATMF_WIDTH))

/*! @brief Set the ATMF field to a new value. */
#define MCG_WR_SC_ATMF(base, value) (MCG_RMW_SC(base, (MCG_SC_ATMF_MASK | \
MCG_SC_LOCS0_MASK), MCG_SC_ATMF(value)))
#define MCG_BWR_SC_ATMF(base, value) (BME_BFI8(&MCG_SC_REG(base), \
((uint8_t)(value) << MCG_SC_ATMF_SHIFT), MCG_SC_ATMF_SHIFT, \
MCG_SC_ATMF_WIDTH))
/*@*/
```

```

/*
 * @name Register MCG_SC, field ATMS[6] (RW)
*
* Selects the IRCS clock for Auto Trim Test.
*
* Values:
* - 0b0 - 32 kHz Internal Reference Clock selected.
* - 0b1 - 4 MHz Internal Reference Clock selected.
*/
/*@*/
/*! @brief Read current value of the MCG_SC_ATMS field. */
#define MCG_RD_SC_ATMS(base) ((MCG_SC_REG(base) & MCG_SC_ATMS_MASK) >>
MCG_SC_ATMS_SHIFT)
#define MCG_BRD_SC_ATMS(base) (BME_UBFX8(&MCG_SC_REG(base), \
MCG_SC_ATMS_SHIFT, MCG_SC_ATMS_WIDTH))

/*! @brief Set the ATMS field to a new value. */
#define MCG_WR_SC_ATMS(base, value) (MCG_RMW_SC(base, (MCG_SC_ATMS_MASK | \
MCG_SC_LOCS0_MASK), MCG_SC_ATMS(value)))
```

```

#define MCG_BWR_SC_ATMS(base, value) (BME_BFI8(&MCG_SC_REG(base),  

((uint8_t)(value) << MCG_SC_ATMS_SHIFT), MCG_SC_ATMS_SHIFT,  

MCG_SC_ATMS_WIDTH))  

/*@}*/

/*!  

 * @name Register MCG_SC, field ATME[7] (RW)  

 *  

 * Enables the Auto Trim Machine to start automatically trimming the  

selected  

 * Internal Reference Clock. ATME deasserts after the Auto Trim Machine  

has  

 * completed trimming all trim bits of the IRCS clock selected by the  

ATMS bit. Writing  

 * to C1, C3, C4, and SC registers or entering Stop mode aborts the auto  

trim  

 * operation and clears this bit.  

 *  

 * Values:  

 * - 0b0 - Auto Trim Machine disabled.  

 * - 0b1 - Auto Trim Machine enabled.  

 */  

/*@{*/  

/*! @brief Read current value of the MCG_SC_ATME field. */  

#define MCG_RD_SC_ATME(base) ((MCG_SC_REG(base) & MCG_SC_ATME_MASK) >>  

MCG_SC_ATME_SHIFT)  

#define MCG_BRD_SC_ATME(base) (BME_UBFX8(&MCG_SC_REG(base),  

MCG_SC_ATME_SHIFT, MCG_SC_ATME_WIDTH))

/*! @brief Set the ATME field to a new value. */  

#define MCG_WR_SC_ATME(base, value) (MCG_RMW_SC(base, (MCG_SC_ATME_MASK |  

MCG_SC_LOCS0_MASK), MCG_SC_ATME(value)))  

#define MCG_BWR_SC_ATME(base, value) (BME_BFI8(&MCG_SC_REG(base),  

((uint8_t)(value) << MCG_SC_ATME_SHIFT), MCG_SC_ATME_SHIFT,  

MCG_SC_ATME_WIDTH))  

/*@}*/

*****  

*****  

 * MCG_ATCVH - MCG Auto Trim Compare Value High Register  

*****  

*****  

/*!  

 * @brief MCG_ATCVH - MCG Auto Trim Compare Value High Register (RW)  

 *  

 * Reset value: 0x00U  

 */  

/*!  

 * @name Constants and macros for entire MCG_ATCVH register  

 */  

/*@{*/  

#define MCG_RD_ATCVH(base) (MCG_ATCVH_REG(base))

```

```

#define MCG_WR_ATCVH(base, value) (MCG_ATCVH_REG(base) = (value))
#define MCG_RMW_ATCVH(base, mask, value) (MCG_WR_ATCVH(base,
(MCG_RD_ATCVH(base) & ~mask)) | (value)))
#define MCG_SET_ATCVH(base, value) (BME_OR8(&MCG_ATCVH_REG(base),
(uint8_t)(value)))
#define MCG_CLR_ATCVH(base, value) (BME_AND8(&MCG_ATCVH_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_ATCVH(base, value) (BME_XOR8(&MCG_ATCVH_REG(base),
(uint8_t)(value)))
/*@}*/

/******************
*****
 * MCG_ATCVL - MCG Auto Trim Compare Value Low Register
*****
*******/

/*!
 * @brief MCG_ATCVL - MCG Auto Trim Compare Value Low Register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire MCG_ATCVL register
 */
/*@{*/
#define MCG_RD_ATCVL(base) (MCG_ATCVL_REG(base))
#define MCG_WR_ATCVL(base, value) (MCG_ATCVL_REG(base) = (value))
#define MCG_RMW_ATCVL(base, mask, value) (MCG_WR_ATCVL(base,
(MCG_RD_ATCVL(base) & ~mask)) | (value)))
#define MCG_SET_ATCVL(base, value) (BME_OR8(&MCG_ATCVL_REG(base),
(uint8_t)(value)))
#define MCG_CLR_ATCVL(base, value) (BME_AND8(&MCG_ATCVL_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_ATCVL(base, value) (BME_XOR8(&MCG_ATCVL_REG(base),
(uint8_t)(value)))
/*@}*/

/******************
*****
 * MCG_C7 - MCG Control 7 Register
*****
*******/

/*!
 * @brief MCG_C7 - MCG Control 7 Register (ROZ)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire MCG_C7 register
 */

```

```

/*@{ */
#define MCG_RD_C7(base)           (MCG_C7_REG(base) )
/*}@*/



/***** ****
 * MCG_C8 - MCG Control 8 Register
***** */

/*!!
 * @brief MCG_C8 - MCG Control 8 Register (RW)
 *
 * Reset value: 0x80U
 */
/*!!
 * @name Constants and macros for entire MCG_C8 register
 */
/*@{ */

#define MCG_RD_C8(base)           (MCG_C8_REG(base) )
#define MCG_WR_C8(base, value)    (MCG_C8_REG(base) = (value))
#define MCG_RMW_C8(base, mask, value) (MCG_WR_C8(base, (MCG_RD_C8(base) &
~(mask)) | (value)))
#define MCG_SET_C8(base, value)   (BME_OR8(&MCG_C8_REG(base),
(uint8_t)(value)))
#define MCG_CLR_C8(base, value)   (BME_AND8(&MCG_C8_REG(base),
(uint8_t)(~(value))))
#define MCG_TOG_C8(base, value)   (BME_XOR8(&MCG_C8_REG(base),
(uint8_t)(value)))
/*}@*/



/*
 * Constants & macros for individual MCG_C8 bitfields
 */

/*!!
 * @name Register MCG_C8, field LOLRE[6] (RW)
 *
 * Determines if a interrupt or a reset request is made following a PLL
loss of
 * lock.
 *
 * Values:
 * - 0b0 - Interrupt request is generated on a PLL loss of lock
indication. The
 *       PLL loss of lock interrupt enable bit must also be set to generate
the
 *       interrupt request.
 * - 0b1 - Generate a reset request on a PLL loss of lock indication.
 */
/*@{ */
/*!! @brief Read current value of the MCG_C8_LOLRE field. */

```

```

#define MCG_RD_C8_LOLRE(base) ((MCG_C8_REG(base) & MCG_C8_LOLRE_MASK) >>
MCG_C8_LOLRE_SHIFT)
#define MCG_BRD_C8_LOLRE(base) (BME_UBFX8(&MCG_C8_REG(base),
MCG_C8_LOLRE_SHIFT, MCG_C8_LOLRE_WIDTH))

/*! @brief Set the LOLRE field to a new value. */
#define MCG_WR_C8_LOLRE(base, value) (MCG_RMW_C8(base, MCG_C8_LOLRE_MASK,
MCG_C8_LOLRE(value)))
#define MCG_BWR_C8_LOLRE(base, value) (BME_BFI8(&MCG_C8_REG(base),
((uint8_t)(value) << MCG_C8_LOLRE_SHIFT), MCG_C8_LOLRE_SHIFT,
MCG_C8_LOLRE_WIDTH))
/*@}*/

/********************* MCG Control 9 Register ********************/
***** */

/*! 
 * @brief MCG_C9 - MCG Control 9 Register (ROZ)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire MCG_C9 register
 */
/*@{ */
#define MCG_RD_C9(base) (MCG_C9_REG(base))
/*@}*/

/********************* MCG Control 10 Register ********************/
***** */

/*! 
 * @brief MCG_C10 - MCG Control 10 Register (ROZ)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire MCG_C10 register
 */
/*@{ */
#define MCG_RD_C10(base) (MCG_C10_REG(base))
/*@}*/

/* MCG C2[EREFS] backward compatibility */
#define MCG_RD_C2_EREFS(base) (MCG_RD_C2_EREFS0(base))
#define MCG_BRD_C2_EREFS(base) (MCG_BRD_C2_EREFS0(base))

```

```

#define MCG_WR_C2_EREFS(base, value)  (MCG_WR_C2_EREFS0((base), (value)))
#define MCG_BWR_C2_EREFS(base, value)  (MCG_BWR_C2_EREFS0((base),
(value)))
/* MCG C2[HGO] backward compatibility */
#define MCG_RD_C2_HGO(base)          (MCG_RD_C2_HGO0(base))
#define MCG_BRD_C2_HGO(base)         (MCG_BRD_C2_HGO0(base))
#define MCG_WR_C2_HGO(base, value)   (MCG_WR_C2_HGO0((base), (value)))
#define MCG_BWR_C2_HGO(base, value)  (MCG_BWR_C2_HGO0((base), (value)))
/* MCG C2[RANGE] backward compatibility */
#define MCG_RD_C2_RANGE(base)        (MCG_RD_C2_RANGE0(base))
#define MCG_BRD_C2_RANGE(base)       (MCG_BRD_C2_RANGE0(base))
#define MCG_WR_C2_RANGE(base, value) (MCG_WR_C2_RANGE0((base), (value)))
#define MCG_BWR_C2_RANGE(base, value) (MCG_BWR_C2_RANGE0((base),
(value)))

/*
 * MKL25Z4 MCM
 *
 * Core Platform Miscellaneous Control Module
 *
 * Registers defined in this header file:
 * - MCM_PLASC - Crossbar Switch (AXBS) Slave Configuration
 * - MCM_PLAMC - Crossbar Switch (AXBS) Master Configuration
 * - MCM_PLACR - Platform Control Register
 * - MCM_CPO - Compute Operation Control Register
 */

#define MCM_INSTANCE_COUNT (1U) /*!< Number of instances of the MCM
module. */
#define MCM_IDX (0U) /*!< Instance number for MCM. */

/********************* MCM_PLASC - Crossbar Switch (AXBS) Slave Configuration *****/
**** */
* MCM_PLASC - Crossbar Switch (AXBS) Slave Configuration
**** */
/*!
* @brief MCM_PLASC - Crossbar Switch (AXBS) Slave Configuration (RO)
*
* Reset value: 0x0007U
*
* PLASC is a 16-bit read-only register identifying the presence/absence
of bus
* slave connections to the device's crossbar switch.
*/
/*!
* @name Constants and macros for entire MCM_PLASC register
*/
/*@{ */
#define MCM_RD_PLASC(base)          (MCM_PLASC_REG(base))
/*}@*/

```

```

/*
 * Constants & macros for individual MCM_PLASC bitfields
 */

/*!
 * @name Register MCM_PLASC, field ASC[7:0] (RO)
 *
 * Values:
 * - 0b00000000 - A bus slave connection to AXBS input port n is absent
 * - 0b00000001 - A bus slave connection to AXBS input port n is present
 */
/*@{ */

/*! @brief Read current value of the MCM_PLASC_ASC field. */
#define MCM_RD_PLASC_ASC(base) ((MCM_PLASC_REG(base) &
MCM_PLASC_ASC_MASK) >> MCM_PLASC_ASC_SHIFT)
#define MCM_BRD_PLASC_ASC(base) (MCM_RD_PLASC_ASC(base))
/*}@*/



/********************* ****
**** * MCM_PLAMC - Crossbar Switch (AXBS) Master Configuration
***** ****
****/


/*!
 * @brief MCM_PLAMC - Crossbar Switch (AXBS) Master Configuration (RO)
 *
 * Reset value: 0x000DU
 *
 * PLAMC is a 16-bit read-only register identifying the presence/absence
of bus
 * master connections to the device's crossbar switch.
 */
/*!
 * @name Constants and macros for entire MCM_PLAMC register
 */
/*@{ */

#define MCM_RD_PLAMC(base) (MCM_PLAMC_REG(base) )
/*}@*/



/*
 * Constants & macros for individual MCM_PLAMC bitfields
 */

/*!
 * @name Register MCM_PLAMC, field AMC[7:0] (RO)
 *
 * Values:
 * - 0b00000000 - A bus master connection to AXBS input port n is absent
 * - 0b00000001 - A bus master connection to AXBS input port n is present
 */
/*@{ */

/*! @brief Read current value of the MCM_PLAMC_AMC field. */

```

```

#define MCM_RD_PLAMC_AMC(base) ((MCM_PLAMC_REG(base) &
MCM_PLAMC_AMC_MASK) >> MCM_PLAMC_AMC_SHIFT)
#define MCM_BRD_PLAMC_AMC(base) (MCM_RD_PLAMC_AMC(base))
/*@}*/

/*********************  

*****  

 * MCM_PLACR - Platform Control Register  

*****/  

  

/*!  

 * @brief MCM_PLACR - Platform Control Register (RW)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The PLACR register selects the arbitration policy for the crossbar  

masters  

 * and configures the flash memory controller. The speculation buffer and  

cache in  

 * the flash memory controller is configurable via MCM_PLACR[15:10]. The  

 * speculation buffer is enabled only for instructions after reset. It is  

possible to  

 * have these states for the speculation buffer: DFCS EFDS Description 0  

0  

 * Speculation buffer is on for instruction and off for data. 0 1  

Speculation buffer is on  

 * for instruction and on for data. 1 X Speculation buffer is off. The  

cache in  

 * flash controller is enabled and caching both instruction and data type  

fetches  

 * after reset. It is possible to have these states for the cache: DFCC  

DFCIC  

 * DFCDA Description 0 0 0 Cache is on for both instruction and data. 0 0  

1 Cache  

 * is on for instruction and off for data. 0 1 0 Cache is off for  

instruction and  

 * on for data. 0 1 1 Cache is off for both instruction and data. 1 X X  

Cache is  

 * off.  

 */  

/*!  

 * @name Constants and macros for entire MCM_PLACR register  

*/  

/*@*/
#define MCM_RD_PLACR(base) (MCM_PLACR_REG(base))
#define MCM_WR_PLACR(base, value) (MCM_PLACR_REG(base) = (value))
#define MCM_RMW_PLACR(base, mask, value) (MCM_WR_PLACR(base,
(MCM_RD_PLACR(base) & ~mask)) | (value)))
#define MCM_SET_PLACR(base, value) (MCM_WR_PLACR(base, MCM_RD_PLACR(base)
| (value)))
#define MCM_CLR_PLACR(base, value) (MCM_WR_PLACR(base, MCM_RD_PLACR(base)
& ~value)))

```

```

#define MCM_TOG_PLACR(base, value) (MCM_WR_PLACR(base, MCM_RD_PLACR(base)
^ (value)))
/*@}*/

/*
 * Constants & macros for individual MCM_PLACR bitfields
 */

/*!!
 * @name Register MCM_PLACR, field ARB[9] (RW)
 *
 * Values:
 * - 0b0 - Fixed-priority arbitration for the crossbar masters
 * - 0b1 - Round-robin arbitration for the crossbar masters
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_ARB field. */
#define MCM_RD_PLACR_ARB(base) ((MCM_PLACR_REG(base) &
MCM_PLACR_ARB_MASK) >> MCM_PLACR_ARB_SHIFT)
#define MCM_BRD_PLACR_ARB(base) (MCM_RD_PLACR_ARB(base))

/*! @brief Set the ARB field to a new value. */
#define MCM_WR_PLACR_ARB(base, value) (MCM_RMW_PLACR(base,
MCM_PLACR_ARB_MASK, MCM_PLACR_ARB(value)))
#define MCM_BWR_PLACR_ARB(base, value) (MCM_WR_PLACR_ARB(base, value))
/*@*/ */

/*!!
 * @name Register MCM_PLACR, field CFCC[10] (WORZ)
 *
 * Writing a 1 to this field clears the cache. Writing a 0 to this field
is
 * ignored. This field always reads as 0.
 */
/*@{*/
/*! @brief Set the CFCC field to a new value. */
#define MCM_WR_PLACR_CFCC(base, value) (MCM_RMW_PLACR(base,
MCM_PLACR_CFCC_MASK, MCM_PLACR_CFCC(value)))
#define MCM_BWR_PLACR_CFCC(base, value) (MCM_WR_PLACR_CFCC(base, value))
/*@*/ */

/*!!
 * @name Register MCM_PLACR, field DFCDA[11] (RW)
 *
 * This field is used to disable flash controller data caching.
 *
 * Values:
 * - 0b0 - Enable flash controller data caching
 * - 0b1 - Disable flash controller data caching.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_DFCDA field. */
#define MCM_RD_PLACR_DFCDA(base) ((MCM_PLACR_REG(base) &
MCM_PLACR_DFCDA_MASK) >> MCM_PLACR_DFCDA_SHIFT)

```

```

#define MCM_BRD_PLACR_DFCDA(base)  (MCM_RD_PLACR_DFCDA(base))

/*! @brief Set the DFCDA field to a new value. */
#define MCM_WR_PLACR_DFCDA(base, value)  (MCM_RMW_PLACR(base,
MCM_PLACR_DFCDA_MASK, MCM_PLACR_DFCDA(value)))
#define MCM_BWR_PLACR_DFCDA(base, value)  (MCM_WR_PLACR_DFCDA(base,
value))
/*@}*/

/*!!
 * @name Register MCM_PLACR, field DFCIC[12] (RW)
 *
 * This field is used to disable flash controller instruction caching.
 *
 * Values:
 * - 0b0 - Enable flash controller instruction caching.
 * - 0b1 - Disable flash controller instruction caching.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_DFCIC field. */
#define MCM_RD_PLACR_DFCIC(base)  ((MCM_PLACR_REG(base) &
MCM_PLACR_DFCIC_MASK) >> MCM_PLACR_DFCIC_SHIFT)
#define MCM_BRD_PLACR_DFCIC(base)  (MCM_RD_PLACR_DFCIC(base))

/*! @brief Set the DFCIC field to a new value. */
#define MCM_WR_PLACR_DFCIC(base, value)  (MCM_RMW_PLACR(base,
MCM_PLACR_DFCIC_MASK, MCM_PLACR_DFCIC(value)))
#define MCM_BWR_PLACR_DFCIC(base, value)  (MCM_WR_PLACR_DFCIC(base,
value))
/*@}*/

/*!!
 * @name Register MCM_PLACR, field DFCC[13] (RW)
 *
 * This field is used to disable flash controller cache.
 *
 * Values:
 * - 0b0 - Enable flash controller cache.
 * - 0b1 - Disable flash controller cache.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_DFCC field. */
#define MCM_RD_PLACR_DFCC(base)  ((MCM_PLACR_REG(base) &
MCM_PLACR_DFCC_MASK) >> MCM_PLACR_DFCC_SHIFT)
#define MCM_BRD_PLACR_DFCC(base)  (MCM_RD_PLACR_DFCC(base))

/*! @brief Set the DFCC field to a new value. */
#define MCM_WR_PLACR_DFCC(base, value)  (MCM_RMW_PLACR(base,
MCM_PLACR_DFCC_MASK, MCM_PLACR_DFCC(value)))
#define MCM_BWR_PLACR_DFCC(base, value)  (MCM_WR_PLACR_DFCC(base, value))
/*@}*/

/*!!
 * @name Register MCM_PLACR, field EFDS[14] (RW)

```

```

/*
 * This field is used to enable flash data speculation.
 *
 * Values:
 * - 0b0 - Disable flash data speculation.
 * - 0b1 - Enable flash data speculation.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_EFDS field. */
#define MCM_RD_PLACR_EFDS(base) ((MCM_PLACR_REG(base) &
MCM_PLACR_EFDS_MASK) >> MCM_PLACR_EFDS_SHIFT)
#define MCM_BRD_PLACR_EFDS(base) (MCM_RD_PLACR_EFDS(base))

/*! @brief Set the EFDS field to a new value. */
#define MCM_WR_PLACR_EFDS(base, value) (MCM_RMW_PLACR(base,
MCM_PLACR_EFDS_MASK, MCM_PLACR_EFDS(value)))
#define MCM_BWR_PLACR_EFDS(base, value) (MCM_WR_PLACR_EFDS(base, value))
/*}@*/
```

```

/*!
 * @name Register MCM_PLACR, field DFCS[15] (RW)
 *
 * This field is used to disable flash controller speculation.
 *
 * Values:
 * - 0b0 - Enable flash controller speculation.
 * - 0b1 - Disable flash controller speculation.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_DFCS field. */
#define MCM_RD_PLACR_DFCS(base) ((MCM_PLACR_REG(base) &
MCM_PLACR_DFCS_MASK) >> MCM_PLACR_DFCS_SHIFT)
#define MCM_BRD_PLACR_DFCS(base) (MCM_RD_PLACR_DFCS(base))

/*! @brief Set the DFCS field to a new value. */
#define MCM_WR_PLACR_DFCS(base, value) (MCM_RMW_PLACR(base,
MCM_PLACR_DFCS_MASK, MCM_PLACR_DFCS(value)))
#define MCM_BWR_PLACR_DFCS(base, value) (MCM_WR_PLACR_DFCS(base, value))
/*}@*/
```

```

/*!
 * @name Register MCM_PLACR, field ESFC[16] (RW)
 *
 * This field is used to enable stalling flash controller when flash is
busy.
 *
 * Values:
 * - 0b0 - Disable stalling flash controller when flash is busy.
 * - 0b1 - Enable stalling flash controller when flash is busy.
 */
/*@{*/
/*! @brief Read current value of the MCM_PLACR_ESFC field. */
#define MCM_RD_PLACR_ESFC(base) ((MCM_PLACR_REG(base) &
MCM_PLACR_ESFC_MASK) >> MCM_PLACR_ESFC_SHIFT)
```

```

#define MCM_BRD_PLACR_ESFC(base)  (MCM_RD_PLACR_ESFC(base))

/*! @brief Set the ESFC field to a new value. */
#define MCM_WR_PLACR_ESFC(base, value)  (MCM_RMW_PLACR(base,
MCM_PLACR_ESFC_MASK, MCM_PLACR_ESFC(value)))
#define MCM_BWR_PLACR_ESFC(base, value)  (MCM_WR_PLACR_ESFC(base, value))
/*@}*/

/***** ****
 * MCM_CPO - Compute Operation Control Register
*****
 * @name MCM_CPO - Compute Operation Control Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * This register controls the Compute Operation.
*/
/*!
 * @name Constants and macros for entire MCM_CPO register
 */
/*@{*/
#define MCM_RD_CPO(base)          (MCM_CPO_REG(base))
#define MCM_WR_CPO(base, value)   (MCM_CPO_REG(base) = (value))
#define MCM_RMW_CPO(base, mask, value)  (MCM_WR_CPO(base,
(MCM_RD_CPO(base) & ~mask) | (value)))
#define MCM_SET_CPO(base, value)   (MCM_WR_CPO(base, MCM_RD_CPO(base) | (value)))
#define MCM_CLR_CPO(base, value)   (MCM_WR_CPO(base, MCM_RD_CPO(base) &
~(value)))
#define MCM_TOG_CPO(base, value)   (MCM_WR_CPO(base, MCM_RD_CPO(base) ^ (value)))
/*@}*/

/*
 * Constants & macros for individual MCM_CPO bitfields
 */
/*!
 * @name Register MCM_CPO, field CPOREQ[0] (RW)
 *
 * This bit is auto-cleared by vector fetching if CPOWOI = 1.
 *
 * Values:
 * - 0b0 - Request is cleared.
 * - 0b1 - Request Compute Operation.
*/
/*@*/
/*! @brief Read current value of the MCM_CPO_CPOREQ field. */

```

```

#define MCM_RD_CPO_CPOREQ(base) ((MCM_CPO_REG(base) &
MCM_CPO_CPOREQ_MASK) >> MCM_CPO_CPOREQ_SHIFT)
#define MCM_BRD_CPO_CPOREQ(base) (MCM_RD_CPO_CPOREQ(base))

/*! @brief Set the CPOREQ field to a new value. */
#define MCM_WR_CPO_CPOREQ(base, value) (MCM_RMW_CPO(base,
MCM_CPO_CPOREQ_MASK, MCM_CPO_CPOREQ(value)))
#define MCM_BWR_CPO_CPOREQ(base, value) (MCM_WR_CPO_CPOREQ(base, value))
/*@}*/

/*!!
 * @name Register MCM_CPO, field CPOACK[1] (RO)
 *
 * Values:
 * - 0b0 - Compute operation entry has not completed or compute operation
exit
 *      has completed.
 * - 0b1 - Compute operation entry has completed or compute operation
exit has
 *      not completed.
 */
/*@{*/
/*! @brief Read current value of the MCM_CPO_CPOACK field. */
#define MCM_RD_CPO_CPOACK(base) ((MCM_CPO_REG(base) &
MCM_CPO_CPOACK_MASK) >> MCM_CPO_CPOACK_SHIFT)
#define MCM_BRD_CPO_CPOACK(base) (MCM_RD_CPO_CPOACK(base))
/*@}*/

/*!!
 * @name Register MCM_CPO, field CPOWOI[2] (RW)
 *
 * Values:
 * - 0b0 - No effect.
 * - 0b1 - When set, the CPOREQ is cleared on any interrupt or exception
vector
 *      fetch.
 */
/*@{*/
/*! @brief Read current value of the MCM_CPO_CPOWOI field. */
#define MCM_RD_CPO_CPOWOI(base) ((MCM_CPO_REG(base) &
MCM_CPO_CPOWOI_MASK) >> MCM_CPO_CPOWOI_SHIFT)
#define MCM_BRD_CPO_CPOWOI(base) (MCM_RD_CPO_CPOWOI(base))

/*! @brief Set the CPOWOI field to a new value. */
#define MCM_WR_CPO_CPOWOI(base, value) (MCM_RMW_CPO(base,
MCM_CPO_CPOWOI_MASK, MCM_CPO_CPOWOI(value)))
#define MCM_BWR_CPO_CPOWOI(base, value) (MCM_WR_CPO_CPOWOI(base, value))
/*@}*/

/*
 * MKL25Z4 MTB
 *
 * Micro Trace Buffer
 */

```

```

* Registers defined in this header file:
* - MTB_POSITION - MTB Position Register
* - MTB_MASTER - MTB Master Register
* - MTB_FLOW - MTB Flow Register
* - MTB_BASE - MTB Base Register
* - MTB_MODECTRL - Integration Mode Control Register
* - MTB_TAGSET - Claim TAG Set Register
* - MTB_TAGCLEAR - Claim TAG Clear Register
* - MTB_LOCKACCESS - Lock Access Register
* - MTB_LOCKSTAT - Lock Status Register
* - MTB_AUTHSTAT - Authentication Status Register
* - MTB_DEVICEARCH - Device Architecture Register
* - MTB_DEVICECFG - Device Configuration Register
* - MTB_DEVICETYPID - Device Type Identifier Register
* - MTB_PERIPHID - Peripheral ID Register
* - MTB_COMPID - Component ID Register
*/
#define MTB_INSTANCE_COUNT (1U) /*!< Number of instances of the MTB
module. */
#define MTB_IDX (0U) /*!< Instance number for MTB. */

/********************* MTB POSITION Register ********************/
**** */
* MTB_POSITION - MTB Position Register
**** */
/*!
* @brief MTB_POSITION - MTB Position Register (RW)
*
* Reset value: 0x00000000U
*
* The MTB_POSITION register is the trace write address pointer and wrap
bit.
* This register can be modified by the explicit programming model
writes. It is
* also automatically updated by the MTB hardware when trace packets are
being
* recorded.
*/
/*!
* @name Constants and macros for entire MTB_POSITION register
*/
/*@{ */
#define MTB_RD_POSITION(base)      (MTB_POSITION_REG(base))
#define MTB_WR_POSITION(base, value) (MTB_POSITION_REG(base) = (value))
#define MTB_RMW_POSITION(base, mask, value) (MTB_WR_POSITION(base,
(MTB_RD_POSITION(base) & ~mask) | value)))
#define MTB_SET_POSITION(base, value) (MTB_WR_POSITION(base,
MTB_RD_POSITION(base) | value)))
#define MTB_CLR_POSITION(base, value) (MTB_WR_POSITION(base,
MTB_RD_POSITION(base) & ~value)))

```

```

#define MTB_TOG_POSITION(base, value) (MTB_WR_POSITION(base,
MTB_RD_POSITION(base) ^ (value)))
/*@}*/

/*
 * Constants & macros for individual MTB_POSITION bitfields
 */

/*!!
 * @name Register MTB_POSITION, field WRAP[2] (RW)
 *
 * This bit is set to 1 automatically when the POINTER value wraps as
determined
 * by the MTB_MASTER[MASK] bit in the MASTER Trace Control Register. A
debug
 * agent can use the WRAP bit to determine whether the trace information
above and
 * below the pointer address is valid.
 */
/*@{*/
/*! @brief Read current value of the MTB_POSITION_WRAP field. */
#define MTB_RD_POSITION_WRAP(base) ((MTB_POSITION_REG(base) &
MTB_POSITION_WRAP_MASK) >> MTB_POSITION_WRAP_SHIFT)
#define MTB_BRD_POSITION_WRAP(base) (MTB_RD_POSITION_WRAP(base))

/*! @brief Set the WRAP field to a new value. */
#define MTB_WR_POSITION_WRAP(base, value) (MTB_RMW_POSITION(base,
MTB_POSITION_WRAP_MASK, MTB_POSITION_WRAP(value)))
#define MTB_BWR_POSITION_WRAP(base, value) (MTB_WR_POSITION_WRAP(base,
value))
/*@}*/

/*!!
 * @name Register MTB_POSITION, field POINTER[31:3] (RW)
 *
 * Trace packet address pointer. Because a packet consists of two words,
the
 * POINTER field is the address of the first word of a packet. This field
contains
 * bits[31:3] of the RAM address that points to the next unused memory
location
 * for the trace data. This is an empty ascending location and is
automatically
 * updated. A debug agent can add the value of POINTER to the value of
MTB_BASE to
 * obtain the absolute pointer address as seen on the system AHB bus
interface.
 * The size of the RAM is parameterized and the most significant bits of
the
 * POINTER field are RAZ/WI. POSITION register bits greater than or equal
to 15 are
 * RAZ/WI, therefore, the active POINTER field bits are [11:0].
 */
/*@{*/

```

```

/*! @brief Read current value of the MTB_POSITION_POINTER field. */
#define MTB_RD_POSITION_POINTER(base) ((MTB_POSITION_REG(base) &
MTB_POSITION_POINTER_MASK) >> MTB_POSITION_POINTER_SHIFT)
#define MTB_BRD_POSITION_POINTER(base) (MTB_RD_POSITION_POINTER(base))

/*! @brief Set the POSITION field to a new value. */
#define MTB_WR_POSITION_POINTER(base, value) (MTB_RMW_POSITION(base,
MTB_POSITION_POINTER_MASK, MTB_POSITION_POINTER(value)))
#define MTB_BWR_POSITION_POINTER(base, value)
(MTB_WR_POSITION_POINTER(base, value))
/*@}*/

/********************* MTB_MASTER - MTB Master Register ********************
****

/*!
* @brief MTB_MASTER - MTB Master Register (RW)
*
* Reset value: 0x00000080U
*
* The MTB_MASTER register contains the main program trace enable plus
other
* trace controls. This register can be modified by the explicit
programming model
* writes. MTB_MASTER[EN] and MTB_MASTER[HALTREQ] fields are also
automatically
* updated by the MTB hardware. Before the MTB_MASTER[EN] or
MTB_MASTER[TSTARTEN]
* bits are set to 1, software must initialize the MTB_POSITION and
MTB_FLOW
* registers. If the MTB_FLOW[WATERMARK] field is used to stop tracing or
to halt the
* processor, the MTB_MASTER[MASK] field must still be set to a value
that
* prevents the MTB_POSITION[POSITION] field from wrapping before it
reaches the
* MTB_FLOW[WATERMARK] value. The format of this mask field is different
than the
* MTBDWTF_MASKn[MASK].
*/
/*!
* @name Constants and macros for entire MTB_MASTER register
*/
/*@{ */

#define MTB_RD_MASTER(base) (MTB_MASTER_REG(base))
#define MTB_WR_MASTER(base, value) (MTB_MASTER_REG(base) = (value))
#define MTB_RMW_MASTER(base, mask, value) (MTB_WR_MASTER(base,
(MTB_RD_MASTER(base) & ~mask)) | (value)))
#define MTB_SET_MASTER(base, value) (MTB_WR_MASTER(base,
MTB_RD_MASTER(base) | (value)))

```

```

#define MTB_CLR_MASTER(base, value) (MTB_WR_MASTER(base,
MTB_RD_MASTER(base) & ~(value)))
#define MTB_TOG_MASTER(base, value) (MTB_WR_MASTER(base,
MTB_RD_MASTER(base) ^ (value)))
/*@}*/

/*
 * Constants & macros for individual MTB_MASTER bitfields
 */

/*!
 * @name Register MTB_MASTER, field MASK[4:0] (RW)
 *
 * This value determines the maximum size of the trace buffer in RAM. It
 * specifies the most-significant bit of the MTB_POSITION[POSITION] field
that can be
 * updated by automatic increment. If the trace tries to advance past
this power of
 * two, the MTB_POSITION[WRAP] bit is set to 1, the
MTB_POSITION[POSITION[MASK:0]]
 * bits are set to zero, and the MTB_POSITION[POSITION[11:MASK+1]] bits
remain
 * unchanged. This field causes the trace packet information to be stored
in a
 * circular buffer of size  $2^{[MASK+4]}$  bytes, that can be positioned in
memory at
 * multiples of this size. Valid values of this field are zero to 11.
Values greater
 * than the maximum have the same effect as the maximum.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_MASK field. */
#define MTB_RD_MASTER_MASK(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_MASK_MASK) >> MTB_MASTER_MASK_SHIFT)
#define MTB_BRD_MASTER_MASK(base) (MTB_RD_MASTER_MASK(base))

/*! @brief Set the MASK field to a new value. */
#define MTB_WR_MASTER_MASK(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_MASK_MASK, MTB_MASTER_MASK(value)))
#define MTB_BWR_MASTER_MASK(base, value) (MTB_WR_MASTER_MASK(base,
value))
/*@}*/

/*
 * @name Register MTB_MASTER, field TSTARTEN[5] (RW)
 *
 * If this bit is 1 and the TSTART signal is HIGH, then the EN bit is set
to 1.
 * Tracing continues until a stop condition occurs.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_TSTARTEN field. */
#define MTB_RD_MASTER_TSTARTEN(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_TSTARTEN_MASK) >> MTB_MASTER_TSTARTEN_SHIFT)

```

```

#define MTB_BRD_MASTER_TSTARTEN(base) (MTB_RD_MASTER_TSTARTEN(base))

/*! @brief Set the TSTARTEN field to a new value. */
#define MTB_WR_MASTER_TSTARTEN(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_TSTARTEN_MASK, MTB_MASTER_TSTARTEN(value)))
#define MTB_BWR_MASTER_TSTARTEN(base, value)
(MTB_WR_MASTER_TSTARTEN(base, value))
/*@}*/

/*
 * @name Register MTB_MASTER, field TSTOPEN[6] (RW)
 *
 * If this bit is 1 and the TSTOP signal is HIGH, then the EN bit is set
to 0.
 * If a trace packet is being written to memory, the write is completed
before
 * tracing is stopped.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_TSTOPEN field. */
#define MTB_RD_MASTER_TSTOPEN(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_TSTOPEN_MASK) >> MTB_MASTER_TSTOPEN_SHIFT)
#define MTB_BRD_MASTER_TSTOPEN(base) (MTB_RD_MASTER_TSTOPEN(base))

/*! @brief Set the TSTOPEN field to a new value. */
#define MTB_WR_MASTER_TSTOPEN(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_TSTOPEN_MASK, MTB_MASTER_TSTOPEN(value)))
#define MTB_BWR_MASTER_TSTOPEN(base, value) (MTB_WR_MASTER_TSTOPEN(base,
value))
/*@}*/

/*
 * @name Register MTB_MASTER, field SFRWPRI[7] (RW)
 *
 * If this bit is 0, then user or privileged AHB read and write accesses
to the
 * MTB_RAM Special Function Registers (programming model) are permitted.
If this
 * bit is 1, then only privileged write accesses are permitted; user
write
 * accesses are ignored. The HPROT[1] signal determines if an access is
user or
 * privileged. Note MTB_RAM SFR read access are not controlled by this
bit and are
 * always permitted.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_SFRWPRI field. */
#define MTB_RD_MASTER_SFRWPRI(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_SFRWPRI_MASK) >> MTB_MASTER_SFRWPRI_SHIFT)
#define MTB_BRD_MASTER_SFRWPRI(base) (MTB_RD_MASTER_SFRWPRI(base))

/*! @brief Set the SFRWPRI field to a new value. */

```

```

#define MTB_WR_MASTER_SFRWPRI(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_SFRWPRI_MASK, MTB_MASTER_SFRWPRI(value)))
#define MTB_BWR_MASTER_SFRWPRI(base, value)
(MTB_WR_MASTER_SFRWPRI(base, value))
/*@}*/

/*
 * @name Register MTB_MASTER, field RAMPRIV[8] (RW)
 *
 * If this bit is 0, then user or privileged AHB read and write accesses
to the
 * RAM are permitted. If this bit is 1, then only privileged AHB read and
write
 * accesses to the RAM are permitted and user accesses are RAZ/WI. The
HPROT[1]
 * signal determines if an access is a user or privileged mode reference.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_RAMPRIV field. */
#define MTB_RD_MASTER_RAMPRIV(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_RAMPRIV_MASK) >> MTB_MASTER_RAMPRIV_SHIFT)
#define MTB_BRD_MASTER_RAMPRIV(base) (MTB_RD_MASTER_RAMPRIV(base))

/*! @brief Set the RAMPRIV field to a new value. */
#define MTB_WR_MASTER_RAMPRIV(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_RAMPRIV_MASK, MTB_MASTER_RAMPRIV(value)))
#define MTB_BWR_MASTER_RAMPRIV(base, value) (MTB_WR_MASTER_RAMPRIV(base,
value))
/*@}*/

/*
 * @name Register MTB_MASTER, field HALTREQ[9] (RW)
 *
 * This bit is connected to the halt request signal of the trace logic,
EDBGRQ.
 * When HALTREQ is set to 1, the EDBFGRQ is asserted if DBGEN (invasive
debug
 * enable, one of the debug authentication interface signals) is also
HIGH. The
 * HALTREQ bit can be automatically set to 1 using the
MTB_FLOW[WATERMARK] field.
 */
/*@{*/
/*! @brief Read current value of the MTB_MASTER_HALTREQ field. */
#define MTB_RD_MASTER_HALTREQ(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_HALTREQ_MASK) >> MTB_MASTER_HALTREQ_SHIFT)
#define MTB_BRD_MASTER_HALTREQ(base) (MTB_RD_MASTER_HALTREQ(base))

/*! @brief Set the HALTREQ field to a new value. */
#define MTB_WR_MASTER_HALTREQ(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_HALTREQ_MASK, MTB_MASTER_HALTREQ(value)))
#define MTB_BWR_MASTER_HALTREQ(base, value) (MTB_WR_MASTER_HALTREQ(base,
value))
/*@}*/

```

```

/*!!
 * @name Register MTB_MASTER, field EN[31] (RW)
 *
 * When this bit is 1, trace data is written into the RAM memory location
 * addressed by MTB_POSITION[POINTER]. The MTB_POSITION[POINTER] value
auto increments
 * after the trace data packet is written. The EN bit can be
automatically set to
 * 0 using the MTB_FLOW[WATERMARK] field and the MTB_FLOW[AUTOSTOP] bit.
The EN
 * bit is automatically set to 1 if the TSTARTEN bit is 1 and the TSTART
signal
 * is HIGH. The EN bit is automatically set to 0 if TSTOPOPEN bit is 1 and
the TSTOP
 * signal is HIGH. If the EN bit is set to 0 because the
MTB_FLOW[WATERMARK]
 * field is set, then it is not automatically set to 1 if the TSTARTEN
bit is 1 and
 * the TSTART input is HIGH. In this case, tracing can only be restarted
if the
 * MTB_FLOW[WATERMARK] or MTB_POSITION[POINTER] value is changed by
software.
 */
/*@{*/
/*!! @brief Read current value of the MTB_MASTER_EN field. */
#define MTB_RD_MASTER_EN(base) ((MTB_MASTER_REG(base) &
MTB_MASTER_EN_MASK) >> MTB_MASTER_EN_SHIFT)
#define MTB_BRD_MASTER_EN(base) (MTB_RD_MASTER_EN(base))

/*!! @brief Set the EN field to a new value. */
#define MTB_WR_MASTER_EN(base, value) (MTB_RMW_MASTER(base,
MTB_MASTER_EN_MASK, MTB_MASTER_EN(value)))
#define MTB_BWR_MASTER_EN(base, value) (MTB_WR_MASTER_EN(base, value))
/*@}*/
*****  

* MTB_FLOW - MTB Flow Register  

*****  

/*!!
 * @brief MTB_FLOW - MTB Flow Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * The MTB_FLOW register contains the watermark address and the
 * autostop/autohalt control bits.
 */
/*!
 * @name Constants and macros for entire MTB_FLOW register
 */

```

```

/*@{*/
#define MTB_RD_FLOW(base)          (MTB_FLOW_REG(base))
#define MTB_WR_FLOW(base, value)   (MTB_FLOW_REG(base) = (value))
#define MTB_RMW_FLOW(base, mask, value) (MTB_WR_FLOW(base,
(MTB_RD_FLOW(base) & ~mask) | (value)))
#define MTB_SET_FLOW(base, value)  (MTB_WR_FLOW(base, MTB_RD_FLOW(base) | (value)))
#define MTB_CLR_FLOW(base, value)  (MTB_WR_FLOW(base, MTB_RD_FLOW(base) & ~value))
#define MTB_TOG_FLOW(base, value)  (MTB_WR_FLOW(base, MTB_RD_FLOW(base) ^ (value)))
/*@}*/

/*
 * Constants & macros for individual MTB_FLOW bitfields
 */

/*!
 * @name Register MTB_FLOW, field AUTOSTOP[0] (RW)
 *
 * If this bit is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then
 * the
 * * MTB_MASTER[EN] bit is automatically set to 0. This stops tracing.
 */
/*@*/
/*! @brief Read current value of the MTB_FLOW_AUTOSTOP field. */
#define MTB_RD_FLOW_AUTOSTOP(base) ((MTB_FLOW_REG(base) &
MTB_FLOW_AUTOSTOP_MASK) >> MTB_FLOW_AUTOSTOP_SHIFT)
#define MTB_BRD_FLOW_AUTOSTOP(base) (MTB_RD_FLOW_AUTOSTOP(base))

/*! @brief Set the AUTOSTOP field to a new value. */
#define MTB_WR_FLOW_AUTOSTOP(base, value) (MTB_RMW_FLOW(base,
MTB_FLOW_AUTOSTOP_MASK, MTB_FLOW_AUTOSTOP(value)))
#define MTB_BWR_FLOW_AUTOSTOP(base, value) (MTB_WR_FLOW_AUTOSTOP(base,
value))
/*@*/*/

/*!
 * @name Register MTB_FLOW, field AUTOHALT[1] (RW)
 *
 * If this bit is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then
 * the
 * * MTB_MASTER[HALTREQ] bit is automatically set to 1. If the DBGEN signal
 * is HIGH,
 * * the MTB asserts this halt request to the Cortex-M0+ processor by
 * asserting the
 * * EDBGREQ signal.
 */
/*@*/
/*! @brief Read current value of the MTB_FLOW_AUTOHALT field. */
#define MTB_RD_FLOW_AUTOHALT(base) ((MTB_FLOW_REG(base) &
MTB_FLOW_AUTOHALT_MASK) >> MTB_FLOW_AUTOHALT_SHIFT)
#define MTB_BRD_FLOW_AUTOHALT(base) (MTB_RD_FLOW_AUTOHALT(base))

```

```

/*! @brief Set the AUTOHALT field to a new value. */
#define MTB_WR_FLOW_AUTOHALT(base, value) (MTB_RMW_FLOW(base,
MTB_FLOW_AUTOHALT_MASK, MTB_FLOW_AUTOHALT(value)))
#define MTB_BWR_FLOW_AUTOHALT(base, value) (MTB_WR_FLOW_AUTOHALT(base,
value))
/*@}*/

/*!!
 * @name Register MTB_FLOW, field WATERMARK[31:3] (RW)
 *
 * This field contains an address in the same format as the
 * MTB_POSITION[POINTER] field. When the MTB_POSITION[POINTER] matches
the WATERMARK field value,
 * actions defined by the AUTOHALT and AUTOSTOP bits are performed.
 */
/*@{*/
/*! @brief Read current value of the MTB_FLOW_WATERMARK field. */
#define MTB_RD_FLOW_WATERMARK(base) ((MTB_FLOW_REG(base) &
MTB_FLOW_WATERMARK_MASK) >> MTB_FLOW_WATERMARK_SHIFT)
#define MTB_BRD_FLOW_WATERMARK(base) (MTB_RD_FLOW_WATERMARK(base))

/*! @brief Set the WATERMARK field to a new value. */
#define MTB_WR_FLOW_WATERMARK(base, value) (MTB_RMW_FLOW(base,
MTB_FLOW_WATERMARK_MASK, MTB_FLOW_WATERMARK(value)))
#define MTB_BWR_FLOW_WATERMARK(base, value) (MTB_WR_FLOW_WATERMARK(base,
value))
/*@}*/

*****  

* MTB_BASE - MTB Base Register  

*****  

  

/*!!
 * @brief MTB_BASE - MTB Base Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * The read-only MTB_BASE Register indicates where the RAM is located in
the
 * processor memory map. This register is provided to enable auto
discovery of the
 * MTB RAM location, by a debug agent and is defined by a hardware design
 * parameter. For these devices, the base address is defined by the
expression:
 * MTB_BASE[BASEADDR] = 0x2000_0000 - (RAM_Size/4)
 */
/*!
 * @name Constants and macros for entire MTB_BASE register
 */
/*@{*/
#define MTB_RD_BASE(base) (MTB_BASE_REG(base))

```

```

/*@}*/

/***** MTB_MODECTRL *****
 * MTB_MODECTRL - Integration Mode Control Register
***** */

/*!
 * @brief MTB_MODECTRL - Integration Mode Control Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * This register enables the device to switch from a functional mode, or
default
 * behavior, into integration mode. It is hardwired to specific values
used
 * during the auto-discovery process by an external debug agent.
 */
/*!
 * @name Constants and macros for entire MTB_MODECTRL register
 */
/*@{*/
#define MTB_RD_MODECTRL(base)      (MTB_MODECTRL_REG(base))
/*@}*/

/***** MTB_TAGSET *****
 * MTB_TAGSET - Claim TAG Set Register
***** */

/*!
 * @brief MTB_TAGSET - Claim TAG Set Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * The Claim Tag Set Register returns the number of bits that can be set
on a
 * read, and enables individual bits to be set on a write. It is
hardwired to
 * specific values used during the auto-discovery process by an external
debug agent.
 */
/*!
 * @name Constants and macros for entire MTB_TAGSET register
 */
/*@{*/
#define MTB_RD_TAGSET(base)       (MTB_TAGSET_REG(base))
/*@*/}

```

```

*****
* MTB_TAGCLEAR - Claim TAG Clear Register
*****
**** */

/*
 * @brief MTB_TAGCLEAR - Claim TAG Clear Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * The read/write Claim Tag Clear Register is used to read the claim
status on
 * debug resources. A read indicates the claim tag status. Writing 1 to a
specific
 * bit clears the corresponding claim tag to 0. It is hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
*/
/**!
 * @name Constants and macros for entire MTB_TAGCLEAR register
 */
/*@{ */
#define MTB_RD_TAGCLEAR(base)      (MTB_TAGCLEAR_REG(base))
/*}@*/



*****
* MTB_LOCKACCESS - Lock Access Register
*****
**** */

/*
 * @brief MTB_LOCKACCESS - Lock Access Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * The Lock Access Register enables a write access to component
registers. It is
 * hardwired to specific values used during the auto-discovery process by
an
 * external debug agent.
 */
*/
/**!
 * @name Constants and macros for entire MTB_LOCKACCESS register
 */
/*@{ */
#define MTB_RD_LOCKACCESS(base)    (MTB_LOCKACCESS_REG(base))
/*}@*/

```

```

/*********************  

*****  

 * MTB_LOCKSTAT - Lock Status Register  

*****  

****/  

  

/*!  

 * @brief MTB_LOCKSTAT - Lock Status Register (RO)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The Lock Status Register indicates the status of the lock control  

mechanism.  

 * This register is used in conjunction with the Lock Access Register. It  

is  

 * hardwired to specific values used during the auto-discovery process by  

an external  

 * debug agent.  

 */  

/*!  

 * @name Constants and macros for entire MTB_LOCKSTAT register  

 */  

/*@{ */  

#define MTB_RD_LOCKSTAT(base)      (MTB_LOCKSTAT_REG(base))  

/*@}/  

  

/*********************  

*****  

 * MTB_AUTHSTAT - Authentication Status Register  

*****  

****/  

  

/*!  

 * @brief MTB_AUTHSTAT - Authentication Status Register (RO)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The Authentication Status Register reports the required security level  

and  

 * current status of the security enable bit pairs. Where functionality  

changes on  

 * a given security level, this change must be reported in this register.  

It is  

 * connected to specific signals used during the auto-discovery process  

by an  

 * external debug agent. MTB_AUTHSTAT[3:2] indicates if nonsecure,  

noninvasive debug  

 * is enabled or disabled, while MTB_AUTHSTAT[1:0] indicates the  

enabled/disabled  

 * state of nonsecure, invasive debug. For both 2-bit fields, 0b10  

indicates the  

 * functionality is disabled and 0b11 indicates it is enabled.

```

```

/*
/*!
 * @name Constants and macros for entire MTB_AUTHSTAT register
 */
/*@{ */
#define MTB_RD_AUTHSTAT(base)      (MTB_AUTHSTAT_REG(base))
/*}@*/ */

/*
 * Constants & macros for individual MTB_AUTHSTAT bitfields
 */

/*!
 * @name Register MTB_AUTHSTAT, field BIT0[0] (RO)
 *
 * Connected to DBGEN.
 */
/*@{ */
/*! @brief Read current value of the MTB_AUTHSTAT_BIT0 field. */
#define MTB_RD_AUTHSTAT_BIT0(base) ((MTB_AUTHSTAT_REG(base) &
MTB_AUTHSTAT_BIT0_MASK) >> MTB_AUTHSTAT_BIT0_SHIFT)
#define MTB_BRD_AUTHSTAT_BIT0(base) (MTB_RD_AUTHSTAT_BIT0(base))
/*}@*/ */

/*!
 * @name Register MTB_AUTHSTAT, field BIT1[1] (ROO)
 *
 * Hardwired to 1.
 */
/*@{ */
/*! @brief Read current value of the MTB_AUTHSTAT_BIT1 field. */
#define MTB_RD_AUTHSTAT_BIT1(base) ((MTB_AUTHSTAT_REG(base) &
MTB_AUTHSTAT_BIT1_MASK) >> MTB_AUTHSTAT_BIT1_SHIFT)
#define MTB_BRD_AUTHSTAT_BIT1(base) (MTB_RD_AUTHSTAT_BIT1(base))
/*}@*/ */

/*!
 * @name Register MTB_AUTHSTAT, field BIT2[2] (RO)
 *
 * Connected to NIDEN or DBGEN signal.
 */
/*@{ */
/*! @brief Read current value of the MTB_AUTHSTAT_BIT2 field. */
#define MTB_RD_AUTHSTAT_BIT2(base) ((MTB_AUTHSTAT_REG(base) &
MTB_AUTHSTAT_BIT2_MASK) >> MTB_AUTHSTAT_BIT2_SHIFT)
#define MTB_BRD_AUTHSTAT_BIT2(base) (MTB_RD_AUTHSTAT_BIT2(base))
/*}@*/ */

/*!
 * @name Register MTB_AUTHSTAT, field BIT3[3] (ROO)
 *
 * Hardwired to 1.
 */
/*@{ */

```

```

/*! @brief Read current value of the MTB_AUTHSTAT_BIT3 field. */
#define MTB_RD_AUTHSTAT_BIT3(base) ((MTB_AUTHSTAT_REG(base) &
MTB_AUTHSTAT_BIT3_MASK) >> MTB_AUTHSTAT_BIT3_SHIFT)
#define MTB_BRD_AUTHSTAT_BIT3(base) (MTB_RD_AUTHSTAT_BIT3(base))
/*@}*/

/********************* ****
 * MTB_DEVICEARCH - Device Architecture Register
***** */

/*!!
 * @brief MTB_DEVICEARCH - Device Architecture Register (RO)
 *
 * Reset value: 0x47700A31U
 *
 * This register indicates the device architecture. It is hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!!
 * @name Constants and macros for entire MTB_DEVICEARCH register
 */
/*@{*/
#define MTB_RD_DEVICEARCH(base) (MTB_DEVICEARCH_REG(base))
/*@}*/

/********************* ****
 * MTB_DEVICECFG - Device Configuration Register
***** */

/*!!
 * @brief MTB_DEVICECFG - Device Configuration Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * This register indicates the device configuration. It is hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!!
 * @name Constants and macros for entire MTB_DEVICECFG register
 */
/*@{*/
#define MTB_RD_DEVICECFG(base) (MTB_DEVICECFG_REG(base))
/*@*/}

```

```

*****
* MTB_DEVICETYPID - Device Type Identifier Register
*****
*****/



/*!
* @brief MTB_DEVICETYPID - Device Type Identifier Register (RO)
*
* Reset value: 0x00000031U
*
* This register indicates the device type ID. It is hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!
* @name Constants and macros for entire MTB_DEVICETYPID register
*/
/*@{*/
#define MTB_RD_DEVICETYPID(base) (MTB_DEVICETYPID_REG(base))
/*}@*/



*****
* MTB_PERIPHID - Peripheral ID Register
*****
*****/



/*!
* @brief MTB_PERIPHID - Peripheral ID Register (RO)
*
* Reset value: 0x00000000U
*
* These registers indicate the peripheral IDs. They are hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!
* @name Constants and macros for entire MTB_PERIPHID register
*/
/*@{*/
#define MTB_RD_PERIPHID(base, index) (MTB_PERIPHID_REG(base, index))
/*}@*/



*****
* MTB_COMPID - Component ID Register
*****
*****/

```

```

/*!
 * @brief MTB_COMPID - Component ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the component IDs. They are hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
 * agent.
 */
/*!
 * @name Constants and macros for entire MTB_COMPID register
 */
/*@{ */
#define MTB_RD_COMPID(base, index) (MTB_COMPID_REG(base, index))
/*}@*/
/*
 * MKL25Z4 MTBDWT
 *
 * MTB data watchpoint and trace
 *
 * Registers defined in this header file:
 * - MTBDWT_CTRL - MTB DWT Control Register
 * - MTBDWT_COMP - MTB_DWT Comparator Register
 * - MTBDWT_MASK - MTB_DWT Comparator Mask Register
 * - MTBDWT_FCT - MTB_DWT Comparator Function Register 0
 * - MTBDWT_TBCTRL - MTB_DWT Trace Buffer Control Register
 * - MTBDWT_DEVICECFG - Device Configuration Register
 * - MTBDWT_DEVICETYPID - Device Type Identifier Register
 * - MTBDWT_PERIPHID - Peripheral ID Register
 * - MTBDWT_COMPID - Component ID Register
 */
#define MTBDWT_INSTANCE_COUNT (1U) /*!< Number of instances of the MTBDWT
module. */
#define MTBDWT_IDX (0U) /*!< Instance number for MTBDWT. */

*****  

* MTBDWT_CTRL - MTB DWT Control Register  

*****  

*/!  

 * @brief MTBDWT_CTRL - MTB DWT Control Register (RO)
 *
 * Reset value: 0x2F000000U
 *
 * The MTBDWT_CTRL register provides read-only information on the
 * watchpoint
 * configuration for the MTB_DWT.

```

```

/*
 */
/*!
 * @name Constants and macros for entire MTBDWT_CTRL register
 */
/*@{ */
#define MTBDWT_RD_CTRL(base)      (MTBDWT_CTRL_REG(base))
/*}@*/ */

/*
 * Constants & macros for individual MTBDWT_CTRL bitfields
 */

/*!
 * @name Register MTBDWT_CTRL, field DWTCFGCTRL[27:0] (RO)
 *
 * This field is hardwired to 0xF00_0000, disabling all the remaining DWT
 * functionality. The specific fields and their state are:
MTBDWT_CTRL[27] = NOTRCPKT =
    * 1, trace sample and exception trace is not supported MTBDWT_CTRL[26] =
    * NOEXTTRIG = 1, external match signals are not supported
MTBDWT_CTRL[25] = NOCYCCNT =
    * 1, cycle counter is not supported MTBDWT_CTRL[24] = NOPRFCNT = 1,
profiling
    * counters are not supported MTBDWT_CTRL[22] = CYCEBTENA = 0, no POSTCNT
    * underflow packets generated MTBDWT_CTRL[21] = FOLDEVTENA = 0, no
folded instruction
    * counter overflow events MTBDWT_CTRL[20] = LSUEVTENA = 0, no LSU
counter overflow
    * events MTBDWT_CTRL[19] = SLEEPEVTENA = 0, no sleep counter overflow
events
    * MTBDWT_CTRL[18] = EXCEVTENA = 0, no exception overhead counter events
    * MTBDWT_CTRL[17] = CPIEVTEA = 0, no CPI counter overflow events
MTBDWT_CTRL[16] =
    * EXCTRCENA = 0, generation of exception trace disabled MTBDWT_CTRL[12]
= PCSAMPLENA =
    * 0, no periodic PC sample packets generated MTBDWT_CTRL[11:10] =
SYNCTAP = 0,
    * no synchronization packets MTBDWT_CTRL[9] = CYCTAP = 0, cycle counter
is not
    * supported MTBDWT_CTRL[8:5] = POSTINIT = 0, cycle counter is not
supported
    * MTBDWT_CTRL[4:1] = POSTPRESET = 0, cycle counter is not supported
MTBDWT_CTRL[0] =
    * CYCCNTENA = 0, cycle counter is not supported
*/
/*@{ */
/*! @brief Read current value of the MTBDWT_CTRL_DWTCFGCTRL field. */
#define MTBDWT_RD_CTRL_DWTCFGCTRL(base) ((MTBDWT_CTRL_REG(base) &
MTBDWT_CTRL_DWTCFGCTRL_MASK) >> MTBDWT_CTRL_DWTCFGCTRL_SHIFT)
#define MTBDWT_BRD_CTRL_DWTCFGCTRL(base)
(MTBDWT_RD_CTRL_DWTCFGCTRL(base))
/*}@*/ */

/*

```

```

* @name Register MTBDWT_CTRL, field NUMCMP[31:28] (RO)
*
* The MTB_DWT implements two comparators.
*/
/*@{ */
/*! @brief Read current value of the MTBDWT_CTRL_NUMCMP field. */
#define MTBDWT_RD_CTRL_NUMCMP(base) ((MTBDWT_CTRL_REG(base) &
MTBDWT_CTRL_NUMCMP_MASK) >> MTBDWT_CTRL_NUMCMP_SHIFT)
#define MTBDWT_BRD_CTRL_NUMCMP(base) (MTBDWT_RD_CTRL_NUMCMP(base))
/*}@*/



/********************* **** */
***** */
* MTBDWT_COMP - MTB_DWT Comparator Register
***** */
/*!
* @brief MTBDWT_COMP - MTB_DWT Comparator Register (RW)
*
* Reset value: 0x00000000U
*
* The MTBDWT_COMPn registers provide the reference value for comparator
n.
*/
/*!
* @name Constants and macros for entire MTBDWT_COMP register
*/
/*@{ */
#define MTBDWT_RD_COMP(base, index) (MTBDWT_COMP_REG(base, index))
#define MTBDWT_WR_COMP(base, index, value) (MTBDWT_COMP_REG(base, index)
= (value))
#define MTBDWT_RMW_COMP(base, index, mask, value) (MTBDWT_WR_COMP(base,
index, (MTBDWT_RD_COMP(base, index) & ~(mask)) | (value)))
#define MTBDWT_SET_COMP(base, index, value) (MTBDWT_WR_COMP(base, index,
MTBDWT_RD_COMP(base, index) | (value)))
#define MTBDWT_CLR_COMP(base, index, value) (MTBDWT_WR_COMP(base, index,
MTBDWT_RD_COMP(base, index) & ~(value)))
#define MTBDWT_TOG_COMP(base, index, value) (MTBDWT_WR_COMP(base, index,
MTBDWT_RD_COMP(base, index) ^ (value)))
/*}@*/



/********************* **** */
***** */
* MTBDWT_MASK - MTB_DWT Comparator Mask Register
***** */
/*!
* @brief MTBDWT_MASK - MTB_DWT Comparator Mask Register (RW)
*
* Reset value: 0x00000000U

```

```

/*
 * The MTBDWT_MASKn registers define the size of the ignore mask applied
to the
 * reference address for address range matching by comparator n. Note the
format
 * of this mask field is different than the MTB_MASTER[MASK].
 */
/*!
 * @name Constants and macros for entire MTBDWT_MASK register
 */
/*@{*/
#define MTBDWT_RD_MASK(base, index) (MTBDWT_MASK_REG(base, index))
#define MTBDWT_WR_MASK(base, index, value) (MTBDWT_MASK_REG(base, index)
= (value))
#define MTBDWT_RMW_MASK(base, index, mask, value) (MTBDWT_WR_MASK(base,
index, (MTBDWT_RD_MASK(base, index) & ~(mask)) | (value)))
#define MTBDWT_SET_MASK(base, index, value) (MTBDWT_WR_MASK(base, index,
MTBDWT_RD_MASK(base, index) | (value)))
#define MTBDWT_CLR_MASK(base, index, value) (MTBDWT_WR_MASK(base, index,
MTBDWT_RD_MASK(base, index) & ~(value)))
#define MTBDWT_TOG_MASK(base, index, value) (MTBDWT_WR_MASK(base, index,
MTBDWT_RD_MASK(base, index) ^ (value)))
/*}@*/
/*
 * Constants & macros for individual MTBDWT_MASK bitfields
 */
/*!
 * @name Register MTBDWT_MASK, field MASK[4:0] (RW)
 *
 * The value of the ignore mask, 0-31 bits, is applied to address range
 * matching. MASK = 0 is used to include all bits of the address in the
comparison,
 * except if MASK = 0 and the comparator is configured to watch
instruction fetch
 * addresses, address bit [0] is ignored by the hardware since all
fetches must be at
 * least halfword aligned. For MASK != 0 and regardless of watch type,
address
 * bits [x-1:0] are ignored in the address comparison. Using a mask means
the
 * comparator matches on a range of addresses, defined by the unmasked
most
 * significant bits of the address, bits [31:x]. The maximum MASK value
is 24, producing a
 * 16 Mbyte mask. An attempted write of a MASK value > 24 is limited by
the
 * MTBDWT hardware to 24. If MTBDWT_COMP0 is used as a data value
comparator, then
 * MTBDWT_MASK0 should be programmed to zero.
 */
/*@{*/
/*! @brief Read current value of the MTBDWT_MASK_MASK field. */

```

```

#define MTBDWT_RD_MASK_MASK(base, index) ((MTBDWT_MASK_REG(base, index) &
MTBDWT_MASK_MASK_MASK) >> MTBDWT_MASK_MASK_SHIFT)
#define MTBDWT_BRD_MASK_MASK(base, index) (MTBDWT_RD_MASK_MASK(base,
index))

/*! @brief Set the MASK field to a new value. */
#define MTBDWT_WR_MASK_MASK(base, index, value) (MTBDWT_RMW_MASK(base,
index, MTBDWT_MASK_MASK_MASK, MTBDWT_MASK_MASK(value)))
#define MTBDWT_BWR_MASK_MASK(base, index, value)
(MTBDWT_WR_MASK_MASK(base, index, value))
/*@}*/

/********************* MTBDWT_FCT - MTB_DWT Comparator Function Register 0 *****/
***** */

* MTBDWT_FCT - MTB_DWT Comparator Function Register 0

***** */

/*! !
* @brief MTBDWT_FCT - MTB_DWT Comparator Function Register 0 (RW)
*
* Reset value: 0x00000000U
*
* The MTBDWT_FCTn registers control the operation of comparator n.
*/
/*! !
* @name Constants and macros for entire MTBDWT_FCT register
*/
/*@{ */

#define MTBDWT_RD_FCT(base, index) (MTBDWT_FCT_REG(base, index))
#define MTBDWT_WR_FCT(base, index, value) (MTBDWT_FCT_REG(base, index) =
(value))
#define MTBDWT_RMW_FCT(base, index, mask, value) (MTBDWT_WR_FCT(base,
index, (MTBDWT_RD_FCT(base, index) & ~mask) | value)))
#define MTBDWT_SET_FCT(base, index, value) (MTBDWT_WR_FCT(base, index,
MTBDWT_RD_FCT(base, index) | value))
#define MTBDWT_CLR_FCT(base, index, value) (MTBDWT_WR_FCT(base, index,
MTBDWT_RD_FCT(base, index) & ~value))
#define MTBDWT_TOG_FCT(base, index, value) (MTBDWT_WR_FCT(base, index,
MTBDWT_RD_FCT(base, index) ^ value))
/*@}*/

/*
* Constants & macros for individual MTBDWT_FCT bitfields
*/

/*! !
* @name Register MTBDWT_FCT, field FUNCTION[3:0] (RW)
*
* Selects the action taken on a comparator match. If MTBDWT_COMP0 is
used for a
* data value and MTBDWT_COMP1 for an address value, then
MTBDWT_FCT1[FUNCTION]

```

```

    * must be set to zero. For this configuration, MTBDWT_MASK1 can be set
    to a
    * non-zero value, so the combined comparators match on a range of
addresses.
    *
    * Values:
    * - 0b0000 - Disabled.
    * - 0b0100 - Instruction fetch.
    * - 0b0101 - Data operand read.
    * - 0b0110 - Data operand write.
    * - 0b0111 - Data operand (read + write).
    */
/*@{*/
/*! @brief Read current value of the MTBDWT_FCT_FUNCTION field. */
#define MTBDWT_RD_FCT_FUNCTION(base, index) ((MTBDWT_FCT_REG(base, index)
& MTBDWT_FCT_FUNCTION_MASK) >> MTBDWT_FCT_FUNCTION_SHIFT)
#define MTBDWT_BRD_FCT_FUNCTION(base, index)
(MTBDWT_RD_FCT_FUNCTION(base, index))

/*! @brief Set the FUNCTION field to a new value. */
#define MTBDWT_WR_FCT_FUNCTION(base, index, value) (MTBDWT_RMW_FCT(base,
index, MTBDWT_FCT_FUNCTION_MASK, MTBDWT_FCT_FUNCTION(value)))
#define MTBDWT_BWR_FCT_FUNCTION(base, index, value)
(MTBDWT_WR_FCT_FUNCTION(base, index, value))
/*@}*/

/*
 * @name Register MTBDWT_FCT, field DATAVMATCH[8] (RW)
 *
 * The assertion of this bit enables data value comparison. For this
 * implementation, MTBDWT_COMP0 supports address or data value
comparisons; MTBDWT_COMP1
 * only supports address comparisons.
*
* Values:
* - 0b0 - Perform address comparison.
* - 0b1 - Perform data value comparison.
*/
/*@{*/
/*! @brief Read current value of the MTBDWT_FCT_DATAVMATCH field. */
#define MTBDWT_RD_FCT_DATAVMATCH(base, index) ((MTBDWT_FCT_REG(base,
index) & MTBDWT_FCT_DATAVMATCH_MASK) >> MTBDWT_FCT_DATAVMATCH_SHIFT)
#define MTBDWT_BRD_FCT_DATAVMATCH(base, index)
(MTBDWT_RD_FCT_DATAVMATCH(base, index))

/*! @brief Set the DATAVMATCH field to a new value. */
#define MTBDWT_WR_FCT_DATAVMATCH(base, index, value)
(MTBDWT_RMW_FCT(base, index, MTBDWT_FCT_DATAVMATCH_MASK,
MTBDWT_FCT_DATAVMATCH(value)))
#define MTBDWT_BWR_FCT_DATAVMATCH(base, index, value)
(MTBDWT_WR_FCT_DATAVMATCH(base, index, value))
/*@}*/

/*

```

```

* @name Register MTBDWT_FCT, field DATAVSIZE[11:10] (RW)
*
* For data value matching, this field defines the size of the required
data
* comparison.
*
* Values:
* - 0b00 - Byte.
* - 0b01 - Halfword.
* - 0b10 - Word.
* - 0b11 - Reserved. Any attempts to use this value results in
UNPREDICTABLE
*      behavior.
*/
/*@{ */
/*! @brief Read current value of the MTBDWT_FCT_DATAVSIZE field. */
#define MTBDWT_RD_FCT_DATAVSIZE(base, index) ((MTBDWT_FCT_REG(base,
index) & MTBDWT_FCT_DATAVSIZE_MASK) >> MTBDWT_FCT_DATAVSIZE_SHIFT)
#define MTBDWT_BRD_FCT_DATAVSIZE(base, index)
(MTBDWT_RD_FCT_DATAVSIZE(base, index))

/*! @brief Set the DATAVSIZE field to a new value. */
#define MTBDWT_WR_FCT_DATAVSIZE(base, index, value) (MTBDWT_RMW_FCT(base,
index, MTBDWT_FCT_DATAVSIZE_MASK, MTBDWT_FCT_DATAVSIZE(value)))
#define MTBDWT_BWR_FCT_DATAVSIZE(base, index, value)
(MTBDWT_WR_FCT_DATAVSIZE(base, index, value))
/*}@ */ */

/*!
* @name Register MTBDWT_FCT, field DATAVADDR0[15:12] (RW)
*
* Since the MTB_DWT implements two comparators, the DATAVADDR0 field is
* restricted to values {0,1}. When the DATAVMATCH bit is asserted, this
field defines
* the comparator number to use for linked address comparison. If
MTBDWT_COMP0 is
* used as a data watchpoint and MTBDWT_COMP1 as an address watchpoint,
* DATAVADDR0 must be set.
*/
/*@{ */
/*! @brief Read current value of the MTBDWT_FCT_DATAVADDR0 field. */
#define MTBDWT_RD_FCT_DATAVADDR0(base, index) ((MTBDWT_FCT_REG(base,
index) & MTBDWT_FCT_DATAVADDR0_MASK) >> MTBDWT_FCT_DATAVADDR0_SHIFT)
#define MTBDWT_BRD_FCT_DATAVADDR0(base, index)
(MTBDWT_RD_FCT_DATAVADDR0(base, index))

/*! @brief Set the DATAVADDR0 field to a new value. */
#define MTBDWT_WR_FCT_DATAVADDR0(base, index, value)
(MTBDWT_RMW_FCT(base, index, MTBDWT_FCT_DATAVADDR0_MASK,
MTBDWT_FCT_DATAVADDR0(value)))
#define MTBDWT_BWR_FCT_DATAVADDR0(base, index, value)
(MTBDWT_WR_FCT_DATAVADDR0(base, index, value))
/*}@ */

```

```

/*!
 * @name Register MTBDWT_FCT, field MATCHED[24] (RO)
 *
 * If this read-only flag is asserted, it indicates the operation defined
by the
 * FUNCTION field occurred since the last read of the register. Reading
the
 * register clears this bit.
 *
 * Values:
 * - 0b0 - No match.
 * - 0b1 - Match occurred.
 */
/*@{*/
/*! @brief Read current value of the MTBDWT_FCT_MATCHED field. */
#define MTBDWT_RD_FCT_MATCHED(base, index) ((MTBDWT_FCT_REG(base, index)
& MTBDWT_FCT_MATCHED_MASK) >> MTBDWT_FCT_MATCHED_SHIFT)
#define MTBDWT_BRD_FCT_MATCHED(base, index) (MTBDWT_RD_FCT_MATCHED(base,
index))
/*}@*/



/***** MTBDWT_TBCTRL - MTB_DWT Trace Buffer Control Register *****/
*****/


/*!
 * @brief MTBDWT_TBCTRL - MTB_DWT Trace Buffer Control Register (RW)
 *
 * Reset value: 0x20000000U
 *
 * The MTBDWT_TBCTRL register defines how the watchpoint comparisons
control the
 * actual trace buffer operation. Recall the MTB supports starting and
stopping
 * the program trace based on the watchpoint comparisons signaled via
TSTART and
 * TSTOP. The watchpoint comparison signals are enabled in the MTB's
control
 * logic by setting the appropriate enable bits, MTB_MASTER[TSTARTEN,
TSTOPEN]. In
 * the event of simultaneous assertion of both TSTART and TSTOP, TSTART
takes
 * priority.
 */
/*!
 * @name Constants and macros for entire MTBDWT_TBCTRL register
 */
/*@{*/
#define MTBDWT_RD_TBCTRL(base) (MTBDWT_TBCTRL_REG(base))
#define MTBDWT_WR_TBCTRL(base, value) (MTBDWT_TBCTRL_REG(base) = (value))

```

```

#define MTBDWT_RMW_TBCTRL(base, mask, value) (MTBDWT_WR_TBCTRL(base, (MTBDWT_RD_TBCTRL(base) & ~(mask)) | (value)))
#define MTBDWT_SET_TBCTRL(base, value) (MTBDWT_WR_TBCTRL(base, MTBDWT_RD_TBCTRL(base) | (value)))
#define MTBDWT_CLR_TBCTRL(base, value) (MTBDWT_WR_TBCTRL(base, MTBDWT_RD_TBCTRL(base) & ~(value)))
#define MTBDWT_TOG_TBCTRL(base, value) (MTBDWT_WR_TBCTRL(base, MTBDWT_RD_TBCTRL(base) ^ (value)))
/*@}*/

/*
 * Constants & macros for individual MTBDWT_TBCTRL bitfields
 */

/*!!
 * @name Register MTBDWT_TBCTRL, field ACOMP0[0] (RW)
 *
 * When the MTBDWT_FCT0[MATCHED] is set, it indicates MTBDWT_COMP0 address
 * compare has triggered and the trace buffer's recording state is changed. The
 * assertion of MTBDWT_FCT0[MATCHED] is caused by the following conditions: Address
 * match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH] = 0 Data match in
 * MTBDWT_COMP0
 * when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,0} Data match in
 * MTBDWT_COMP0
 * and address match in MTBDWT_COMP1 when MTBDWT_FCT0[DATAVMATCH,
 * DATAVADDR0] =
 * {1,1}
 *
 * Values:
 * - 0b0 - Trigger TSTOP based on the assertion of MTBDWT_FCT0[MATCHED].
 * - 0b1 - Trigger TSTART based on the assertion of MTBDWT_FCT0[MATCHED].
 */
/*@{*/
/*!! @brief Read current value of the MTBDWT_TBCTRL_ACOMP0 field. */
#define MTBDWT_RD_TBCTRL_ACOMP0(base) ((MTBDWT_TBCTRL_REG(base) & MTBDWT_TBCTRL_ACOMP0_MASK) >> MTBDWT_TBCTRL_ACOMP0_SHIFT)
#define MTBDWT_BRD_TBCTRL_ACOMP0(base) (MTBDWT_RD_TBCTRL_ACOMP0(base))

/*!! @brief Set the ACOMP0 field to a new value. */
#define MTBDWT_WR_TBCTRL_ACOMP0(base, value) (MTBDWT_RMW_TBCTRL(base, MTBDWT_TBCTRL_ACOMP0_MASK, MTBDWT_TBCTRL_ACOMP0(value)))
#define MTBDWT_BWR_TBCTRL_ACOMP0(base, value)
(MTBDWT_WR_TBCTRL_ACOMP0(base, value))
/*@}*/

/*!!
 * @name Register MTBDWT_TBCTRL, field ACOMP1[1] (RW)
 *
 * When the MTBDWT_FCT1[MATCHED] is set, it indicates MTBDWT_COMP1 address
*/

```

```

    * compare has triggered and the trace buffer's recording state is
    changed.
    *
    * Values:
    * - 0b0 - Trigger TSTOP based on the assertion of MTBDWT_FCT1[MATCHED].
    * - 0b1 - Trigger TSTART based on the assertion of MTBDWT_FCT1[MATCHED].
    */
/*@{*/
/*! @brief Read current value of the MTBDWT_TBCTRL_ACOMP1 field. */
#define MTBDWT_RD_TBCTRL_ACOMP1(base) ((MTBDWT_TBCTRL_REG(base) &
MTBDWT_TBCTRL_ACOMP1_MASK) >> MTBDWT_TBCTRL_ACOMP1_SHIFT)
#define MTBDWT_BRD_TBCTRL_ACOMP1(base) (MTBDWT_RD_TBCTRL_ACOMP1(base))

/*! @brief Set the ACOMP1 field to a new value. */
#define MTBDWT_WR_TBCTRL_ACOMP1(base, value) (MTBDWT_RMW_TBCTRL(base,
MTBDWT_TBCTRL_ACOMP1_MASK, MTBDWT_TBCTRL_ACOMP1(value)))
#define MTBDWT_BWR_TBCTRL_ACOMP1(base, value)
(MTBDWT_WR_TBCTRL_ACOMP1(base, value))
/*@}*/

/*!
* @name Register MTBDWT_TBCTRL, field NUMCOMP[31:28] (RO)
*
* This read-only field specifies the number of comparators in the
MTB_DWT. This
* implementation includes two registers.
*/
/*@{*/
/*! @brief Read current value of the MTBDWT_TBCTRL_NUMCOMP field. */
#define MTBDWT_RD_TBCTRL_NUMCOMP(base) ((MTBDWT_TBCTRL_REG(base) &
MTBDWT_TBCTRL_NUMCOMP_MASK) >> MTBDWT_TBCTRL_NUMCOMP_SHIFT)
#define MTBDWT_BRD_TBCTRL_NUMCOMP(base) (MTBDWT_RD_TBCTRL_NUMCOMP(base))
/*@}*/

*****  

*****  

    * MTBDWT_DEVICECFG - Device Configuration Register  

*****  

*****  

/*!  

* @brief MTBDWT_DEVICECFG - Device Configuration Register (RO)
*
* Reset value: 0x00000000U
*
* This register indicates the device configuration. It is hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!  

* @name Constants and macros for entire MTBDWT_DEVICECFG register
*/

```

```

/*@{*/
#define MTBDWT_RD_DEVICECFG(base)  (MTBDWT_DEVICECFG_REG(base))
/*}@*/



/***** *****
 * MTBDWT_DEVICETYPID - Device Type Identifier Register
***** */

/*!
 * @brief MTBDWT_DEVICETYPID - Device Type Identifier Register (RO)
 *
 * Reset value: 0x00000004U
 *
 * This register indicates the device type ID. It is hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire MTBDWT_DEVICETYPID register
 */
/*@{*/
#define MTBDWT_RD_DEVICETYPID(base)  (MTBDWT_DEVICETYPID_REG(base))
/*}@*/



/***** *****
 * MTBDWT_PERIPHID - Peripheral ID Register
***** */

/*!
 * @brief MTBDWT_PERIPHID - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire MTBDWT_PERIPHID register
 */
/*@{*/
#define MTBDWT_RD_PERIPHID(base, index)  (MTBDWT_PERIPHID_REG(base,
index))
/*}@*/

```

```

*****
 * MTBDWT_COMPID - Component ID Register
*****
**** */

/*!
 * @brief MTBDWT_COMPID - Component ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the component IDs. They are hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire MTBDWT_COMPID register
 */
/*@{ */
#define MTBDWT_RD_COMPID(base, index) (MTBDWT_COMPID_REG(base, index))
/*@} */

/*
 * MKL25Z4 NV
 *
 * Flash configuration field
 *
 * Registers defined in this header file:
 * - NV_BACKKEY3 - Backdoor Comparison Key 3.
 * - NV_BACKKEY2 - Backdoor Comparison Key 2.
 * - NV_BACKKEY1 - Backdoor Comparison Key 1.
 * - NV_BACKKEY0 - Backdoor Comparison Key 0.
 * - NV_BACKKEY7 - Backdoor Comparison Key 7.
 * - NV_BACKKEY6 - Backdoor Comparison Key 6.
 * - NV_BACKKEY5 - Backdoor Comparison Key 5.
 * - NV_BACKKEY4 - Backdoor Comparison Key 4.
 * - NV_FPROT3 - Non-volatile P-Flash Protection 1 - Low Register
 * - NV_FPROT2 - Non-volatile P-Flash Protection 1 - High Register
 * - NV_FPROT1 - Non-volatile P-Flash Protection 0 - Low Register
 * - NV_FPROT0 - Non-volatile P-Flash Protection 0 - High Register
 * - NV_FSEC - Non-volatile Flash Security Register
 * - NV_FOPT - Non-volatile Flash Option Register
*/
#define NV_INSTANCE_COUNT (1U) /*!< Number of instances of the NV module.
*/
#define FTFA_FlashConfig_IDX (0U) /*!< Instance number for
FTFA_FlashConfig. */

*****
 * NV_BACKKEY3 - Backdoor Comparison Key 3.

```

```
*****
**** */

/*!
 * @brief NV_BACKKEY3 - Backdoor Comparison Key 3. (RO)
 *
 * Reset value: 0xFFU
 */
/*!!
 * @name Constants and macros for entire NV_BACKKEY3 register
 */
/*@{ */
#define NV_RD_BACKKEY3(base)      (NV_BACKKEY3_REG(base))
/*}@ */

/*****
 * NV_BACKKEY2 - Backdoor Comparison Key 2.
 *****/

***** */

/*!
 * @brief NV_BACKKEY2 - Backdoor Comparison Key 2. (RO)
 *
 * Reset value: 0xFFU
 */
/*!!
 * @name Constants and macros for entire NV_BACKKEY2 register
 */
/*@{ */
#define NV_RD_BACKKEY2(base)      (NV_BACKKEY2_REG(base))
/*}@ */

/*****
 * NV_BACKKEY1 - Backdoor Comparison Key 1.
 *****/

***** */

/*!
 * @brief NV_BACKKEY1 - Backdoor Comparison Key 1. (RO)
 *
 * Reset value: 0xFFU
 */
/*!!
 * @name Constants and macros for entire NV_BACKKEY1 register
 */
/*@{ */
#define NV_RD_BACKKEY1(base)      (NV_BACKKEY1_REG(base))
/*}@ */
```

```

*****
* NV_BACKKEY0 - Backdoor Comparison Key 0.

*****
**** */

/*!
* @brief NV_BACKKEY0 - Backdoor Comparison Key 0. (RO)
*
* Reset value: 0xFFU
*/
/*!
* @name Constants and macros for entire NV_BACKKEY0 register
*/
/*@{ */
#define NV_RD_BACKKEY0(base)      (NV_BACKKEY0_REG(base))
/*}@*/



*****
* NV_BACKKEY7 - Backdoor Comparison Key 7.

*****
**** */

/*!
* @brief NV_BACKKEY7 - Backdoor Comparison Key 7. (RO)
*
* Reset value: 0xFFU
*/
/*!
* @name Constants and macros for entire NV_BACKKEY7 register
*/
/*@{ */
#define NV_RD_BACKKEY7(base)      (NV_BACKKEY7_REG(base))
/*}@*/



*****
* NV_BACKKEY6 - Backdoor Comparison Key 6.

*****
**** */

/*!
* @brief NV_BACKKEY6 - Backdoor Comparison Key 6. (RO)
*
* Reset value: 0xFFU
*/
/*!
* @name Constants and macros for entire NV_BACKKEY6 register
*/
/*@{ */

```

```

#define NV_RD_BACKKEY6(base)      (NV_BACKKEY6_REG(base))
/*@}*/

*****  

* NV_BACKKEY5 - Backdoor Comparison Key 5.  

*****  

/*!  

 * @brief NV_BACKKEY5 - Backdoor Comparison Key 5. (RO)  

 *  

 * Reset value: 0xFFU  

 */  

/*!  

 * @name Constants and macros for entire NV_BACKKEY5 register  

 */  

/*@{*/  

#define NV_RD_BACKKEY5(base)      (NV_BACKKEY5_REG(base))
/*@}*/

*****  

* NV_BACKKEY4 - Backdoor Comparison Key 4.  

*****  

/*!  

 * @brief NV_BACKKEY4 - Backdoor Comparison Key 4. (RO)  

 *  

 * Reset value: 0xFFU  

 */  

/*!  

 * @name Constants and macros for entire NV_BACKKEY4 register  

 */  

/*@{*/  

#define NV_RD_BACKKEY4(base)      (NV_BACKKEY4_REG(base))
/*@*/

*****  

* NV_FPROT3 - Non-volatile P-Flash Protection 1 - Low Register  

*****  

/*!  

 * @brief NV_FPROT3 - Non-volatile P-Flash Protection 1 - Low Register  

 (RO)  

 *  

 * Reset value: 0xFFU  

 */

```

```

/*!
 * @name Constants and macros for entire NV_FPROT3 register
 */
/*@{*/
#define NV_RD_FPROT3(base)          (NV_FPROT3_REG(base))
/*}@*/



/********************* NV_FPROT2 - Non-volatile P-Flash Protection 1 - High Register *****

* NV_FPROT2 - Non-volatile P-Flash Protection 1 - High Register
(RO)
*
* Reset value: 0xFFU
*/
/*!

* @name Constants and macros for entire NV_FPROT2 register
*/
/*@{*/
#define NV_RD_FPROT2(base)          (NV_FPROT2_REG(base))
/*}@*/



/********************* NV_FPROT1 - Non-volatile P-Flash Protection 0 - Low Register *****

* NV_FPROT1 - Non-volatile P-Flash Protection 0 - Low Register
(RO)
*
* Reset value: 0xFFU
*/
/*!

* @name Constants and macros for entire NV_FPROT1 register
*/
/*@{*/
#define NV_RD_FPROT1(base)          (NV_FPROT1_REG(base))
/*}@*/



/********************* NV_FPROT0 - Non-volatile P-Flash Protection 0 - High Register *****

* NV_FPROT0 - Non-volatile P-Flash Protection 0 - High Register

```

```

/*!
 * @brief NV_FPROT0 - Non-volatile P-Flash Protection 0 - High Register
(RO)
 *
 * Reset value: 0xFFU
 */
/*!
 * @name Constants and macros for entire NV_FPROT0 register
 */
/*@{*/
#define NV_RD_FPROT0(base)          (NV_FPROT0_REG(base))
/*}@*/

```

---

```

*****  

* NV_FSEC - Non-volatile Flash Security Register  

*****  

*****  


```

```

/*!
 * @brief NV_FSEC - Non-volatile Flash Security Register (RO)
 *
 * Reset value: 0xFFU
 *
 * Allows the user to customize the operation of the MCU at boot time
 */
/*!
 * @name Constants and macros for entire NV_FSEC register
 */
/*@{*/
#define NV_RD_FSEC(base)           (NV_FSEC_REG(base))
/*}@*/

```

```

/*
 * Constants & macros for individual NV_FSEC bitfields
 */

```

```

/*!
 * @name Register NV_FSEC, field SEC[1:0] (RO)
 *
 * Values:
 * - 0b10 - MCU security status is unsecure
 * - 0b11 - MCU security status is secure
 */
/*@{*/
/*! @brief Read current value of the NV_FSEC_SEC field. */
#define NV_RD_FSEC_SEC(base) ((NV_FSEC_REG(base) & NV_FSEC_SEC_MASK) >>
NV_FSEC_SEC_SHIFT)
#define NV_BRD_FSEC_SEC(base) (NV_RD_FSEC_SEC(base))
/*}@*/

```

```

/*!
 * @name Register NV_FSEC, field FSLACC[3:2] (RO)

```

```

/*
 * Values:
 * - 0b10 - Freescale factory access denied
 * - 0b11 - Freescale factory access granted
 */
/*@{*/
/*! @brief Read current value of the NV_FSEC_FSLACC field. */
#define NV_RD_FSEC_FSLACC(base) ((NV_FSEC_REG(base) &
NV_FSEC_FSLACC_MASK) >> NV_FSEC_FSLACC_SHIFT)
#define NV_BRD_FSEC_FSLACC(base) (NV_RD_FSEC_FSLACC(base))
/*}@*/



/*!
 * @name Register NV_FSEC, field MEEN[5:4] (RO)
 *
 * Values:
 * - 0b10 - Mass erase is disabled
 * - 0b11 - Mass erase is enabled
 */
/*@{*/
/*! @brief Read current value of the NV_FSEC_MEEN field. */
#define NV_RD_FSEC_MEEN(base) ((NV_FSEC_REG(base) & NV_FSEC_MEEN_MASK) >>
NV_FSEC_MEEN_SHIFT)
#define NV_BRD_FSEC_MEEN(base) (NV_RD_FSEC_MEEN(base))
/*}@*/



/*!
 * @name Register NV_FSEC, field KEYEN[7:6] (RO)
 *
 * Values:
 * - 0b10 - Backdoor key access enabled
 * - 0b11 - Backdoor key access disabled
 */
/*@{*/
/*! @brief Read current value of the NV_FSEC_KEYEN field. */
#define NV_RD_FSEC_KEYEN(base) ((NV_FSEC_REG(base) & NV_FSEC_KEYEN_MASK)
>> NV_FSEC_KEYEN_SHIFT)
#define NV_BRD_FSEC_KEYEN(base) (NV_RD_FSEC_KEYEN(base))
/*}@*/



*****  

* NV_FOPT - Non-volatile Flash Option Register  

*****  




/*!
 * @brief NV_FOPT - Non-volatile Flash Option Register (RO)
 *
 * Reset value: 0xFFU
 */
/*!  

 * @name Constants and macros for entire NV_FOPT register

```

```

/*
/*@{ */
#define NV_RD_FOPT(base)           (NV_FOPT_REG(base))
/*}@*/ */

/*
 * Constants & macros for individual NV_FOPT bitfields
 */

/*!
 * @name Register NV_FOPT, field LPBOOT0[0] (RO)
 *
 * Values:
 * - 0b0 - Core and system clock divider (OUTDIV1) is 0x7 (divide by 8)
when
 *      LPBOOT1=0 or 0x1 (divide by 2) when LPBOOT1=1.
 * - 0b1 - Core and system clock divider (OUTDIV1) is 0x3 (divide by 4)
when
 *      LPBOOT1=0 or 0x0 (divide by 1) when LPBOOT1=1.
 */
/*@{ */

/*! @brief Read current value of the NV_FOPT_LPBOOT0 field. */
#define NV_RD_FOPT_LPBOOT0(base) ((NV_FOPT_REG(base) &
NV_FOPT_LPBOOT0_MASK) >> NV_FOPT_LPBOOT0_SHIFT)
#define NV_BRD_FOPT_LPBOOT0(base) (NV_RD_FOPT_LPBOOT0(base))
/*}@*/ */

/*!
 * @name Register NV_FOPT, field NMI_DIS[2] (RO)
 *
 * Values:
 * - 0b0 - NMI interrupts are always blocked
 * - 0b1 - NMI_b pin/interrupts reset default to enabled
 */
/*@{ */

/*! @brief Read current value of the NV_FOPT_NMI_DIS field. */
#define NV_RD_FOPT_NMI_DIS(base) ((NV_FOPT_REG(base) &
NV_FOPT_NMI_DIS_MASK) >> NV_FOPT_NMI_DIS_SHIFT)
#define NV_BRD_FOPT_NMI_DIS(base) (NV_RD_FOPT_NMI_DIS(base))
/*}@*/ */

/*!
 * @name Register NV_FOPT, field RESET_PIN_CFG[3] (RO)
 *
 * Values:
 * - 0b0 - RESET pin is disabled following a POR and cannot be enabled as
reset
 *      function
 * - 0b1 - RESET_b pin is dedicated
 */
/*@{ */

/*! @brief Read current value of the NV_FOPT_RESET_PIN_CFG field. */
#define NV_RD_FOPT_RESET_PIN_CFG(base) ((NV_FOPT_REG(base) &
NV_FOPT_RESET_PIN_CFG_MASK) >> NV_FOPT_RESET_PIN_CFG_SHIFT)

```

```

#define NV_BRD_FOPT_RESET_PIN_CFG(base)  (NV_RD_FOPT_RESET_PIN_CFG(base))
/*@}*/

/*
 * @name Register NV_FOPT, field LPBOOT1[4] (RO)
 *
 * Values:
 * - 0b0 - Core and system clock divider (OUTDIV1) is 0x7 (divide by 8)
when
 *      LPBOOT0=0 or 0x3 (divide by 4) when LPBOOT0=1.
 * - 0b1 - Core and system clock divider (OUTDIV1) is 0x1 (divide by 2)
when
 *      LPBOOT0=0 or 0x0 (divide by 1) when LPBOOT0=1.
 */
/*@{*/
/*! @brief Read current value of the NV_FOPT_LPBOOT1 field. */
#define NV_RD_FOPT_LPBOOT1(base) ((NV_FOPT_REG(base) &
NV_FOPT_LPBOOT1_MASK) >> NV_FOPT_LPBOOT1_SHIFT)
#define NV_BRD_FOPT_LPBOOT1(base) (NV_RD_FOPT_LPBOOT1(base))
/*@}*/

/*
 * @name Register NV_FOPT, field FAST_INIT[5] (RO)
 *
 * Values:
 * - 0b0 - Slower initialization
 * - 0b1 - Fast Initialization
 */
/*@{*/
/*! @brief Read current value of the NV_FOPT_FAST_INIT field. */
#define NV_RD_FOPT_FAST_INIT(base) ((NV_FOPT_REG(base) &
NV_FOPT_FAST_INIT_MASK) >> NV_FOPT_FAST_INIT_SHIFT)
#define NV_BRD_FOPT_FAST_INIT(base) (NV_RD_FOPT_FAST_INIT(base))
/*@}*/

/*
 * MKL25Z4 OSC
 *
 * Oscillator
 *
 * Registers defined in this header file:
 * - OSC_CR - OSC Control Register
 */
#define OSC_INSTANCE_COUNT (1U) /*!< Number of instances of the OSC
module. */
#define OSC0_IDX (0U) /*!< Instance number for OSC0. */

*****  

* OSC_CR - OSC Control Register  

*****  

*****/

```

```

/*!
 * @brief OSC_CR - OSC Control Register (RW)
 *
 * Reset value: 0x00U
 *
 * After OSC is enabled and starts generating the clocks, the
 * configurations
 * such as low power and frequency range, must not be changed.
 */
/*!
 * @name Constants and macros for entire OSC_CR register
 */
/*@{*/
#define OSC_RD_CR(base)          (OSC_CR_REG(base))
#define OSC_WR_CR(base, value)    (OSC_CR_REG(base) = (value))
#define OSC_RMW_CR(base, mask, value) (OSC_WR_CR(base, (OSC_RD_CR(base) &
~(mask)) | (value)))
#define OSC_SET_CR(base, value)   (BME_OR8(&OSC_CR_REG(base),
(uint8_t)(value)))
#define OSC_CLR_CR(base, value)   (BME_AND8(&OSC_CR_REG(base),
(uint8_t)(~(value))))
#define OSC_TOG_CR(base, value)   (BME_XOR8(&OSC_CR_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual OSC_CR bitfields
 */
/*!
 * @name Register OSC_CR, field SC16P[0] (RW)
 *
 * Configures the oscillator load.
 *
 * Values:
 * - 0b0 - Disable the selection.
 * - 0b1 - Add 16 pF capacitor to the oscillator load.
 */
/*@*/
/*! @brief Read current value of the OSC_CR_SC16P field. */
#define OSC_RD_CR_SC16P(base) ((OSC_CR_REG(base) & OSC_CR_SC16P_MASK) >>
OSC_CR_SC16P_SHIFT)
#define OSC_BRD_CR_SC16P(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_SC16P_SHIFT, OSC_CR_SC16P_WIDTH))

/*! @brief Set the SC16P field to a new value. */
#define OSC_WR_CR_SC16P(base, value) (OSC_RMW_CR(base, OSC_CR_SC16P_MASK,
OSC_CR_SC16P(value)))
#define OSC_BWR_CR_SC16P(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_SC16P_SHIFT), OSC_CR_SC16P_SHIFT,
OSC_CR_SC16P_WIDTH))
/*@}*/

```

```

/*!
 * @name Register OSC_CR, field SC8P[1] (RW)
 *
 * Configures the oscillator load.
 *
 * Values:
 * - 0b0 - Disable the selection.
 * - 0b1 - Add 8 pF capacitor to the oscillator load.
 */
/*@*/
/*! @brief Read current value of the OSC_CR_SC8P field. */
#define OSC_RD_CR_SC8P(base) ((OSC_CR_REG(base) & OSC_CR_SC8P_MASK) >>
OSC_CR_SC8P_SHIFT)
#define OSC_BRD_CR_SC8P(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_SC8P_SHIFT, OSC_CR_SC8P_WIDTH))

/*! @brief Set the SC8P field to a new value. */
#define OSC_WR_CR_SC8P(base, value) (OSC_RMW_CR(base, OSC_CR_SC8P_MASK,
OSC_CR_SC8P(value)))
#define OSC_BWR_CR_SC8P(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_SC8P_SHIFT), OSC_CR_SC8P_SHIFT,
OSC_CR_SC8P_WIDTH))
/*@*/ */

/*!
 * @name Register OSC_CR, field SC4P[2] (RW)
 *
 * Configures the oscillator load.
 *
 * Values:
 * - 0b0 - Disable the selection.
 * - 0b1 - Add 4 pF capacitor to the oscillator load.
 */
/*@*/
/*! @brief Read current value of the OSC_CR_SC4P field. */
#define OSC_RD_CR_SC4P(base) ((OSC_CR_REG(base) & OSC_CR_SC4P_MASK) >>
OSC_CR_SC4P_SHIFT)
#define OSC_BRD_CR_SC4P(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_SC4P_SHIFT, OSC_CR_SC4P_WIDTH))

/*! @brief Set the SC4P field to a new value. */
#define OSC_WR_CR_SC4P(base, value) (OSC_RMW_CR(base, OSC_CR_SC4P_MASK,
OSC_CR_SC4P(value)))
#define OSC_BWR_CR_SC4P(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_SC4P_SHIFT), OSC_CR_SC4P_SHIFT,
OSC_CR_SC4P_WIDTH))
/*@*/ */

/*!
 * @name Register OSC_CR, field SC2P[3] (RW)
 *
 * Configures the oscillator load.
 *
 * Values:

```

```

* - 0b0 - Disable the selection.
* - 0b1 - Add 2 pF capacitor to the oscillator load.
*/
/*@{*/
/*! @brief Read current value of the OSC_CR_SC2P field. */
#define OSC_RD_CR_SC2P(base) ((OSC_CR_REG(base) & OSC_CR_SC2P_MASK) >>
OSC_CR_SC2P_SHIFT)
#define OSC_BRD_CR_SC2P(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_SC2P_SHIFT, OSC_CR_SC2P_WIDTH))

/*! @brief Set the SC2P field to a new value. */
#define OSC_WR_CR_SC2P(base, value) (OSC_RMW_CR(base, OSC_CR_SC2P_MASK,
OSC_CR_SC2P(value)))
#define OSC_BWR_CR_SC2P(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_SC2P_SHIFT), OSC_CR_SC2P_SHIFT,
OSC_CR_SC2P_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register OSC_CR, field EREFSTEN[5] (RW)
 *
 * Controls whether or not the external reference clock (OSCERCLK)
remains
 * enabled when MCU enters Stop mode.
 *
 * Values:
 * - 0b0 - External reference clock is disabled in Stop mode.
 * - 0b1 - External reference clock stays enabled in Stop mode if ERCLKEN
is set
 * before entering Stop mode.
*/
/*@{*/
/*! @brief Read current value of the OSC_CR_EREFSTEN field. */
#define OSC_RD_CR_EREFSTEN(base) ((OSC_CR_REG(base) &
OSC_CR_EREFSTEN_MASK) >> OSC_CR_EREFSTEN_SHIFT)
#define OSC_BRD_CR_EREFSTEN(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_EREFSTEN_SHIFT, OSC_CR_EREFSTEN_WIDTH))

/*! @brief Set the EREFSTEN field to a new value. */
#define OSC_WR_CR_EREFSTEN(base, value) (OSC_RMW_CR(base,
OSC_CR_EREFSTEN_MASK, OSC_CR_EREFSTEN(value)))
#define OSC_BWR_CR_EREFSTEN(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_EREFSTEN_SHIFT), OSC_CR_EREFSTEN_SHIFT,
OSC_CR_EREFSTEN_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register OSC_CR, field ERCLKEN[7] (RW)
 *
 * Enables external reference clock (OSCERCLK).
 *
 * Values:
 * - 0b0 - External reference clock is inactive.
 * - 0b1 - External reference clock is enabled.
```

```

*/
/*@{*/
/*! @brief Read current value of the OSC_CR_ERCLKEN field. */
#define OSC_RD_CR_ERCLKEN(base) ((OSC_CR_REG(base) & OSC_CR_ERCLKEN_MASK)
>> OSC_CR_ERCLKEN_SHIFT)
#define OSC_BRD_CR_ERCLKEN(base) (BME_UBFX8(&OSC_CR_REG(base),
OSC_CR_ERCLKEN_SHIFT, OSC_CR_ERCLKEN_WIDTH))

/*! @brief Set the ERCLKEN field to a new value. */
#define OSC_WR_CR_ERCLKEN(base, value) (OSC_RMW_CR(base,
OSC_CR_ERCLKEN_MASK, OSC_CR_ERCLKEN(value)))
#define OSC_BWR_CR_ERCLKEN(base, value) (BME_BFI8(&OSC_CR_REG(base),
((uint8_t)(value) << OSC_CR_ERCLKEN_SHIFT), OSC_CR_ERCLKEN_SHIFT,
OSC_CR_ERCLKEN_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 PIT
 *
 * Periodic Interrupt Timer
 *
 * Registers defined in this header file:
 * - PIT_MCR - PIT Module Control Register
 * - PIT_LTMR64H - PIT Upper Lifetime Timer Register
 * - PIT_LTMR64L - PIT Lower Lifetime Timer Register
 * - PIT_LDVAL - Timer Load Value Register
 * - PIT_CVAL - Current Timer Value Register
 * - PIT_TCTRL - Timer Control Register
 * - PIT_TFLG - Timer Flag Register
 */
#define PIT_INSTANCE_COUNT (1U) /*!< Number of instances of the PIT
module. */
#define PIT_IDX (0U) /*!< Instance number for PIT. */

*****  

* PIT_MCR - PIT Module Control Register  

*****  

*  

/*!  

 * @brief PIT_MCR - PIT Module Control Register (RW)  

 *  

 * Reset value: 0x00000002U  

 *  

 * This register enables or disables the PIT timer clocks and controls  

the  

* timers when the PIT enters the Debug mode.  

 */  

/*!  

 * @name Constants and macros for entire PIT_MCR register  

 */

```

```

/*@{*/
#define PIT_RD_MCR(base)          (PIT_MCR_REG(base))
#define PIT_WR_MCR(base, value)    (PIT_MCR_REG(base) = (value))
#define PIT_RMW_MCR(base, mask, value) (PIT_WR_MCR(base,
(PIT_RD_MCR(base) & ~mask) | (value)))
#define PIT_SET_MCR(base, value)   (BME_OR32(&PIT_MCR_REG(base),
(uint32_t)(value)))
#define PIT_CLR_MCR(base, value)   (BME_AND32(&PIT_MCR_REG(base),
(uint32_t)(~(value))))
#define PIT_TOG_MCR(base, value)   (BME_XOR32(&PIT_MCR_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual PIT_MCR bitfields
 */

/*!
 * @name Register PIT_MCR, field FRZ[0] (RW)
 *
 * Allows the timers to be stopped when the device enters the Debug mode.
 *
 * Values:
 * - 0b0 - Timers continue to run in Debug mode.
 * - 0b1 - Timers are stopped in Debug mode.
 */
/*@*/
/*! @brief Read current value of the PIT_MCR_FRZ field. */
#define PIT_RD_MCR_FRZ(base) ((PIT_MCR_REG(base) & PIT_MCR_FRZ_MASK) >>
PIT_MCR_FRZ_SHIFT)
#define PIT_BRD_MCR_FRZ(base) (BME_UBFX32(&PIT_MCR_REG(base),
PIT_MCR_FRZ_SHIFT, PIT_MCR_FRZ_WIDTH))

/*! @brief Set the FRZ field to a new value. */
#define PIT_WR_MCR_FRZ(base, value) (PIT_RMW_MCR(base, PIT_MCR_FRZ_MASK,
PIT_MCR_FRZ(value)))
#define PIT_BWR_MCR_FRZ(base, value) (BME_BFI32(&PIT_MCR_REG(base),
((uint32_t)(value) << PIT_MCR_FRZ_SHIFT), PIT_MCR_FRZ_SHIFT,
PIT_MCR_FRZ_WIDTH))
/*@*/ */

/*!
 * @name Register PIT_MCR, field MDIS[1] (RW)
 *
 * Disables the standard timers. The RTI timer is not affected by this
field.
 * This field must be enabled before any other setup is done.
 *
 * Values:
 * - 0b0 - Clock for standard PIT timers is enabled.
 * - 0b1 - Clock for standard PIT timers is disabled.
 */
/*@*/
/*! @brief Read current value of the PIT_MCR_MDIS field. */

```

```

#define PIT_RD_MCR_MDIS(base) ((PIT_MCR_REG(base) & PIT_MCR_MDIS_MASK) >>
PIT_MCR_MDIS_SHIFT)
#define PIT_BRD_MCR_MDIS(base) (BME_UBFX32(&PIT_MCR_REG(base),
PIT_MCR_MDIS_SHIFT, PIT_MCR_MDIS_WIDTH))

/*! @brief Set the MDIS field to a new value. */
#define PIT_WR_MCR_MDIS(base, value) (PIT_RMW_MCR(base,
PIT_MCR_MDIS_MASK, PIT_MCR_MDIS(value)))
#define PIT_BWR_MCR_MDIS(base, value) (BME_BFI32(&PIT_MCR_REG(base),
((uint32_t)(value) << PIT_MCR_MDIS_SHIFT), PIT_MCR_MDIS_SHIFT,
PIT_MCR_MDIS_WIDTH))
/*@}*/

/*********************  

*****  

 * PIT_LTMR64H - PIT Upper Lifetime Timer Register  

*****/  

  

/*!  

 * @brief PIT_LTMR64H - PIT Upper Lifetime Timer Register (RO)  

 *  

 * Reset value: 0x00000000U  

 *  

 * This register is intended for applications that chain timer 0 and  

timer 1 to  

 * build a 64-bit lifetimer.  

 */  

/**!  

 * @name Constants and macros for entire PIT_LTMR64H register  

 */  

/*@*/  

#define PIT_RD_LTMR64H(base) (PIT_LTMR64H_REG(base))  

/*@*/  

  

/*********************  

*****  

 * PIT_LTMR64L - PIT Lower Lifetime Timer Register  

*****/  

  

/*!  

 * @brief PIT_LTMR64L - PIT Lower Lifetime Timer Register (RO)  

 *  

 * Reset value: 0x00000000U  

 *  

 * This register is intended for applications that chain timer 0 and  

timer 1 to  

 * build a 64-bit lifetimer. To use LTMR64H and LTMR64L, timer 0 and  

timer 1 need  

 * to be chained. To obtain the correct value, first read LTMR64H and  

then

```

```

 * LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first
access,
 * LTMR64L will have the value of CVAL0 at the time of the first access,
therefore
 * the application does not need to worry about carry-over effects of the
running
 * counter.
 */
/*!
 * @name Constants and macros for entire PIT_LTMR64L register
 */
/*@{ */
#define PIT_RD_LTMR64L(base)      (PIT_LTMR64L_REG(base))
/*}@*/



/********************* ****
 * PIT_LDVAL - Timer Load Value Register
***** ****
 */

/*!
 * @brief PIT_LDVAL - Timer Load Value Register (RW)
 *
 * Reset value: 0x00000000U
 *
 * These registers select the timeout period for the timer interrupts.
 */
/*!
 * @name Constants and macros for entire PIT_LDVAL register
 */
/*@{ */
#define PIT_RD_LDVAL(base, index) (PIT_LDVAL_REG(base, index))
#define PIT_WR_LDVAL(base, index, value) (PIT_LDVAL_REG(base, index) = (value))
#define PIT_RMW_LDVAL(base, index, mask, value) (PIT_WR_LDVAL(base, index, (PIT_RD_LDVAL(base, index) & ~mask) | value)))
#define PIT_SET_LDVAL(base, index, value) (BME_OR32(&PIT_LDVAL_REG(base, index), (uint32_t)(value)))
#define PIT_CLR_LDVAL(base, index, value) (BME_AND32(&PIT_LDVAL_REG(base, index), (uint32_t)(~value))))
#define PIT_TOG_LDVAL(base, index, value) (BME_XOR32(&PIT_LDVAL_REG(base, index), (uint32_t)(value)))
/*}@*/



/********************* ****
 * PIT_CVAL - Current Timer Value Register
***** ****
 */

/*!

```

```

* @brief PIT_CVAL - Current Timer Value Register (RO)
*
* Reset value: 0x00000000U
*
* These registers indicate the current timer position.
*/
/*!
* @name Constants and macros for entire PIT_CVAL register
*/
/*@{*/
#define PIT_RD_CVAL(base, index) (PIT_CVAL_REG(base, index))
/*}@*/

```

---

```

*****  

* PIT_TCTRL - Timer Control Register  

*****  

*/!  

* @brief PIT_TCTRL - Timer Control Register (RW)  

*  

* Reset value: 0x00000000U  

*  

* These register contain the control bits for each timer.
*/
/*!
* @name Constants and macros for entire PIT_TCTRL register
*/
/*@{*/
#define PIT_RD_TCTRL(base, index) (PIT_TCTRL_REG(base, index))
#define PIT_WR_TCTRL(base, index, value) (PIT_TCTRL_REG(base, index) =  

(value))
#define PIT_RMW_TCTRL(base, index, mask, value) (PIT_WR_TCTRL(base,  

index, (PIT_RD_TCTRL(base, index) & ~(mask)) | (value)))
#define PIT_SET_TCTRL(base, index, value) (BME_OR32(&PIT_TCTRL_REG(base,  

index), (uint32_t)(value)))
#define PIT_CLR_TCTRL(base, index, value) (BME_AND32(&PIT_TCTRL_REG(base,  

index), (uint32_t)(~(value))))
#define PIT_TOG_TCTRL(base, index, value) (BME_XOR32(&PIT_TCTRL_REG(base,  

index), (uint32_t)(value)))
/*}@*/

```

---

```

/*
* Constants & macros for individual PIT_TCTRL bitfields
*/

```

---

```

/*!
* @name Register PIT_TCTRL, field TEN[0] (RW)
*  

* Enables or disables the timer.
*  

* Values:

```

```

* - 0b0 - Timer n is disabled.
* - 0b1 - Timer n is enabled.
*/
/*@{*/
/*! @brief Read current value of the PIT_TCTRL_TEN field. */
#define PIT_RD_TCTRL_TEN(base, index) ((PIT_TCTRL_REG(base, index) &
PIT_TCTRL_TEN_MASK) >> PIT_TCTRL_TEN_SHIFT)
#define PIT_BRD_TCTRL_TEN(base, index) (BME_UBFX32(&PIT_TCTRL_REG(base,
index), PIT_TCTRL_TEN_SHIFT, PIT_TCTRL_TEN_WIDTH))

/*! @brief Set the TEN field to a new value. */
#define PIT_WR_TCTRL_TEN(base, index, value) (PIT_RMW_TCTRL(base, index,
PIT_TCTRL_TEN_MASK, PIT_TCTRL_TEN(value)))
#define PIT_BWR_TCTRL_TEN(base, index, value)
(BME_BFI32(&PIT_TCTRL_REG(base, index), ((uint32_t)(value) <<
PIT_TCTRL_TEN_SHIFT), PIT_TCTRL_TEN_SHIFT, PIT_TCTRL_TEN_WIDTH))
/*@}*/

/*
* @name Register PIT_TCTRL, field TIE[1] (RW)
*
* When an interrupt is pending, or, TFLGn[TIF] is set, enabling the
interrupt
* will immediately cause an interrupt event. To avoid this, the
associated
* TFLGn[TIF] must be cleared first.
*
* Values:
* - 0b0 - Interrupt requests from Timer n are disabled.
* - 0b1 - Interrupt will be requested whenever TIF is set.
*/
/*@{*/
/*! @brief Read current value of the PIT_TCTRL_TIE field. */
#define PIT_RD_TCTRL_TIE(base, index) ((PIT_TCTRL_REG(base, index) &
PIT_TCTRL_TIE_MASK) >> PIT_TCTRL_TIE_SHIFT)
#define PIT_BRD_TCTRL_TIE(base, index) (BME_UBFX32(&PIT_TCTRL_REG(base,
index), PIT_TCTRL_TIE_SHIFT, PIT_TCTRL_TIE_WIDTH))

/*! @brief Set the TIE field to a new value. */
#define PIT_WR_TCTRL_TIE(base, index, value) (PIT_RMW_TCTRL(base, index,
PIT_TCTRL_TIE_MASK, PIT_TCTRL_TIE(value)))
#define PIT_BWR_TCTRL_TIE(base, index, value)
(BME_BFI32(&PIT_TCTRL_REG(base, index), ((uint32_t)(value) <<
PIT_TCTRL_TIE_SHIFT), PIT_TCTRL_TIE_SHIFT, PIT_TCTRL_TIE_WIDTH))
/*@}*/

/*
* @name Register PIT_TCTRL, field CHN[2] (RW)
*
* When activated, Timer n-1 needs to expire before timer n can decrement
by 1.
* Timer 0 can not be changed.
*
* Values:

```

```

* - 0b0 - Timer is not chained.
* - 0b1 - Timer is chained to previous timer. For example, for Channel
2, if
*      this field is set, Timer 2 is chained to Timer 1.
*/
/*@{*/
/*! @brief Read current value of the PIT_TCTRL_CHN field. */
#define PIT_RD_TCTRL_CHN(base, index) ((PIT_TCTRL_REG(base, index) &
PIT_TCTRL_CHN_MASK) >> PIT_TCTRL_CHN_SHIFT)
#define PIT_BRD_TCTRL_CHN(base, index) (BME_UBFX32(&PIT_TCTRL_REG(base,
index), PIT_TCTRL_CHN_SHIFT, PIT_TCTRL_CHN_WIDTH))

/*! @brief Set the CHN field to a new value. */
#define PIT_WR_TCTRL_CHN(base, index, value) (PIT_RMW_TCTRL(base, index,
PIT_TCTRL_CHN_MASK, PIT_TCTRL_CHN(value)))
#define PIT_BWR_TCTRL_CHN(base, index, value)
(BME_BFI32(&PIT_TCTRL_REG(base, index), ((uint32_t)(value) <<
PIT_TCTRL_CHN_SHIFT), PIT_TCTRL_CHN_SHIFT, PIT_TCTRL_CHN_WIDTH))
/*}@*/ */

/********************* */
***** */
* PIT_TFLG - Timer Flag Register
***** */
/*!
* @brief PIT_TFLG - Timer Flag Register (RW)
*
* Reset value: 0x00000000U
*
* These registers hold the PIT interrupt flags.
*/
/*!
* @name Constants and macros for entire PIT_TFLG register
*/
/*@{*/
#define PIT_RD_TFLG(base, index) (PIT_TFLG_REG(base, index))
#define PIT_WR_TFLG(base, index, value) (PIT_TFLG_REG(base, index) =
(value))
#define PIT_RMW_TFLG(base, index, mask, value) (PIT_WR_TFLG(base, index,
(PIT_RD_TFLG(base, index) & ~mask)) | (value)))
#define PIT_SET_TFLG(base, index, value) (BME_OR32(&PIT_TFLG_REG(base,
index), (uint32_t)(value)))
#define PIT_CLR_TFLG(base, index, value) (BME_AND32(&PIT_TFLG_REG(base,
index), (uint32_t)(~value)))
#define PIT_TOG_TFLG(base, index, value) (BME_XOR32(&PIT_TFLG_REG(base,
index), (uint32_t)(value)))
/*}@*/ */

/*
* Constants & macros for individual PIT_TFLG bitfields
*/

```

```

/*!
 * @name Register PIT_TFLG, field TIF[0] (W1C)
 *
 * Sets to 1 at the end of the timer period. Writing 1 to this flag
 * clears it.
 * Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF
 * causes an
 * interrupt request.
 *
 * Values:
 * - 0b0 - Timeout has not yet occurred.
 * - 0b1 - Timeout has occurred.
 */
/*@{*/
/*! @brief Read current value of the PIT_TFLG_TIF field. */
#define PIT_RD_TFLG_TIF(base, index) ((PIT_TFLG_REG(base, index) &
PIT_TFLG_TIF_MASK) >> PIT_TFLG_TIF_SHIFT)
#define PIT_BRD_TFLG_TIF(base, index) (BME_UBFX32(&PIT_TFLG_REG(base,
index), PIT_TFLG_TIF_SHIFT, PIT_TFLG_TIF_WIDTH))

/*! @brief Set the TIF field to a new value. */
#define PIT_WR_TFLG_TIF(base, index, value) (PIT_RMW_TFLG(base, index,
PIT_TFLG_TIF_MASK, PIT_TFLG_TIF(value)))
#define PIT_BWR_TFLG_TIF(base, index, value)
(BME_BFI32(&PIT_TFLG_REG(base, index), ((uint32_t)(value) <<
PIT_TFLG_TIF_SHIFT), PIT_TFLG_TIF_SHIFT, PIT_TFLG_TIF_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 PMC
 *
 * Power Management Controller
 *
 * Registers defined in this header file:
 * - PMC_LVDSC1 - Low Voltage Detect Status And Control 1 register
 * - PMC_LVDSC2 - Low Voltage Detect Status And Control 2 register
 * - PMC_REGSC - Regulator Status And Control register
 */
#define PMC_INSTANCE_COUNT (1U) /*!< Number of instances of the PMC
module. */
#define PMC_IDX (0U) /*!< Instance number for PMC. */

*****  

* PMC_LVDSC1 - Low Voltage Detect Status And Control 1 register  

*****/  

/*!
 * @brief PMC_LVDSC1 - Low Voltage Detect Status And Control 1 register
(RW)

```

```

/*
 * Reset value: 0x10U
 *
 * This register contains status and control bits to support the low
voltage
 * detect function. This register should be written during the reset
initialization
 * program to set the desired controls even if the desired settings are
the same
 * as the reset settings. While the device is in the very low power or
low
 * leakage modes, the LVD system is disabled regardless of LVDSC1
settings. To protect
 * systems that must have LVD always on, configure the SMC's power mode
 * protection register (PMPROT) to disallow any very low power or low
leakage modes from
 * being enabled. See the device's data sheet for the exact LVD trip
voltages. The
 * LVDV bits are reset solely on a POR Only event. The register's other
bits are
 * reset on Chip Reset Not VLLS. For more information about these reset
types,
 * refer to the Reset section details.
 */
/*!
 * @name Constants and macros for entire PMC_LVDSC1 register
 */
/*@{*/
#define PMC_RD_LVDSC1(base)          (PMC_LVDSC1_REG(base))
#define PMC_WR_LVDSC1(base, value)   (PMC_LVDSC1_REG(base) = (value))
#define PMC_RMW_LVDSC1(base, mask, value) (PMC_WR_LVDSC1(base,
(PCM_RD_LVDSC1(base) & ~mask)) | (value)))
#define PMC_SET_LVDSC1(base, value)  (BME_OR8(&PMC_LVDSC1_REG(base),
(uint8_t)(value)))
#define PMC_CLR_LVDSC1(base, value)  (BME_AND8(&PMC_LVDSC1_REG(base),
(uint8_t)(~value)))
#define PMC_TOG_LVDSC1(base, value)  (BME_XOR8(&PMC_LVDSC1_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual PMC_LVDSC1 bitfields
 */
/*!
 * @name Register PMC_LVDSC1, field LVDV[1:0] (RW)
 *
 * Selects the LVD trip point voltage (V LVD).
 *
 * Values:
 * - 0b00 - Low trip point selected (V LVD = V LVDL)
 * - 0b01 - High trip point selected (V LVD = V LVDH)
 * - 0b10 - Reserved
 * - 0b11 - Reserved

```

```

/*
/*@{*/
/*! @brief Read current value of the PMC_LVDSC1_LVDV field. */
#define PMC_RD_LVDSC1_LVDV(base) ((PMC_LVDSC1_REG(base) &
PMC_LVDSC1_LVDV_MASK) >> PMC_LVDSC1_LVDV_SHIFT)
#define PMC_BRD_LVDSC1_LVDV(base) (BME_UBFX8(&PMC_LVDSC1_REG(base),
PMC_LVDSC1_LVDV_SHIFT, PMC_LVDSC1_LVDV_WIDTH))

/*! @brief Set the LVDV field to a new value. */
#define PMC_WR_LVDSC1_LVDV(base, value) (PMC_RMW_LVDSC1(base,
PMC_LVDSC1_LVDV_MASK, PMC_LVDSC1_LVDV(value)))
#define PMC_BWR_LVDSC1_LVDV(base, value) (BME_BFI8(&PMC_LVDSC1_REG(base),
((uint8_t)(value) << PMC_LVDSC1_LVDV_SHIFT), PMC_LVDSC1_LVDV_SHIFT,
PMC_LVDSC1_LVDV_WIDTH))
/*}@*/ */

/*!
* @name Register PMC_LVDSC1, field LVDRE[4] (RW)
*
* This write-once bit enables LVDF events to generate a hardware reset.
* Additional writes are ignored.
*
* Values:
* - 0b0 - LVDF does not generate hardware resets
* - 0b1 - Force an MCU reset when LVDF = 1
*/
/*@{*/
/*! @brief Read current value of the PMC_LVDSC1_LVDRE field. */
#define PMC_RD_LVDSC1_LVDRE(base) ((PMC_LVDSC1_REG(base) &
PMC_LVDSC1_LVDRE_MASK) >> PMC_LVDSC1_LVDRE_SHIFT)
#define PMC_BRD_LVDSC1_LVDRE(base) (BME_UBFX8(&PMC_LVDSC1_REG(base),
PMC_LVDSC1_LVDRE_SHIFT, PMC_LVDSC1_LVDRE_WIDTH))

/*! @brief Set the LVDRE field to a new value. */
#define PMC_WR_LVDSC1_LVDRE(base, value) (PMC_RMW_LVDSC1(base,
PMC_LVDSC1_LVDRE_MASK, PMC_LVDSC1_LVDRE(value)))
#define PMC_BWR_LVDSC1_LVDRE(base, value)
(BME_BFI8(&PMC_LVDSC1_REG(base), ((uint8_t)(value) <<
PMC_LVDSC1_LVDRE_SHIFT), PMC_LVDSC1_LVDRE_SHIFT, PMC_LVDSC1_LVDRE_WIDTH))
/*}@*/ */

/*!
* @name Register PMC_LVDSC1, field LVDIE[5] (RW)
*
* Enables hardware interrupt requests for LVDF.
*
* Values:
* - 0b0 - Hardware interrupt disabled (use polling)
* - 0b1 - Request a hardware interrupt when LVDF = 1
*/
/*@{*/
/*! @brief Read current value of the PMC_LVDSC1_LVDIE field. */
#define PMC_RD_LVDSC1_LVDIE(base) ((PMC_LVDSC1_REG(base) &
PMC_LVDSC1_LVDIE_MASK) >> PMC_LVDSC1_LVDIE_SHIFT)

```

```

#define PMC_BRD_LVDSC1_LVDIE(base)  (BME_UBFX8(&PMC_LVDSC1_REG(base),  

PMC_LVDSC1_LVDIE_SHIFT, PMC_LVDSC1_LVDIE_WIDTH))

/*! @brief Set the LVDIE field to a new value. */  

#define PMC_WR_LVDSC1_LVDIE(base, value)  (PMC_RMW_LVDSC1(base,  

PMC_LVDSC1_LVDIE_MASK, PMC_LVDSC1_LVDIE(value)))  

#define PMC_BWR_LVDSC1_LVDIE(base, value)  

(BME_BFI8(&PMC_LVDSC1_REG(base), ((uint8_t)(value) <<  

PMC_LVDSC1_LVDIE_SHIFT), PMC_LVDSC1_LVDIE_SHIFT, PMC_LVDSC1_LVDIE_WIDTH))  

/*@}*/

/*!  

 * @name Register PMC_LVDSC1, field LVDACK[6] (WORZ)  

 *  

 * This write-only bit is used to acknowledge low voltage detection  

errors.  

 * Write 1 to clear LVDF. Reads always return 0.  

 */  

/*@{*/  

/*! @brief Set the LVDACK field to a new value. */  

#define PMC_WR_LVDSC1_LVDACK(base, value)  (PMC_RMW_LVDSC1(base,  

PMC_LVDSC1_LVDACK_MASK, PMC_LVDSC1_LVDACK(value)))  

#define PMC_BWR_LVDSC1_LVDACK(base, value)  

(BME_BFI8(&PMC_LVDSC1_REG(base), ((uint8_t)(value) <<  

PMC_LVDSC1_LVDACK_SHIFT), PMC_LVDSC1_LVDACK_SHIFT,  

PMC_LVDSC1_LVDACK_WIDTH))  

/*@}*/

/*!  

 * @name Register PMC_LVDSC1, field LVDF[7] (RO)  

 *  

 * This read-only status bit indicates a low-voltage detect event.  

 *  

 * Values:  

 * - 0b0 - Low-voltage event not detected  

 * - 0b1 - Low-voltage event detected  

 */  

/*@{*/  

/*! @brief Read current value of the PMC_LVDSC1_LVDF field. */  

#define PMC_RD_LVDSC1_LVDF(base)  ((PMC_LVDSC1_REG(base) &  

PMC_LVDSC1_LVDF_MASK) >> PMC_LVDSC1_LVDF_SHIFT)  

#define PMC_BRD_LVDSC1_LVDF(base)  (BME_UBFX8(&PMC_LVDSC1_REG(base),  

PMC_LVDSC1_LVDF_SHIFT, PMC_LVDSC1_LVDF_WIDTH))  

/*@}*/

*****  

*****  

* PMC_LVDSC2 - Low Voltage Detect Status And Control 2 register  

*****  

*****/  

/*!

```

```

 * @brief PMC_LVDSC2 - Low Voltage Detect Status And Control 2 register
(RW)
 *
 * Reset value: 0x00U
 *
 * This register contains status and control bits to support the low
voltage
 * warning function. While the device is in the very low power or low
leakage modes,
 * the LVD system is disabled regardless of LVDSC2 settings. See the
device's
 * data sheet for the exact LVD trip voltages. The LVW trip voltages
depend on LVWV
 * and LVDV bits. The LVWV bits are reset solely on a POR Only event. The
 * register's other bits are reset on Chip Reset Not VLLS. For more
information about
 * these reset types, refer to the Reset section details.
 */
/*!
 * @name Constants and macros for entire PMC_LVDSC2 register
 */
/*@{*/
#define PMC_RD_LVDSC2(base)      (PMC_LVDSC2_REG(base))
#define PMC_WR_LVDSC2(base, value) (PMC_LVDSC2_REG(base) = (value))
#define PMC_RMW_LVDSC2(base, mask, value) (PMC_WR_LVDSC2(base,
(PCM_RD_LVDSC2(base) & ~mask)) | (value)))
#define PMC_SET_LVDSC2(base, value) (BME_OR8(&PMC_LVDSC2_REG(base),
(uint8_t)(value)))
#define PMC_CLR_LVDSC2(base, value) (BME_AND8(&PMC_LVDSC2_REG(base),
(uint8_t)(~value)))
#define PMC_TOG_LVDSC2(base, value) (BME_XOR8(&PMC_LVDSC2_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual PMC_LVDSC2 bitfields
 */
/*!
 * @name Register PMC_LVDSC2, field LVWV[1:0] (RW)
 *
 * Selects the LVW trip point voltage (VLVW). The actual voltage for the
warning
 * depends on LVDSC1[LVDV].
 *
 * Values:
 * - 0b00 - Low trip point selected (VLVW = VLVW1)
 * - 0b01 - Mid 1 trip point selected (VLVW = VLVW2)
 * - 0b10 - Mid 2 trip point selected (VLVW = VLVW3)
 * - 0b11 - High trip point selected (VLVW = VLVW4)
 */
/*@*/
/*! @brief Read current value of the PMC_LVDSC2_LVWV field. */

```

```

#define PMC_RD_LVDSC2_LVWV(base) ((PMC_LVDSC2_REG(base) &
PMC_LVDSC2_LVWV_MASK) >> PMC_LVDSC2_LVWV_SHIFT)
#define PMC_BRD_LVDSC2_LVWV(base) (BME_UBFX8(&PMC_LVDSC2_REG(base),
PMC_LVDSC2_LVWV_SHIFT, PMC_LVDSC2_LVWV_WIDTH))

/*! @brief Set the LVWV field to a new value. */
#define PMC_WR_LVDSC2_LVWV(base, value) (PMC_RMW_LVDSC2(base,
PMC_LVDSC2_LVWV_MASK, PMC_LVDSC2_LVWV(value)))
#define PMC_BWR_LVDSC2_LVWV(base, value) (BME_BFI8(&PMC_LVDSC2_REG(base),
((uint8_t)(value) << PMC_LVDSC2_LVWV_SHIFT), PMC_LVDSC2_LVWV_SHIFT,
PMC_LVDSC2_LVWV_WIDTH))
/*@}*/

/*!
 * @name Register PMC_LVDSC2, field LVWIE[5] (RW)
 *
 * Enables hardware interrupt requests for LVWF.
 *
 * Values:
 * - 0b0 - Hardware interrupt disabled (use polling)
 * - 0b1 - Request a hardware interrupt when LVWF = 1
 */
/*@{*/
/*! @brief Read current value of the PMC_LVDSC2_LVWIE field. */
#define PMC_RD_LVDSC2_LVWIE(base) ((PMC_LVDSC2_REG(base) &
PMC_LVDSC2_LVWIE_MASK) >> PMC_LVDSC2_LVWIE_SHIFT)
#define PMC_BRD_LVDSC2_LVWIE(base) (BME_UBFX8(&PMC_LVDSC2_REG(base),
PMC_LVDSC2_LVWIE_SHIFT, PMC_LVDSC2_LVWIE_WIDTH))

/*! @brief Set the LVWIE field to a new value. */
#define PMC_WR_LVDSC2_LVWIE(base, value) (PMC_RMW_LVDSC2(base,
PMC_LVDSC2_LVWIE_MASK, PMC_LVDSC2_LVWIE(value)))
#define PMC_BWR_LVDSC2_LVWIE(base, value)
(BME_BFI8(&PMC_LVDSC2_REG(base), ((uint8_t)(value) <<
PMC_LVDSC2_LVWIE_SHIFT), PMC_LVDSC2_LVWIE_SHIFT, PMC_LVDSC2_LVWIE_WIDTH))
/*@}*/

/*!
 * @name Register PMC_LVDSC2, field LVWACK[6] (WORZ)
 *
 * This write-only bit is used to acknowledge low voltage warning errors.
Write
 * 1 to clear LVWF. Reads always return 0.
 */
/*@{*/
/*! @brief Set the LVWACK field to a new value. */
#define PMC_WR_LVDSC2_LVWACK(base, value) (PMC_RMW_LVDSC2(base,
PMC_LVDSC2_LVWACK_MASK, PMC_LVDSC2_LVWACK(value)))
#define PMC_BWR_LVDSC2_LVWACK(base, value)
(BME_BFI8(&PMC_LVDSC2_REG(base), ((uint8_t)(value) <<
PMC_LVDSC2_LVWACK_SHIFT), PMC_LVDSC2_LVWACK_SHIFT,
PMC_LVDSC2_LVWACK_WIDTH))
/*@}*/

```

```

/*!
 * @name Register PMC_LVDSC2, field LVWF[7] (RO)
 *
 * This read-only status bit indicates a low-voltage warning event. LVWF
is set
 * when VSupply transitions below the trip point, or after reset and
VSupply is
 * already below VLVW. LVWF bit may be 1 after power on reset, therefore,
to use
 * LVW interrupt function, before enabling LVWIE, LVWF must be cleared by
writing
 * LWWACK first.
 *
 * Values:
 * - 0b0 - Low-voltage warning event not detected
 * - 0b1 - Low-voltage warning event detected
 */
/*@{*/
/*! @brief Read current value of the PMC_LVDSC2_LVWF field. */
#define PMC_RD_LVDSC2_LVWF(base) ((PMC_LVDSC2_REG(base) &
PMC_LVDSC2_LVWF_MASK) >> PMC_LVDSC2_LVWF_SHIFT)
#define PMC_BRD_LVDSC2_LVWF(base) (BME_UBFX8(&PMC_LVDSC2_REG(base),
PMC_LVDSC2_LVWF_SHIFT, PMC_LVDSC2_LVWF_WIDTH))
/*@}*/
*****  

* PMC_REGSC - Regulator Status And Control register  

*****  

/*!
 * @brief PMC_REGSC - Regulator Status And Control register (RW)
 *
 * Reset value: 0x04U
 *
 * The PMC contains an internal voltage regulator. The voltage regulator
design
 * uses a bandgap reference that is also available through a buffer as
input to
 * certain internal peripherals, such as the CMP and ADC. The internal
regulator
 * provides a status bit (REGONS) indicating the regulator is in run
regulation.
 * This register is reset on Chip Reset Not VLLS and by reset types that
trigger
 * Chip Reset not VLLS. See the Reset section for more information.
 */
/*!
 * @name Constants and macros for entire PMC_REGSC register
 */
/*@*/
#define PMC_RD_REGSC(base) (PMC_REGSC_REG(base))

```

```

#define PMC_WR_REGSC(base, value) (PMC_REGSC_REG(base) = (value))
#define PMC_RMW_REGSC(base, mask, value) (PMC_WR_REGSC(base,
(PCM_RD_REGSC(base) & ~mask) | (value)))
#define PMC_SET_REGSC(base, value) (BME_OR8(&PMC_REGSC_REG(base),
(uint8_t)(value)))
#define PMC_CLR_REGSC(base, value) (BME_AND8(&PMC_REGSC_REG(base),
(uint8_t)(~(value))))
#define PMC_TOG_REGSC(base, value) (BME_XOR8(&PMC_REGSC_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual PMC_REGSC bitfields
 */

/*!!
 * @name Register PMC_REGSC, field BGBE[0] (RW)
 *
 * Enables the bandgap buffer.
 *
 * Values:
 * - 0b0 - Bandgap buffer not enabled
 * - 0b1 - Bandgap buffer enabled
 */
/*@*/
/*! @brief Read current value of the PMC_REGSC_BGBE field. */
#define PMC_RD_REGSC_BGBE(base) ((PMC_REGSC_REG(base) &
PMC_REGSC_BGBE_MASK) >> PMC_REGSC_BGBE_SHIFT)
#define PMC_BRD_REGSC_BGBE(base) (BME_UBFX8(&PMC_REGSC_REG(base),
PMC_REGSC_BGBE_SHIFT, PMC_REGSC_BGBE_WIDTH))

/*!! @brief Set the BGBE field to a new value. */
#define PMC_WR_REGSC_BGBE(base, value) (PMC_RMW_REGSC(base,
(PCM_REGSC_BGBE_MASK | PMC_REGSC_ACKISO_MASK), PMC_REGSC_BGBE(value)))
#define PMC_BWR_REGSC_BGBE(base, value) (BME_BFI8(&PMC_REGSC_REG(base),
((uint8_t)(value) << PMC_REGSC_BGBE_SHIFT), PMC_REGSC_BGBE_SHIFT,
PMC_REGSC_BGBE_WIDTH))
/*@*/

/*!!
 * @name Register PMC_REGSC, field REGONS[2] (RO)
 *
 * This read-only bit provides the current status of the internal voltage
 * regulator.
 *
 * Values:
 * - 0b0 - Regulator is in stop regulation or in transition to/from it
 * - 0b1 - Regulator is in run regulation
 */
/*@*/
/*! @brief Read current value of the PMC_REGSC_REGONS field. */
#define PMC_RD_REGSC_REGONS(base) ((PMC_REGSC_REG(base) &
PMC_REGSC_REGONS_MASK) >> PMC_REGSC_REGONS_SHIFT)

```

```

#define PMC_BRD_REGSC_REGONS(base) (BME_UBFX8(&PMC_REGSC_REG(base),  

PMC_REGSC_REGONS_SHIFT, PMC_REGSC_REGONS_WIDTH))  

/*@}*/

/*!  

 * @name Register PMC_REGSC, field ACKISO[3] (W1C)  

 *  

 * Reading this bit indicates whether certain peripherals and the I/O  

pads are  

 * in a latched state as a result of having been in a VLLS mode. Writing  

one to  

 * this bit when it is set releases the I/O pads and certain peripherals  

to their  

 * normal run mode state. After recovering from a VLLS mode, user should  

restore  

 * chip configuration before clearing ACKISO. In particular, pin  

configuration for  

 * enabled LLWU wakeup pins should be restored to avoid any LLWU flag  

from being  

 * falsely set when ACKISO is cleared.  

 *  

 * Values:  

 * - 0b0 - Peripherals and I/O pads are in normal run state  

 * - 0b1 - Certain peripherals and I/O pads are in an isolated and  

latched state  

 */  

/*@*/  

/*! @brief Read current value of the PMC_REGSC_ACKISO field. */  

#define PMC_RD_REGSC_ACKISO(base) ((PMC_REGSC_REG(base) &  

PMC_REGSC_ACKISO_MASK) >> PMC_REGSC_ACKISO_SHIFT)  

#define PMC_BRD_REGSC_ACKISO(base) (BME_UBFX8(&PMC_REGSC_REG(base),  

PMC_REGSC_ACKISO_SHIFT, PMC_REGSC_ACKISO_WIDTH))

/*! @brief Set the ACKISO field to a new value. */  

#define PMC_WR_REGSC_ACKISO(base, value) (PMC_RMW_REGSC(base,  

PMC_REGSC_ACKISO_MASK, PMC_REGSC_ACKISO(value)))  

#define PMC_BWR_REGSC_ACKISO(base, value) (BME_BFI8(&PMC_REGSC_REG(base),  

((uint8_t)(value) << PMC_REGSC_ACKISO_SHIFT), PMC_REGSC_ACKISO_SHIFT,  

PMC_REGSC_ACKISO_WIDTH))  

/*@*/  

/*!  

 * @name Register PMC_REGSC, field BGEN[4] (RW)  

 *  

 * BGEN controls whether the bandgap is enabled in lower power modes of  

 * operation (VLPx, LLS, and VLLSx). When on-chip peripherals require the  

bandgap voltage  

 * reference in low power modes of operation, set BGEN to continue to  

enable the  

 * bandgap operation. When the bandgap voltage reference is not needed in  

low  

 * power modes, clear BGEN to avoid excess power consumption.  

 *  

 * Values:
```

```

 * - 0b0 - Bandgap voltage reference is disabled in VLPx , LLS , and
VLLSx modes
 * - 0b1 - Bandgap voltage reference is enabled in VLPx , LLS , and VLLSx
modes
 */
/*@{*/
/*! @brief Read current value of the PMC_REGSC_BGEN field. */
#define PMC_RD_REGSC_BGEN(base) ((PMC_REGSC_REG(base) &
PMC_REGSC_BGEN_MASK) >> PMC_REGSC_BGEN_SHIFT)
#define PMC_BRD_REGSC_BGEN(base) (BME_UBFX8(&PMC_REGSC_REG(base),
PMC_REGSC_BGEN_SHIFT, PMC_REGSC_BGEN_WIDTH))

/*! @brief Set the BGEN field to a new value. */
#define PMC_WR_REGSC_BGEN(base, value) (PMC_RMW_REGSC(base,
(PMC_REGSC_BGEN_MASK | PMC_REGSC_ACKISO_MASK), PMC_REGSC_BGEN(value)))
#define PMC_BWR_REGSC_BGEN(base, value) (BME_BFI8(&PMC_REGSC_REG(base),
((uint8_t)(value) << PMC_REGSC_BGEN_SHIFT), PMC_REGSC_BGEN_SHIFT,
PMC_REGSC_BGEN_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 PORT
 *
 * Pin Control and Interrupts
 *
 * Registers defined in this header file:
 * - PORT_PCR - Pin Control Register n
 * - PORT_GPCLR - Global Pin Control Low Register
 * - PORT_GPCHR - Global Pin Control High Register
 * - PORT_ISFR - Interrupt Status Flag Register
 */
#define PORT_INSTANCE_COUNT (5U) /*!< Number of instances of the PORT
module. */
#define PORTA_IDX (0U) /*!< Instance number for PORTA. */
#define PORTB_IDX (1U) /*!< Instance number for PORTB. */
#define PORTC_IDX (2U) /*!< Instance number for PORTC. */
#define PORTD_IDX (3U) /*!< Instance number for PORTD. */
#define PORTE_IDX (4U) /*!< Instance number for PORTE. */

*****  

* PORT_PCR - Pin Control Register n  

*****  

/*!  

 * @brief PORT_PCR - Pin Control Register n (RW)  

 * Reset value: 0x00000706U  

 * Refer to the Signal Multiplexing and Signal Descriptions chapter for
the

```

```

 * reset value of this device. See the GPIO Configuration section for
details on the
 * available functions for each pin. Do not modify pin configuration
registers
 * associated with pins not available in your selected package. All un-
bonded pins
 * not available in your package will default to DISABLE state for lowest
power
 * consumption.
 */
/*!
 * @name Constants and macros for entire PORT_PCR register
 */
/*@{*/
#define PORT_RD_PCR(base, index) (PORT_PCR_REG(base, index))
#define PORT_WR_PCR(base, index, value) (PORT_PCR_REG(base, index) =
(value))
#define PORT_RMW_PCR(base, index, mask, value) (PORT_WR_PCR(base, index,
(PORT_RD_PCR(base, index) & ~mask)) | (value)))
#define PORT_SET_PCR(base, index, value) (BME_OR32(&PORT_PCR_REG(base,
index), (uint32_t)(value)))
#define PORT_CLR_PCR(base, index, value) (BME_AND32(&PORT_PCR_REG(base,
index), (uint32_t)(~value)))
#define PORT_TOG_PCR(base, index, value) (BME_XOR32(&PORT_PCR_REG(base,
index), (uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual PORT_PCR bitfields
 */

/*!
 * @name Register PORT_PCR, field PS[0] (RW)
 *
 * This bit is read only for pins that do not support a configurable pull
 * resistor direction. Pull configuration is valid in all digital pin
muxing modes.
 *
 * Values:
 * - 0b0 - Internal pulldown resistor is enabled on the corresponding
pin, if
 *   the corresponding Port Pull Enable field is set.
 * - 0b1 - Internal pullup resistor is enabled on the corresponding pin,
if the
 *   corresponding Port Pull Enable field is set.
 */
/*@{*/
/*! @brief Read current value of the PORT_PCR_PS field. */
#define PORT_RD_PCR_PS(base, index) (((PORT_PCR_REG(base, index) &
PORT_PCR_PS_MASK) >> PORT_PCR_PS_SHIFT)
#define PORT_BRD_PCR_PS(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_PS_SHIFT, PORT_PCR_PS_WIDTH))

/*! @brief Set the PS field to a new value. */

```

```

#define PORT_WR_PCR_PS(base, index, value) (PORT_RMW_PCR(base, index,
(PORT_PCR_PS_MASK | PORT_PCR_ISF_MASK), PORT_PCR_PS(value)))
#define PORT_BWR_PCR_PS(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_PS_SHIFT), PORT_PCR_PS_SHIFT, PORT_PCR_PS_WIDTH))
/*@}*/

/*
 * @name Register PORT_PCR, field PE[1] (RW)
 *
 * This bit is read only for pins that do not support a configurable pull
 * resistor. Refer to the Chapter of Signal Multiplexing and Signal
Descriptions for
 * the pins that support a configurable pull resistor. Pull configuration
is valid
 * in all digital pin muxing modes.
 *
 * Values:
 * - 0b0 - Internal pullup or pulldown resistor is not enabled on the
 * corresponding pin.
 * - 0b1 - Internal pullup or pulldown resistor is enabled on the
corresponding
 * pin, if the pin is configured as a digital input.
 */
/*@*/
/*! @brief Read current value of the PORT_PCR_PE field. */
#define PORT_RD_PCR_PE(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_PE_MASK) >> PORT_PCR_PE_SHIFT)
#define PORT_BRD_PCR_PE(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_PE_SHIFT, PORT_PCR_PE_WIDTH))

/*! @brief Set the PE field to a new value. */
#define PORT_WR_PCR_PE(base, index, value) (PORT_RMW_PCR(base, index,
(PORT_PCR_PE_MASK | PORT_PCR_ISF_MASK), PORT_PCR_PE(value)))
#define PORT_BWR_PCR_PE(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_PE_SHIFT), PORT_PCR_PE_SHIFT, PORT_PCR_PE_WIDTH))
/*@*/

/*
 * @name Register PORT_PCR, field SRE[2] (RW)
 *
 * This bit is read only for pins that do not support a configurable slew
rate.
 * Slew rate configuration is valid in all digital pin muxing modes.
 *
 * Values:
 * - 0b0 - Fast slew rate is configured on the corresponding pin, if the
pin is
 * configured as a digital output.
 * - 0b1 - Slow slew rate is configured on the corresponding pin, if the
pin is
 * configured as a digital output.
 */

```

```

/*@{*/
/*! @brief Read current value of the PORT_PCR_SRE field. */
#define PORT_RD_PCR_SRE(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_SRE_MASK) >> PORT_PCR_SRE_SHIFT)
#define PORT_BRD_PCR_SRE(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_SRE_SHIFT, PORT_PCR_SRE_WIDTH))

/*! @brief Set the SRE field to a new value. */
#define PORT_WR_PCR_SRE(base, index, value) (PORT_RMW_PCR(base, index,
(PORT_PCR_SRE_MASK | PORT_PCR_ISF_MASK), PORT_PCR_SRE(value)))
#define PORT_BWR_PCR_SRE(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_SRE_SHIFT), PORT_PCR_SRE_SHIFT, PORT_PCR_SRE_WIDTH))
/*}@*/
```

/\*!

\* @name Register PORT\_PCR, field PFE[4] (RW)

\*

\* This bit is read only for pins that do not support a configurable passive input filter. Passive filter configuration is valid in all digital pin muxing modes.

\*

\* Values:

\* - 0b0 - Passive input filter is disabled on the corresponding pin.

\* - 0b1 - Passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. Refer to the device data sheet for filter characteristics.

\*/

```

/*@{*/
/*! @brief Read current value of the PORT_PCR_PFE field. */
#define PORT_RD_PCR_PFE(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_PFE_MASK) >> PORT_PCR_PFE_SHIFT)
#define PORT_BRD_PCR_PFE(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_PFE_SHIFT, PORT_PCR_PFE_WIDTH))

/*! @brief Set the PFE field to a new value. */
#define PORT_WR_PCR_PFE(base, index, value) (PORT_RMW_PCR(base, index,
(PORT_PCR_PFE_MASK | PORT_PCR_ISF_MASK), PORT_PCR_PFE(value)))
#define PORT_BWR_PCR_PFE(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_PFE_SHIFT), PORT_PCR_PFE_SHIFT, PORT_PCR_PFE_WIDTH))
/*}@*/
```

/\*!

\* @name Register PORT\_PCR, field DSE[6] (RW)

\*

\* This bit is read only for pins that do not support a configurable drive strength. Drive strength configuration is valid in all digital pin muxing modes.

```

/*
 * Values:
 * - 0b0 - Low drive strength is configured on the corresponding pin, if
pin is
 *     configured as a digital output.
 * - 0b1 - High drive strength is configured on the corresponding pin, if
pin is
 *     configured as a digital output.
 */
/*@{*/
/*! @brief Read current value of the PORT_PCR_DSE field. */
#define PORT_RD_PCR_DSE(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_DSE_MASK) >> PORT_PCR_DSE_SHIFT)
#define PORT_BRD_PCR_DSE(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_DSE_SHIFT, PORT_PCR_DSE_WIDTH))

/*! @brief Set the DSE field to a new value. */
#define PORT_WR_PCR_DSE(base, index, value) (PORT_RMW_PCR(base, index,
(PORT_PCR_DSE_MASK | PORT_PCR_ISF_MASK), PORT_PCR_DSE(value)))
#define PORT_BWR_PCR_DSE(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_DSE_SHIFT), PORT_PCR_DSE_SHIFT, PORT_PCR_DSE_WIDTH))
/*}@*/ */

/*!
 * @name Register PORT_PCR, field MUX[10:8] (RW)
 *
 * Not all pins support all pin muxing slots. Unimplemented pin muxing
slots are
 * reserved and may result in configuring the pin for a different pin
muxing
 * slot. The corresponding pin is configured in the following pin muxing
slot as
 * follows:
 *
 * Values:
 * - 0b000 - Pin disabled (analog).
 * - 0b001 - Alternative 1 (GPIO).
 * - 0b010 - Alternative 2 (chip-specific).
 * - 0b011 - Alternative 3 (chip-specific).
 * - 0b100 - Alternative 4 (chip-specific).
 * - 0b101 - Alternative 5 (chip-specific).
 * - 0b110 - Alternative 6 (chip-specific).
 * - 0b111 - Alternative 7 (chip-specific).
 */
/*@{*/
/*! @brief Read current value of the PORT_PCR_MUX field. */
#define PORT_RD_PCR_MUX(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_MUX_MASK) >> PORT_PCR_MUX_SHIFT)
#define PORT_BRD_PCR_MUX(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_MUX_SHIFT, PORT_PCR_MUX_WIDTH))

/*! @brief Set the MUX field to a new value. */

```

```

#define PORT_WR_PCR_MUX(base, index, value) (PORT_RMW_PCR(base, index, (PORT_PCR_MUX_MASK | PORT_PCR_ISF_MASK), PORT_PCR_MUX(value)))
#define PORT_BWR_PCR_MUX(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) << PORT_PCR_MUX_SHIFT), PORT_PCR_MUX_SHIFT, PORT_PCR_MUX_WIDTH))
/*@}*/

/*
 * @name Register PORT_PCR, field IRQC[19:16] (RW)
 *
 * This field is read only for pins that do not support interrupt generation.
 * The pin interrupt configuration is valid in all digital pin muxing modes. The
 * corresponding pin is configured to generate interrupt/DMA request as follows:
 *
 * Values:
 * - 0b0000 - Interrupt/DMA request disabled.
 * - 0b0001 - DMA request on rising edge.
 * - 0b0010 - DMA request on falling edge.
 * - 0b0011 - DMA request on either edge.
 * - 0b1000 - Interrupt when logic zero.
 * - 0b1001 - Interrupt on rising edge.
 * - 0b1010 - Interrupt on falling edge.
 * - 0b1011 - Interrupt on either edge.
 * - 0b1100 - Interrupt when logic one.
 */
/*@*/
/**! @brief Read current value of the PORT_PCR_IRQC field. */
#define PORT_RD_PCR_IRQC(base, index) ((PORT_PCR_REG(base, index) & PORT_PCR_IRQC_MASK) >> PORT_PCR_IRQC_SHIFT)
#define PORT_BRD_PCR_IRQC(base, index) (BME_UBFX32(&PORT_PCR_REG(base, index), PORT_PCR_IRQC_SHIFT, PORT_PCR_IRQC_WIDTH))

/**! @brief Set the IRQC field to a new value. */
#define PORT_WR_PCR_IRQC(base, index, value) (PORT_RMW_PCR(base, index, (PORT_PCR_IRQC_MASK | PORT_PCR_ISF_MASK), PORT_PCR_IRQC(value)))
#define PORT_BWR_PCR_IRQC(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) << PORT_PCR_IRQC_SHIFT), PORT_PCR_IRQC_SHIFT, PORT_PCR_IRQC_WIDTH))
/*@*/

/*
 * @name Register PORT_PCR, field ISF[24] (W1C)
 *
 * This bit is read only for pins that do not support interrupt generation. The
 * pin interrupt configuration is valid in all digital pin muxing modes.
 *
 * Values:
 * - 0b0 - Configured interrupt is not detected.
 * - 0b1 - Configured interrupt is detected. If the pin is configured to

```

```

        *      generate a DMA request, then the corresponding flag will be
        *      cleared automatically
        *      at the completion of the requested DMA transfer. Otherwise, the
flag
        *      remains set until a logic one is written to the flag. If the pin
is configured
        *      for a level sensitive interrupt and the pin remains asserted, then
the
        *      flag is set again immediately after it is cleared.
*/
/*@{*/
/*! @brief Read current value of the PORT_PCR_ISF field. */
#define PORT_RD_PCR_ISF(base, index) ((PORT_PCR_REG(base, index) &
PORT_PCR_ISF_MASK) >> PORT_PCR_ISF_SHIFT)
#define PORT_BRD_PCR_ISF(base, index) (BME_UBFX32(&PORT_PCR_REG(base,
index), PORT_PCR_ISF_SHIFT, PORT_PCR_ISF_WIDTH))

/*! @brief Set the ISF field to a new value. */
#define PORT_WR_PCR_ISF(base, index, value) (PORT_RMW_PCR(base, index,
PORT_PCR_ISF_MASK, PORT_PCR_ISF(value)))
#define PORT_BWR_PCR_ISF(base, index, value)
(BME_BFI32(&PORT_PCR_REG(base, index), ((uint32_t)(value) <<
PORT_PCR_ISF_SHIFT), PORT_PCR_ISF_SHIFT, PORT_PCR_ISF_WIDTH))
/*@}*/

/*****
*****
 * PORT_GPCLR - Global Pin Control Low Register
*****
****/

/*!
 * @brief PORT_GPCLR - Global Pin Control Low Register (WORZ)
 *
 * Reset value: 0x00000000U
 *
 * Only 32-bit writes are supported to this register.
 */
/*!
 * @name Constants and macros for entire PORT_GPCLR register
 */
/*@{*/
#define PORT_RD_GPCLR(base)      (PORT_GPCLR_REG(base))
#define PORT_WR_GPCLR(base, value) (PORT_GPCLR_REG(base) = (value))
#define PORT_RMW_GPCLR(base, mask, value) (PORT_WR_GPCLR(base,
(PORT_RD_GPCLR(base) & ~mask)) | (value)))
/*@}*/

/*
 * Constants & macros for individual PORT_GPCLR bitfields
 */
/*!

```

```

* @name Register PORT_GPCLR, field GPWD[15:0] (WORZ)
*
* Write value that is written to all Pin Control Registers bits [15:0]
that are
* selected by GPWE.
*/
/*@{ */
/*! @brief Set the GPWD field to a new value. */
#define PORT_WR_GPCLR_GPWD(base, value) (PORT_RMW_GPCLR(base,
PORT_GPCLR_GPWD_MASK, PORT_GPCLR_GPWD(value)))
#define PORT_BWR_GPCLR_GPWD(base, value)
(BME_BFI32(&PORT_GPCLR_REG(base), ((uint32_t)(value) <<
PORT_GPCLR_GPWD_SHIFT), PORT_GPCLR_GPWD_SHIFT, PORT_GPCLR_GPWD_WIDTH))
/*@} */

/*!!
* @name Register PORT_GPCLR, field GPWE[31:16] (WORZ)
*
* Selects which Pin Control Registers (15 through 0) bits [15:0] update
with
* the value in GPWD.
*
* Values:
* - 0b0000000000000000 - Corresponding Pin Control Register is not
updated with
*   the value in GPWD.
* - 0b0000000000000001 - Corresponding Pin Control Register is updated
with the
*   value in GPWD.
*/
/*@{ */
/*! @brief Set the GPWE field to a new value. */
#define PORT_WR_GPCLR_GPWE(base, value) (PORT_RMW_GPCLR(base,
PORT_GPCLR_GPWE_MASK, PORT_GPCLR_GPWE(value)))
#define PORT_BWR_GPCLR_GPWE(base, value)
(BME_BFI32(&PORT_GPCLR_REG(base), ((uint32_t)(value) <<
PORT_GPCLR_GPWE_SHIFT), PORT_GPCLR_GPWE_SHIFT, PORT_GPCLR_GPWE_WIDTH))
/*@} */

*****  

*****  

* PORT_GPCHR - Global Pin Control High Register  

*****  

*****  

*/!  

* @brief PORT_GPCHR - Global Pin Control High Register (WORZ)
*  

* Reset value: 0x00000000U
*  

* Only 32-bit writes are supported to this register.
*/
/*!

```

```

 * @name Constants and macros for entire PORT_GPCHR register
 */
/*@{*/
#define PORT_RD_GPCHR(base)      (PORT_GPCHR_REG(base))
#define PORT_WR_GPCHR(base, value) (PORT_GPCHR_REG(base) = (value))
#define PORT_RMW_GPCHR(base, mask, value) (PORT_WR_GPCHR(base,
(PORT_RD_GPCHR(base) & ~mask)) | (value)))
/*}@*/
```

/\*
 \* Constants & macros for individual PORT\_GPCHR bitfields
 \*/

```

/*!
 * @name Register PORT_GPCHR, field GPWD[15:0] (WORZ)
 *
 * Write value that is written to all Pin Control Registers bits [15:0]
that are
 * selected by GPWE.
 */
/*@{*/
/*! @brief Set the GPWD field to a new value. */
#define PORT_WR_GPCHR_GPWD(base, value) (PORT_RMW_GPCHR(base,
PORT_GPCHR_GPWD_MASK, PORT_GPCHR_GPWD(value)))
#define PORT_BWR_GPCHR_GPWD(base, value)
(BME_BFI32(&PORT_GPCHR_REG(base), ((uint32_t)(value) <<
PORT_GPCHR_GPWD_SHIFT), PORT_GPCHR_GPWD_SHIFT, PORT_GPCHR_GPWD_WIDTH))
/*}@*/
```

```

/*!
 * @name Register PORT_GPCHR, field GPWE[31:16] (WORZ)
 *
 * Selects which Pin Control Registers (31 through 16) bits [15:0] update
with
 * the value in GPWD.
 *
 * Values:
 * - 0b0000000000000000 - Corresponding Pin Control Register is not
updated with
 *   the value in GPWD.
 * - 0b0000000000000001 - Corresponding Pin Control Register is updated
with the
 *   value in GPWD.
 */
/*@{*/
/*! @brief Set the GPWE field to a new value. */
#define PORT_WR_GPCHR_GPWE(base, value) (PORT_RMW_GPCHR(base,
PORT_GPCHR_GPWE_MASK, PORT_GPCHR_GPWE(value)))
#define PORT_BWR_GPCHR_GPWE(base, value)
(BME_BFI32(&PORT_GPCHR_REG(base), ((uint32_t)(value) <<
PORT_GPCHR_GPWE_SHIFT), PORT_GPCHR_GPWE_SHIFT, PORT_GPCHR_GPWE_WIDTH))
/*}@*/
```

```

*****
* PORT_ISFR - Interrupt Status Flag Register
*****
**** */

/*!
* @brief PORT_ISFR - Interrupt Status Flag Register (W1C)
*
* Reset value: 0x00000000U
*
* The corresponding bit is read only for pins that do not support
interrupt
* generation. The pin interrupt configuration is valid in all digital
pin muxing
* modes. The Interrupt Status Flag for each pin is also visible in the
* corresponding Pin Control Register, and each flag can be cleared in
either location.
*/
/*!
* @name Constants and macros for entire PORT_ISFR register
*/
/*@{ */
#define PORT_RD_ISFR(base)      (PORT_ISFR_REG(base))
#define PORT_WR_ISFR(base, value) (PORT_ISFR_REG(base) = (value))
#define PORT_RMW_ISFR(base, mask, value) (PORT_WR_ISFR(base,
(PORT_RD_ISFR(base) & ~mask) | (value)))
#define PORT_SET_ISFR(base, value) (BME_OR32(&PORT_ISFR_REG(base),
(uint32_t)(value)))
#define PORT_CLR_ISFR(base, value) (BME_AND32(&PORT_ISFR_REG(base),
(uint32_t)(~(value))))
#define PORT_TOG_ISFR(base, value) (BME_XOR32(&PORT_ISFR_REG(base),
(uint32_t)(value)))
/*@} */

/*
* MKL25Z4 RCM
*
* Reset Control Module
*
* Registers defined in this header file:
* - RCM_SRS0 - System Reset Status Register 0
* - RCM_SRS1 - System Reset Status Register 1
* - RCM_RPFC - Reset Pin Filter Control register
* - RCM_RPFW - Reset Pin Filter Width register
*/
#define RCM_INSTANCE_COUNT (1U) /*!< Number of instances of the RCM
module. */
#define RCM_IDX (0U) /*!< Instance number for RCM. */

*****
*****
```

```

 * RCM_SRS0 - System Reset Status Register 0
 ****
 **** */

/*!
 * @brief RCM_SRS0 - System Reset Status Register 0 (RO)
 *
 * Reset value: 0x82U
 *
 * This register includes read-only status flags to indicate the source
 * of the
 * most recent reset. The reset state of these bits depends on what
 * caused the MCU
 * to reset. The reset value of this register depends on the reset
 * source: POR
 * (including LVD) - 0x82 LVD (without POR) - 0x02 VLLS mode wakeup due
 * to RESET
 * pin assertion - 0x41 VLLS mode wakeup due to other wakeup sources -
 * 0x01 Other
 * reset - a bit is set if its corresponding reset source caused the
 * reset
 */
/*!
 * @name Constants and macros for entire RCM_SRS0 register
 */
/*@{*/
#define RCM_RD_SRS0(base)           (RCM_SRS0_REG(base))
/*}@*/

/*
 * Constants & macros for individual RCM_SRS0 bitfields
 */

/*!
 * @name Register RCM_SRS0, field WAKEUP[0] (RO)
 *
 * Indicates a reset has been caused by an enabled LLWU module wakeup
 * source
 * while the chip was in a low leakage mode. In LLS mode, the RESET pin
 * is the only
 * wakeup source that can cause this reset. Any enabled wakeup source in
 * a VLLSx
 * mode causes a reset. This bit is cleared by any reset except WAKEUP.
 *
 * Values:
 * - 0b0 - Reset not caused by LLWU module wakeup source
 * - 0b1 - Reset caused by LLWU module wakeup source
 */
/*@{*/
/*! @brief Read current value of the RCM_SRS0_WAKEUP field. */
#define RCM_RD_SRS0_WAKEUP(base) ((RCM_SRS0_REG(base) &
RCM_SRS0_WAKEUP_MASK) >> RCM_SRS0_WAKEUP_SHIFT)

```

```

#define RCM_BRD_SRS0_WAKEUP(base) (BME_UBFX8(&RCM_SRS0_REG(base),  

RCM_SRS0_WAKEUP_SHIFT, RCM_SRS0_WAKEUP_WIDTH))  

/*@*/
```

/\*!  
\* @name Register RCM\_SRS0, field LVD[1] (RO)  
\*  
\* If the LVDRE bit is set and the supply drops below the LVD trip  
voltage, an  
\* LVD reset occurs. This bit is also set by POR.  
\*  
\* Values:  
\* - 0b0 - Reset not caused by LVD trip or POR  
\* - 0b1 - Reset caused by LVD trip or POR  
\*/  
/\*@{\*/  
/\*! @brief Read current value of the RCM\_SRS0\_LVD field. \*/  
#define RCM\_RD\_SRS0\_LVD(base) ((RCM\_SRS0\_REG(base) & RCM\_SRS0\_LVD\_MASK)  
>> RCM\_SRS0\_LVD\_SHIFT)  
#define RCM\_BRD\_SRS0\_LVD(base) (BME\_UBFX8(&RCM\_SRS0\_REG(base),  
RCM\_SRS0\_LVD\_SHIFT, RCM\_SRS0\_LVD\_WIDTH))  
/\*@\*/

/\*!  
\* @name Register RCM\_SRS0, field LOC[2] (RO)  
\*  
\* Indicates a reset has been caused by a loss of external clock. The MCG  
clock  
\* monitor must be enabled for a loss of clock to be detected. Refer to  
the  
\* detailed MCG description for information on enabling the clock  
monitor.  
\*  
\* Values:  
\* - 0b0 - Reset not caused by a loss of external clock.  
\* - 0b1 - Reset caused by a loss of external clock.  
\*/  
/\*@{\*/  
/\*! @brief Read current value of the RCM\_SRS0\_LOC field. \*/  
#define RCM\_RD\_SRS0\_LOC(base) ((RCM\_SRS0\_REG(base) & RCM\_SRS0\_LOC\_MASK)  
>> RCM\_SRS0\_LOC\_SHIFT)  
#define RCM\_BRD\_SRS0\_LOC(base) (BME\_UBFX8(&RCM\_SRS0\_REG(base),  
RCM\_SRS0\_LOC\_SHIFT, RCM\_SRS0\_LOC\_WIDTH))  
/\*@\*/

/\*!  
\* @name Register RCM\_SRS0, field LOL[3] (RO)  
\*  
\* Indicates a reset has been caused by a loss of lock in the MCG PLL.  
See the  
\* MCG description for information on the loss-of-clock event.  
\*  
\* Values:  
\* - 0b0 - Reset not caused by a loss of lock in the PLL

```

* - 0b1 - Reset caused by a loss of lock in the PLL
*/
/*@{*/
/*! @brief Read current value of the RCM_SRS0_LOL field. */
#define RCM_RD_SRS0_LOL(base) ((RCM_SRS0_REG(base) & RCM_SRS0_LOL_MASK)
>> RCM_SRS0_LOL_SHIFT)
#define RCM_BRD_SRS0_LOL(base) (BME_UBFX8(&RCM_SRS0_REG(base),
RCM_SRS0_LOL_SHIFT, RCM_SRS0_LOL_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register RCM_SRS0, field WDOG[5] (RO)
 *
 * Indicates a reset has been caused by the watchdog timer Computer
Operating
 * Properly (COP) timing out. This reset source can be blocked by
disabling the COP
 * watchdog: write 00 to the SIM's COPC[COPT] field.
 *
 * Values:
 * - 0b0 - Reset not caused by watchdog timeout
 * - 0b1 - Reset caused by watchdog timeout
 */
/*@{*/
/*! @brief Read current value of the RCM_SRS0_WDOG field. */
#define RCM_RD_SRS0_WDOG(base) ((RCM_SRS0_REG(base) & RCM_SRS0_WDOG_MASK)
>> RCM_SRS0_WDOG_SHIFT)
#define RCM_BRD_SRS0_WDOG(base) (BME_UBFX8(&RCM_SRS0_REG(base),
RCM_SRS0_WDOG_SHIFT, RCM_SRS0_WDOG_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register RCM_SRS0, field PIN[6] (RO)
 *
 * Indicates a reset has been caused by an active-low level on the
external
 * RESET pin.
 *
 * Values:
 * - 0b0 - Reset not caused by external reset pin
 * - 0b1 - Reset caused by external reset pin
 */
/*@{*/
/*! @brief Read current value of the RCM_SRS0_PIN field. */
#define RCM_RD_SRS0_PIN(base) ((RCM_SRS0_REG(base) & RCM_SRS0_PIN_MASK)
>> RCM_SRS0_PIN_SHIFT)
#define RCM_BRD_SRS0_PIN(base) (BME_UBFX8(&RCM_SRS0_REG(base),
RCM_SRS0_PIN_SHIFT, RCM_SRS0_PIN_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register RCM_SRS0, field POR[7] (RO)
 *
```

```

 * Indicates a reset has been caused by the power-on detection logic.
Because
 * the internal supply voltage was ramping up at the time, the low-
voltage reset
 * (LVD) status bit is also set to indicate that the reset occurred while
the
 * internal supply was below the LVD threshold.
*
* Values:
* - 0b0 - Reset not caused by POR
* - 0b1 - Reset caused by POR
*/
/*@{*/
/**! @brief Read current value of the RCM_SRS0_POR field. */
#define RCM_RD_SRS0_POR(base) ((RCM_SRS0_REG(base) & RCM_SRS0_POR_MASK)
>> RCM_SRS0_POR_SHIFT)
#define RCM_BRD_SRS0_POR(base) (BME_UBFX8(&RCM_SRS0_REG(base),
RCM_SRS0_POR_SHIFT, RCM_SRS0_POR_WIDTH))
/*@}*/
*****  

* RCM_SRS1 - System Reset Status Register 1  

*****  

/*!  

* @brief RCM_SRS1 - System Reset Status Register 1 (RO)  

*  

* Reset value: 0x00U  

*  

* This register includes read-only status flags to indicate the source  

of the  

* most recent reset. The reset state of these bits depends on what  

caused the MCU  

* to reset. The reset value of this register depends on the reset  

source: POR  

* (including LVD) - 0x00 LVD (without POR) - 0x00 VLLS mode wakeup -  

0x00 Other  

* reset - a bit is set if its corresponding reset source caused the  

reset
*/
/*!  

* @name Constants and macros for entire RCM_SRS1 register
*/
/*@{*/
#define RCM_RD_SRS1(base) (RCM_SRS1_REG(base))
/*@}*/

/*
* Constants & macros for individual RCM_SRS1 bitfields
*/

```

```

/*!
 * @name Register RCM_SRS1, field LOCKUP[1] (RO)
 *
 * Indicates a reset has been caused by the ARM core indication of a
LOCKUP
 * event.
 *
 * Values:
 * - 0b0 - Reset not caused by core LOCKUP event
 * - 0b1 - Reset caused by core LOCKUP event
 */
/*@{*/
/*! @brief Read current value of the RCM_SRS1_LOCKUP field. */
#define RCM_RD_SRS1_LOCKUP(base) ((RCM_SRS1_REG(base) &
RCM_SRS1_LOCKUP_MASK) >> RCM_SRS1_LOCKUP_SHIFT)
#define RCM_BRD_SRS1_LOCKUP(base) (BME_UBX8(&RCM_SRS1_REG(base),
RCM_SRS1_LOCKUP_SHIFT, RCM_SRS1_LOCKUP_WIDTH))
/*}@*/



/*!
 * @name Register RCM_SRS1, field SW[2] (RO)
 *
 * Indicates a reset has been caused by software setting of SYSRESETREQ
bit in
 * Application Interrupt and Reset Control Register in the ARM core.
 *
 * Values:
 * - 0b0 - Reset not caused by software setting of SYSRESETREQ bit
 * - 0b1 - Reset caused by software setting of SYSRESETREQ bit
 */
/*@{*/
/*! @brief Read current value of the RCM_SRS1_SW field. */
#define RCM_RD_SRS1_SW(base) ((RCM_SRS1_REG(base) & RCM_SRS1_SW_MASK) >>
RCM_SRS1_SW_SHIFT)
#define RCM_BRD_SRS1_SW(base) (BME_UBX8(&RCM_SRS1_REG(base),
RCM_SRS1_SW_SHIFT, RCM_SRS1_SW_WIDTH))
/*}@*/



/*!
 * @name Register RCM_SRS1, field MDM_AP[3] (RO)
 *
 * Indicates a reset has been caused by the host debugger system setting
of the
 * System Reset Request bit in the MDM-AP Control Register.
 *
 * Values:
 * - 0b0 - Reset not caused by host debugger system setting of the System
Reset
 *         Request bit
 * - 0b1 - Reset caused by host debugger system setting of the System
Reset
 *         Request bit
 */
/*@{/*

```

```

/*! @brief Read current value of the RCM_SRS1_MDM_AP field. */
#define RCM_RD_SRS1_MDM_AP(base) ((RCM_SRS1_REG(base) &
RCM_SRS1_MDM_AP_MASK) >> RCM_SRS1_MDM_AP_SHIFT)
#define RCM_BRD_SRS1_MDM_AP(base) (BME_UBFX8(&RCM_SRS1_REG(base),
RCM_SRS1_MDM_AP_SHIFT, RCM_SRS1_MDM_AP_WIDTH))
/*@}*/

/*
 * @name Register RCM_SRS1, field SACKERR[5] (RO)
 *
 * Indicates that after an attempt to enter Stop mode, a reset has been
caused
 * by a failure of one or more peripherals to acknowledge within
approximately one
 * second to enter stop mode.
 *
 * Values:
 * - 0b0 - Reset not caused by peripheral failure to acknowledge attempt
to
 *   enter stop mode
 * - 0b1 - Reset caused by peripheral failure to acknowledge attempt to
enter
 *   stop mode
 */
/*@*/
/*! @brief Read current value of the RCM_SRS1_SACKERR field. */
#define RCM_RD_SRS1_SACKERR(base) ((RCM_SRS1_REG(base) &
RCM_SRS1_SACKERR_MASK) >> RCM_SRS1_SACKERR_SHIFT)
#define RCM_BRD_SRS1_SACKERR(base) (BME_UBFX8(&RCM_SRS1_REG(base),
RCM_SRS1_SACKERR_SHIFT, RCM_SRS1_SACKERR_WIDTH))
/*@*/

*****
*****  

* RCM_RPFC - Reset Pin Filter Control register  

*****  

*****  

*!  

* @brief RCM_RPFC - Reset Pin Filter Control register (RW)  

*  

* Reset value: 0x00U  

*  

* The reset values of bits 2-0 are for Chip POR only. They are
unaffected by
 * other reset types. The bus clock filter is reset when disabled or when
entering
 * stop mode. The LPO filter is reset when disabled .
 */
/*!  

* @name Constants and macros for entire RCM_RPFC register  

*/  

/*@*/

```

```

#define RCM_RD_RPFC(base)          (RCM_RPFC_REG(base))
#define RCM_WR_RPFC(base, value)   (RCM_RPFC_REG(base) = (value))
#define RCM_RMW_RPFC(base, mask, value) (RCM_WR_RPFC(base,
(RCM_RD_RPFC(base) & ~mask)) | (value)))
#define RCM_SET_RPFC(base, value)   (BME_OR8(&RCM_RPFC_REG(base),
(uint8_t)(value)))
#define RCM_CLR_RPFC(base, value)   (BME_AND8(&RCM_RPFC_REG(base),
(uint8_t)(~(value))))
#define RCM_TOG_RPFC(base, value)   (BME_XOR8(&RCM_RPFC_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual RCM_RPFC bitfields
 */

/*!
 * @name Register RCM_RPFC, field RSTFLTSRW[1:0] (RW)
 *
 * Selects how the reset pin filter is enabled in run and wait modes.
 *
 * Values:
 * - 0b00 - All filtering disabled
 * - 0b01 - Bus clock filter enabled for normal operation
 * - 0b10 - LPO clock filter enabled for normal operation
 * - 0b11 - Reserved
 */
/*@{*/
/*! @brief Read current value of the RCM_RPFC_RSTFLTSRW field. */
#define RCM_RD_RPFC_RSTFLTSRW(base) ((RCM_RPFC_REG(base) &
RCM_RPFC_RSTFLTSRW_MASK) >> RCM_RPFC_RSTFLTSRW_SHIFT)
#define RCM_BRD_RPFC_RSTFLTSRW(base) (BME_UBFX8(&RCM_RPFC_REG(base),
RCM_RPFC_RSTFLTSRW_SHIFT, RCM_RPFC_RSTFLTSRW_WIDTH))

/*! @brief Set the RSTFLTSRW field to a new value. */
#define RCM_WR_RPFC_RSTFLTSRW(base, value) (RCM_RMW_RPFC(base,
RCM_RPFC_RSTFLTSRW_MASK, RCM_RPFC_RSTFLTSRW(value)))
#define RCM_BWR_RPFC_RSTFLTSRW(base, value)
(BME_BFI8(&RCM_RPFC_REG(base), ((uint8_t)(value) <<
RCM_RPFC_RSTFLTSRW_SHIFT), RCM_RPFC_RSTFLTSRW_SHIFT,
RCM_RPFC_RSTFLTSRW_WIDTH))
/*@}*/

/*!
 * @name Register RCM_RPFC, field RSTFLTSS[2] (RW)
 *
 * Selects how the reset pin filter is enabled in Stop and VLPS modes ,
and also
 * during LLS and VLLS modes. On exit from VLLS mode, this bit should be
 * reconfigured before clearing ACKISO in the PMC.
 *
 * Values:
 * - 0b0 - All filtering disabled
 * - 0b1 - LPO clock filter enabled

```

```

/*
/*@{*/
/*! @brief Read current value of the RCM_RPFC_RSTFLTSS field. */
#define RCM_RD_RPFC_RSTFLTSS(base) ((RCM_RPFC_REG(base) &
RCM_RPFC_RSTFLTSS_MASK) >> RCM_RPFC_RSTFLTSS_SHIFT)
#define RCM_BRD_RPFC_RSTFLTSS(base) (BME_UBFX8(&RCM_RPFC_REG(base),
RCM_RPFC_RSTFLTSS_SHIFT, RCM_RPFC_RSTFLTSS_WIDTH))

/*! @brief Set the RSTFLTSS field to a new value. */
#define RCM_WR_RPFC_RSTFLTSS(base, value) (RCM_RMW_RPFC(base,
RCM_RPFC_RSTFLTSS_MASK, RCM_RPFC_RSTFLTSS(value)))
#define RCM_BWR_RPFC_RSTFLTSS(base, value) (BME_BFI8(&RCM_RPFC_REG(base),
((uint8_t)(value) << RCM_RPFC_RSTFLTSS_SHIFT), RCM_RPFC_RSTFLTSS_SHIFT,
RCM_RPFC_RSTFLTSS_WIDTH))
/*}@*/
*****  

* RCM_RPFW - Reset Pin Filter Width register  

*****  

/*!  

 * @brief RCM_RPFW - Reset Pin Filter Width register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The reset values of the bits in the RSTFLTSEL field are for Chip POR  

only.  

 * They are unaffected by other reset types.  

 */  

/*!  

 * @name Constants and macros for entire RCM_RPFW register  

 */  

/*@{*/
#define RCM_RD_RPFW(base) (RCM_RPFW_REG(base))  

#define RCM_WR_RPFW(base, value) (RCM_RPFW_REG(base) = (value))  

#define RCM_RMW_RPFW(base, mask, value) (RCM_WR_RPFW(base,  

(RCM_RD_RPFW(base) & ~mask) | (value)))  

#define RCM_SET_RPFW(base, value) (BME_OR8(&RCM_RPFW_REG(base),  

(uint8_t)(value)))  

#define RCM_CLR_RPFW(base, value) (BME_AND8(&RCM_RPFW_REG(base),  

(uint8_t)(~(value))))  

#define RCM_TOG_RPFW(base, value) (BME_XOR8(&RCM_RPFW_REG(base),  

(uint8_t)(value)))  

/*}@*/
*****  

* Constants & macros for individual RCM_RPFW bitfields  

*/  

/*!  

 * @name Register RCM_RPFW, field RSTFLTSEL[4:0] (RW)

```

```

/*
 * Selects the reset pin bus clock filter width.
 *
 * Values:
 * - 0b00000 - Bus clock filter count is 1
 * - 0b00001 - Bus clock filter count is 2
 * - 0b00010 - Bus clock filter count is 3
 * - 0b00011 - Bus clock filter count is 4
 * - 0b00100 - Bus clock filter count is 5
 * - 0b00101 - Bus clock filter count is 6
 * - 0b00110 - Bus clock filter count is 7
 * - 0b00111 - Bus clock filter count is 8
 * - 0b01000 - Bus clock filter count is 9
 * - 0b01001 - Bus clock filter count is 10
 * - 0b01010 - Bus clock filter count is 11
 * - 0b01011 - Bus clock filter count is 12
 * - 0b01100 - Bus clock filter count is 13
 * - 0b01101 - Bus clock filter count is 14
 * - 0b01110 - Bus clock filter count is 15
 * - 0b01111 - Bus clock filter count is 16
 * - 0b10000 - Bus clock filter count is 17
 * - 0b10001 - Bus clock filter count is 18
 * - 0b10010 - Bus clock filter count is 19
 * - 0b10011 - Bus clock filter count is 20
 * - 0b10100 - Bus clock filter count is 21
 * - 0b10101 - Bus clock filter count is 22
 * - 0b10110 - Bus clock filter count is 23
 * - 0b10111 - Bus clock filter count is 24
 * - 0b11000 - Bus clock filter count is 25
 * - 0b11001 - Bus clock filter count is 26
 * - 0b11010 - Bus clock filter count is 27
 * - 0b11011 - Bus clock filter count is 28
 * - 0b11100 - Bus clock filter count is 29
 * - 0b11101 - Bus clock filter count is 30
 * - 0b11110 - Bus clock filter count is 31
 * - 0b11111 - Bus clock filter count is 32
 */
/*@{*/
/*! @brief Read current value of the RCM_RPFW_RSTFLTSEL field. */
#define RCM_RD_RPFW_RSTFLTSEL(base) ((RCM_RPFW_REG(base) &
RCM_RPFW_RSTFLTSEL_MASK) >> RCM_RPFW_RSTFLTSEL_SHIFT)
#define RCM_BRD_RPFW_RSTFLTSEL(base) (BME_UBFX8(&RCM_RPFW_REG(base),
RCM_RPFW_RSTFLTSEL_SHIFT, RCM_RPFW_RSTFLTSEL_WIDTH))

/*! @brief Set the RSTFLTSEL field to a new value. */
#define RCM_WR_RPFW_RSTFLTSEL(base, value) (RCM_RMW_RPFW(base,
RCM_RPFW_RSTFLTSEL_MASK, RCM_RPFW_RSTFLTSEL(value)))
#define RCM_BWR_RPFW_RSTFLTSEL(base, value)
(BME_BFI8(&RCM_RPFW_REG(base), ((uint8_t)(value) <<
RCM_RPFW_RSTFLTSEL_SHIFT), RCM_RPFW_RSTFLTSEL_SHIFT,
RCM_RPFW_RSTFLTSEL_WIDTH))
/*}@*/



/*

```

```

* MKL25Z4 ROM
*
* System ROM
*
* Registers defined in this header file:
* - ROM_ENTRY - Entry
* - ROM_TABLEMARK - End of Table Marker Register
* - ROM_SYSACCESS - System Access Register
* - ROM_PERIPHID4 - Peripheral ID Register
* - ROM_PERIPHID5 - Peripheral ID Register
* - ROM_PERIPHID6 - Peripheral ID Register
* - ROM_PERIPHID7 - Peripheral ID Register
* - ROM_PERIPHID0 - Peripheral ID Register
* - ROM_PERIPHID1 - Peripheral ID Register
* - ROM_PERIPHID2 - Peripheral ID Register
* - ROM_PERIPHID3 - Peripheral ID Register
* - ROM_COMPID - Component ID Register
*/
#define ROM_INSTANCE_COUNT (1U) /*!< Number of instances of the ROM
module. */
#define ROM_IDX (0U) /*!< Instance number for ROM. */

/********************* ROM_ENTRY - Entry ********************
*****
* ROM_ENTRY - Entry
*****
*******/

/*!
* @brief ROM_ENTRY - Entry (RO)
*
* Reset value: 0x00000000U
*
* The System ROM Table begins with "n" relative 32-bit addresses, one
for each
* debug component present in the device and terminating with an all-zero
value
* signaling the end of the table at the "n+1"-th value. See Chip
Configuration
* chapter for the debug components these registers point to. It is
hardwired to
* specific values used during the auto-discovery process by an external
debug
* agent.
*/
/*!
* @name Constants and macros for entire ROM_ENTRY register
*/
/*@{ */
#define ROM_RD_ENTRY(base, index) (ROM_ENTRY_REG(base, index))
/*}@*/
```

```

*****
* ROM_TABLEMARK - End of Table Marker Register
*****
*****/



/*!
* @brief ROM_TABLEMARK - End of Table Marker Register (RO)
*
* Reset value: 0x00000000U
*
* This register indicates end of table marker. It is hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!
* @name Constants and macros for entire ROM_TABLEMARK register
*/
/*@{ */
#define ROM_RD_TABLEMARK(base)      (ROM_TABLEMARK_REG(base))
/*}@*/



*****
* ROM_SYSACCESS - System Access Register
*****
*****/



/*!
* @brief ROM_SYSACCESS - System Access Register (RO)
*
* Reset value: 0x00000001U
*
* This register indicates system access. It is hardwired to specific
values
* used during the auto-discovery process by an external debug agent.
*/
/*!
* @name Constants and macros for entire ROM_SYSACCESS register
*/
/*@{ */
#define ROM_RD_SYSACCESS(base)     (ROM_SYSACCESS_REG(base))
/*}@*/



*****
* ROM_PERIPHID4 - Peripheral ID Register
*****
*****/

```

```

/*!
 * @brief ROM_PERIPHID4 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!!
 * @name Constants and macros for entire ROM_PERIPHID4 register
 */
/*@{*/
#define ROM_RD_PERIPHID4(base)    (ROM_PERIPHID4_REG(base))
/*}@*/



*****  

***  

 * ROM_PERIPHID5 - Peripheral ID Register  

*****  

***/  




/*!
 * @brief ROM_PERIPHID5 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
 * specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!!
 * @name Constants and macros for entire ROM_PERIPHID5 register
 */
/*@{*/
#define ROM_RD_PERIPHID5(base)    (ROM_PERIPHID5_REG(base))
/*}@*/



*****  

***  

 * ROM_PERIPHID6 - Peripheral ID Register  

*****  

***/  




/*!
 * @brief ROM_PERIPHID6 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 */

```

```

 * These registers indicate the peripheral IDs. They are hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire ROM_PERIPHID6 register
 */
/*@{*/
#define ROM_RD_PERIPHID6(base)      (ROM_PERIPHID6_REG(base))
/*}@*/



/********************* ROM_PERIPHID7 - Peripheral ID Register *********************


/*
 * @brief ROM_PERIPHID7 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire ROM_PERIPHID7 register
 */
/*@{*/
#define ROM_RD_PERIPHID7(base)      (ROM_PERIPHID7_REG(base))
/*}@*/



/********************* ROM_PERIPHID0 - Peripheral ID Register *********************


/*
 * @brief ROM_PERIPHID0 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
```

```

 * @name Constants and macros for entire ROM_PERIPHID0 register
 */
/*@{ */
#define ROM_RD_PERIPHID0(base)      (ROM_PERIPHID0_REG(base))
/*}@*/



/********************* ****
 * ROM_PERIPHID1 - Peripheral ID Register
 ****
 * @brief ROM_PERIPHID1 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire ROM_PERIPHID1 register
 */
/*@{ */
#define ROM_RD_PERIPHID1(base)      (ROM_PERIPHID1_REG(base))
/*}@*/



/********************* ****
 * ROM_PERIPHID2 - Peripheral ID Register
 ****
 * @brief ROM_PERIPHID2 - Peripheral ID Register (RO)
 *
 * Reset value: 0x00000000U
 *
 * These registers indicate the peripheral IDs. They are hardwired to
specific
 * values used during the auto-discovery process by an external debug
agent.
 */
/*!
 * @name Constants and macros for entire ROM_PERIPHID2 register
 */
/*@{ */
#define ROM_RD_PERIPHID2(base)      (ROM_PERIPHID2_REG(base))
/*}@*/

```

```

*****
* ROM_PERIPHID3 - Peripheral ID Register
*****
**** */

/*!
* @brief ROM_PERIPHID3 - Peripheral ID Register (RO)
*
* Reset value: 0x00000000U
*
* These registers indicate the peripheral IDs. They are hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!
* @name Constants and macros for entire ROM_PERIPHID3 register
*/
/*@{*/
#define ROM_RD_PERIPHID3(base)      (ROM_PERIPHID3_REG(base))
/*}@*/



*****
* ROM_COMPID - Component ID Register
*****
**** */

/*!
* @brief ROM_COMPID - Component ID Register (RO)
*
* Reset value: 0x00000000U
*
* These registers indicate the component IDs. They are hardwired to
specific
* values used during the auto-discovery process by an external debug
agent.
*/
/*!
* @name Constants and macros for entire ROM_COMPID register
*/
/*@{*/
#define ROM_RD_COMPID(base, index)  (ROM_COMPID_REG(base, index))
/*}@*/



/*
* MKL25Z4 RTC
*
* Secure Real Time Clock
*
* Registers defined in this header file:

```

```

* - RTC_TSR - RTC Time Seconds Register
* - RTC_TPR - RTC Time Prescaler Register
* - RTC_TAR - RTC Time Alarm Register
* - RTC_TCR - RTC Time Compensation Register
* - RTC_CR - RTC Control Register
* - RTC_SR - RTC Status Register
* - RTC_LR - RTC Lock Register
* - RTC_IER - RTC Interrupt Enable Register
*/
#define RTC_INSTANCE_COUNT (1U) /*!< Number of instances of the RTC
module. */
#define RTC_IDX (0U) /*!< Instance number for RTC. */

/********************* RTC_TSR - RTC Time Seconds Register *****/
**** */
* RTC_TSR - RTC Time Seconds Register
*****
/*!
* @brief RTC_TSR - RTC Time Seconds Register (RW)
*
* Reset value: 0x00000000U
*/
/*!
* @name Constants and macros for entire RTC_TSR register
*/
/*@{ */
#define RTC_RD_TSR(base)          (RTC_TSR_REG(base))
#define RTC_WR_TSR(base, value)   (RTC_TSR_REG(base) = (value))
#define RTC_RMW_TSR(base, mask, value) (RTC_WR_TSR(base,
(RTC_RD_TSR(base) & ~mask) | (value)))
#define RTC_SET_TSR(base, value)  (BME_OR32(&RTC_TSR_REG(base),
(uint32_t)(value)))
#define RTC_CLR_TSR(base, value)  (BME_AND32(&RTC_TSR_REG(base),
(uint32_t)(~(value))))
#define RTC_TOG_TSR(base, value)  (BME_XOR32(&RTC_TSR_REG(base),
(uint32_t)(value)))
/*@} */

/********************* RTC_TPR - RTC Time Prescaler Register *****/
**** */
* RTC_TPR - RTC Time Prescaler Register
*****
/*!
* @brief RTC_TPR - RTC Time Prescaler Register (RW)
*
* Reset value: 0x00000000U
*/

```

```

/*!
 * @name Constants and macros for entire RTC_TPR register
 */
/*@{*/
#define RTC_RD_TPR(base)          (RTC_TPR_REG(base))
#define RTC_WR_TPR(base, value)   (RTC_TPR_REG(base) = (value))
#define RTC_RMW_TPR(base, mask, value) (RTC_WR_TPR(base,
(RTC_RD_TPR(base) & ~mask) | (value)))
#define RTC_SET_TPR(base, value)  (BME_OR32(&RTC_TPR_REG(base),
(uint32_t)(value)))
#define RTC_CLR_TPR(base, value)  (BME_AND32(&RTC_TPR_REG(base),
(uint32_t)(~(value))))
#define RTC_TOG_TPR(base, value)  (BME_XOR32(&RTC_TPR_REG(base),
(uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual RTC_TPR bitfields
 */

/*!
 * @name Register RTC_TPR, field TPR[15:0] (RW)
 *
 * When the time counter is enabled, the TPR is read only and increments
 * every
 * 32.768 kHz clock cycle. The time counter will read as zero when
 * SR[TOF] or
 * SR[TIF] are set. When the time counter is disabled, the TPR can be
 * read or
 * written. The TSR[TSR] increments when bit 14 of the TPR transitions
 * from a logic one
 * to a logic zero.
 */
/*@{*/
/*! @brief Read current value of the RTC_TPR_TPR field. */
#define RTC_RD_TPR_TPR(base) ((RTC_TPR_REG(base) & RTC_TPR_TPR_MASK) >>
RTC_TPR_TPR_SHIFT)
#define RTC_BRD_TPR_TPR(base) (BME_UBFX32(&RTC_TPR_REG(base),
RTC_TPR_TPR_SHIFT, RTC_TPR_TPR_WIDTH))

/*! @brief Set the TPR field to a new value. */
#define RTC_WR_TPR_TPR(base, value) (RTC_RMW_TPR(base, RTC_TPR_TPR_MASK,
RTC_TPR_TPR(value)))
#define RTC_BWR_TPR_TPR(base, value) (BME_BFI32(&RTC_TPR_REG(base),
((uint32_t)(value) << RTC_TPR_TPR_SHIFT), RTC_TPR_TPR_SHIFT,
RTC_TPR_TPR_WIDTH))
/*}@*/ */

*****  

* RTC_TAR - RTC Time Alarm Register  

*****  

*****/

```

```

/*!
 * @brief RTC_TAR - RTC Time Alarm Register (RW)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire RTC_TAR register
 */
/*@{*/
#define RTC_RD_TAR(base)          (RTC_TAR_REG(base))
#define RTC_WR_TAR(base, value)   (RTC_TAR_REG(base) = (value))
#define RTC_RMW_TAR(base, mask, value) (RTC_WR_TAR(base,
(RTC_RD_TAR(base) & ~mask) | (value)))
#define RTC_SET_TAR(base, value)  (BME_OR32(&RTC_TAR_REG(base),
(uint32_t)(value)))
#define RTC_CLR_TAR(base, value)  (BME_AND32(&RTC_TAR_REG(base),
(uint32_t)(~(value))))
#define RTC_TOG_TAR(base, value)  (BME_XOR32(&RTC_TAR_REG(base),
(uint32_t)(value)))
/*@}*/
*****  

* RTC_TCR - RTC Time Compensation Register  

*****  

/*!
 * @brief RTC_TCR - RTC Time Compensation Register (RW)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire RTC_TCR register
 */
/*@{*/
#define RTC_RD_TCR(base)          (RTC_TCR_REG(base))
#define RTC_WR_TCR(base, value)   (RTC_TCR_REG(base) = (value))
#define RTC_RMW_TCR(base, mask, value) (RTC_WR_TCR(base,
(RTC_RD_TCR(base) & ~mask) | (value)))
#define RTC_SET_TCR(base, value)  (BME_OR32(&RTC_TCR_REG(base),
(uint32_t)(value)))
#define RTC_CLR_TCR(base, value)  (BME_AND32(&RTC_TCR_REG(base),
(uint32_t)(~(value))))
#define RTC_TOG_TCR(base, value)  (BME_XOR32(&RTC_TCR_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual RTC_TCR bitfields
*/

```

```

/*!
 * @name Register RTC_TCR, field TCR[7:0] (RW)
 *
 * Configures the number of 32.768 kHz clock cycles in each second. This
 * register is double buffered and writes do not take affect until the
end of the
 * current compensation interval.
 *
 * Values:
 * - 0b10000000 - Time Prescaler Register overflows every 32896 clock
cycles.
 * - 0b11111111 - Time Prescaler Register overflows every 32769 clock
cycles.
 * - 0b00000000 - Time Prescaler Register overflows every 32768 clock
cycles.
 * - 0b00000001 - Time Prescaler Register overflows every 32767 clock
cycles.
 * - 0b01111111 - Time Prescaler Register overflows every 32641 clock
cycles.
 */
/*@{*/
/*! @brief Read current value of the RTC_TCR_TCR field. */
#define RTC_RD_TCR_TCR(base) ((RTC_TCR_REG(base) & RTC_TCR_TCR_MASK) >>
RTC_TCR_TCR_SHIFT)
#define RTC_BRD_TCR_TCR(base) (BME_UBFX32(&RTC_TCR_REG(base),
RTC_TCR_TCR_SHIFT, RTC_TCR_TCR_WIDTH))

/*! @brief Set the TCR field to a new value. */
#define RTC_WR_TCR_TCR(base, value) (RTC_RMW_TCR(base, RTC_TCR_TCR_MASK,
RTC_TCR_TCR(value)))
#define RTC_BWR_TCR_TCR(base, value) (BME_BFI32(&RTC_TCR_REG(base),
((uint32_t)(value) << RTC_TCR_TCR_SHIFT), RTC_TCR_TCR_SHIFT,
RTC_TCR_TCR_WIDTH))
/*}@*/
```

```

/*!
 * @name Register RTC_TCR, field CIR[15:8] (RW)
 *
 * Configures the compensation interval in seconds from 1 to 256 to
control how
 * frequently the TCR should adjust the number of 32.768 kHz cycles in
each
 * second. The value written should be one less than the number of
seconds. For
 * example, write zero to configure for a compensation interval of one
second. This
 * register is double buffered and writes do not take affect until the
end of the
 * current compensation interval.
 */
/*@{*/
/*! @brief Read current value of the RTC_TCR_CIR field. */
#define RTC_RD_TCR_CIR(base) ((RTC_TCR_REG(base) & RTC_TCR_CIR_MASK) >>
RTC_TCR_CIR_SHIFT)
```

```

#define RTC_BRD_TCR_CIR(base)  (BME_UBFX32(&RTC_TCR_REG(base),  

RTC_TCR_CIR_SHIFT, RTC_TCR_CIR_WIDTH))

/*! @brief Set the CIR field to a new value. */  

#define RTC_WR_TCR_CIR(base, value)  (RTC_RMW_TCR(base, RTC_TCR_CIR_MASK,  

RTC_TCR_CIR(value)))  

#define RTC_BWR_TCR_CIR(base, value)  (BME_BFI32(&RTC_TCR_REG(base),  

(uint32_t)(value) << RTC_TCR_CIR_SHIFT), RTC_TCR_CIR_SHIFT,  

RTC_TCR_CIR_WIDTH))  

/*@}*/

/*!  

 * @name Register RTC_TCR, field TCV[23:16] (RO)  

 *  

 * Current value used by the compensation logic for the present second  

interval.  

 * Updated once a second if the CIC equals 0 with the contents of the TCR  

field.  

 * If the CIC does not equal zero then it is loaded with zero  

(compensation is  

 * not enabled for that second increment).  

 */  

/*@{*/  

/*! @brief Read current value of the RTC_TCR_TCV field. */  

#define RTC_RD_TCR_TCV(base)  ((RTC_TCR_REG(base) & RTC_TCR_TCV_MASK) >>  

RTC_TCR_TCV_SHIFT)  

#define RTC_BRD_TCR_TCV(base)  (BME_UBFX32(&RTC_TCR_REG(base),  

RTC_TCR_TCV_SHIFT, RTC_TCR_TCV_WIDTH))  

/*@}*/

/*!  

 * @name Register RTC_TCR, field CIC[31:24] (RO)  

 *  

 * Current value of the compensation interval counter. If the  

compensation  

 * interval counter equals zero then it is loaded with the contents of  

the CIR. If the  

 * CIC does not equal zero then it is decremented once a second.  

 */  

/*@{*/  

/*! @brief Read current value of the RTC_TCR_CIC field. */  

#define RTC_RD_TCR_CIC(base)  ((RTC_TCR_REG(base) & RTC_TCR_CIC_MASK) >>  

RTC_TCR_CIC_SHIFT)  

#define RTC_BRD_TCR_CIC(base)  (BME_UBFX32(&RTC_TCR_REG(base),  

RTC_TCR_CIC_SHIFT, RTC_TCR_CIC_WIDTH))  

/*@}*/

*****  

*****  

 * RTC_CR - RTC Control Register  

*****  

*****  

*****/

```

```

/*!
 * @brief RTC_CR - RTC Control Register (RW)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire RTC_CR register
 */
/*@{*/
#define RTC_RD_CR(base)          (RTC_CR_REG(base))
#define RTC_WR_CR(base, value)    (RTC_CR_REG(base) = (value))
#define RTC_RMW_CR(base, mask, value) (RTC_WR_CR(base, (RTC_RD_CR(base) & ~mask) | (value)))
#define RTC_SET_CR(base, value)   (BME_OR32(&RTC_CR_REG(base), (uint32_t)(value)))
#define RTC_CLR_CR(base, value)   (BME_AND32(&RTC_CR_REG(base), (uint32_t)(~(value))))
#define RTC_TOG_CR(base, value)   (BME_XOR32(&RTC_CR_REG(base), (uint32_t)(value)))
/*}@*/
```

```

/*
 * Constants & macros for individual RTC_CR bitfields
 */

/*!
 * @name Register RTC_CR, field SWR[0] (RW)
 *
 * Values:
 * - 0b0 - No effect.
 * - 0b1 - Resets all RTC registers except for the SWR bit . The SWR bit
is
 *      cleared by POR and by software explicitly clearing it.
 */
/*@{*/
/*! @brief Read current value of the RTC_CR_SWR field. */
#define RTC_RD_CR_SWR(base) ((RTC_CR_REG(base) & RTC_CR_SWR_MASK) >>
RTC_CR_SWR_SHIFT)
#define RTC_BRD_CR_SWR(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_SWR_SHIFT, RTC_CR_SWR_WIDTH))

/*! @brief Set the SWR field to a new value. */
#define RTC_WR_CR_SWR(base, value) (RTC_RMW_CR(base, RTC_CR_SWR_MASK,
RTC_CR_SWR(value)))
#define RTC_BWR_CR_SWR(base, value) (BME_BFI32(&RTC_CR_REG(base),
((uint32_t)(value) << RTC_CR_SWR_SHIFT), RTC_CR_SWR_SHIFT,
RTC_CR_SWR_WIDTH))
/*}@*/
```

```

/*!
 * @name Register RTC_CR, field WPE[1] (RW)
 *
 * The wakeup pin is optional and not available on all devices.
 */

```

```

* Values:
* - 0b0 - Wakeup pin is disabled.
* - 0b1 - Wakeup pin is enabled and wakeup pin asserts if the RTC
interrupt
*      asserts or the wakeup pin is turned on.
*/
/*@{ */
/*! @brief Read current value of the RTC_CR_WPE field. */
#define RTC_RD_CR_WPE(base) ((RTC_CR_REG(base) & RTC_CR_WPE_MASK) >>
RTC_CR_WPE_SHIFT)
#define RTC_BRD_CR_WPE(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_WPE_SHIFT, RTC_CR_WPE_WIDTH))

/*! @brief Set the WPE field to a new value. */
#define RTC_WR_CR_WPE(base, value) (RTC_RMW_CR(base, RTC_CR_WPE_MASK,
RTC_CR_WPE(value)))
#define RTC_BWR_CR_WPE(base, value) (BME_BFI32(&RTC_CR_REG(base),
((uint32_t)(value) << RTC_CR_WPE_SHIFT), RTC_CR_WPE_SHIFT,
RTC_CR_WPE_WIDTH))
/*@} */

/*!
* @name Register RTC_CR, field SUP[2] (RW)
*
* Values:
* - 0b0 - Non-supervisor mode write accesses are not supported and
generate a
*      bus error.
* - 0b1 - Non-supervisor mode write accesses are supported.
*/
/*@{ */
/*! @brief Read current value of the RTC_CR_SUP field. */
#define RTC_RD_CR_SUP(base) ((RTC_CR_REG(base) & RTC_CR_SUP_MASK) >>
RTC_CR_SUP_SHIFT)
#define RTC_BRD_CR_SUP(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_SUP_SHIFT, RTC_CR_SUP_WIDTH))

/*! @brief Set the SUP field to a new value. */
#define RTC_WR_CR_SUP(base, value) (RTC_RMW_CR(base, RTC_CR_SUP_MASK,
RTC_CR_SUP(value)))
#define RTC_BWR_CR_SUP(base, value) (BME_BFI32(&RTC_CR_REG(base),
((uint32_t)(value) << RTC_CR_SUP_SHIFT), RTC_CR_SUP_SHIFT,
RTC_CR_SUP_WIDTH))
/*@} */

/*!
* @name Register RTC_CR, field UM[3] (RW)
*
* Allows SR[TCE] to be written even when the Status Register is locked.
When
* set, the SR[TCE] can always be written if the SR[TIF] or SR[TOF] are
set or if
* the SR[TCE] is clear.
*

```

```

* Values:
* - 0b0 - Registers cannot be written when locked.
* - 0b1 - Registers can be written when locked under limited conditions.
*/
/*@{*/
/*! @brief Read current value of the RTC_CR UM field. */
#define RTC_RD_CR UM(base) ((RTC_CR_REG(base) & RTC_CR UM_MASK) >>
RTC_CR UM SHIFT)
#define RTC_BRD_CR UM(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR UM SHIFT, RTC_CR UM WIDTH))

/*! @brief Set the UM field to a new value. */
#define RTC_WR_CR UM(base, value) (RTC_RMW_CR(base, RTC_CR UM MASK,
RTC_CR UM(value)))
#define RTC_BWR_CR UM(base, value) (BME_BFI32(&RTC_CR_REG(base),
(uint32_t)(value) << RTC_CR UM SHIFT), RTC_CR UM SHIFT,
RTC_CR UM WIDTH))
/*}@*/ */

/*!
* @name Register RTC_CR, field OSCE[8] (RW)
*
* Values:
* - 0b0 - 32.768 kHz oscillator is disabled.
* - 0b1 - 32.768 kHz oscillator is enabled. After setting this bit, wait
the
*     oscillator startup time before enabling the time counter to allow
the 32.768
*     kHz clock time to stabilize.
*/
/*@{*/
/*! @brief Read current value of the RTC_CR_OSCE field. */
#define RTC_RD_CR OSCE(base) ((RTC_CR_REG(base) & RTC_CR_OSCE_MASK) >>
RTC_CR_OSCE SHIFT)
#define RTC_BRD_CR OSCE(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_OSCE SHIFT, RTC_CR_OSCE WIDTH))

/*! @brief Set the OSCE field to a new value. */
#define RTC_WR_CR OSCE(base, value) (RTC_RMW_CR(base, RTC_CR_OSCE_MASK,
RTC_CR_OSCE(value)))
#define RTC_BWR_CR OSCE(base, value) (BME_BFI32(&RTC_CR_REG(base),
(uint32_t)(value) << RTC_CR_OSCE SHIFT), RTC_CR_OSCE SHIFT,
RTC_CR_OSCE WIDTH))
/*}@*/ */

/*!
* @name Register RTC_CR, field CLKO[9] (RW)
*
* Values:
* - 0b0 - The 32 kHz clock is output to other peripherals.
* - 0b1 - The 32 kHz clock is not output to other peripherals.
*/
/*@{*/
/*! @brief Read current value of the RTC_CR_CLKO field. */

```

```

#define RTC_RD_CR_CLKO(base) ((RTC_CR_REG(base) & RTC_CR_CLKO_MASK) >>
RTC_CR_CLKO_SHIFT)
#define RTC_BRD_CR_CLKO(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_CLKO_SHIFT, RTC_CR_CLKO_WIDTH))

/*! @brief Set the CLKO field to a new value. */
#define RTC_WR_CR_CLKO(base, value) (RTC_RMW_CR(base, RTC_CR_CLKO_MASK,
RTC_CR_CLKO(value)))
#define RTC_BWR_CR_CLKO(base, value) (BME_BFI32(&RTC_CR_REG(base),
((uint32_t)(value) << RTC_CR_CLKO_SHIFT), RTC_CR_CLKO_SHIFT,
RTC_CR_CLKO_WIDTH))
/*@}*/

/*!
* @name Register RTC_CR, field SC16P[10] (RW)
*
* Values:
* - 0b0 - Disable the load.
* - 0b1 - Enable the additional load.
*/
/*@*/
/*! @brief Read current value of the RTC_CR_SC16P field. */
#define RTC_RD_CR_SC16P(base) ((RTC_CR_REG(base) & RTC_CR_SC16P_MASK) >>
RTC_CR_SC16P_SHIFT)
#define RTC_BRD_CR_SC16P(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_SC16P_SHIFT, RTC_CR_SC16P_WIDTH))

/*! @brief Set the SC16P field to a new value. */
#define RTC_WR_CR_SC16P(base, value) (RTC_RMW_CR(base, RTC_CR_SC16P_MASK,
RTC_CR_SC16P(value)))
#define RTC_BWR_CR_SC16P(base, value) (BME_BFI32(&RTC_CR_REG(base),
((uint32_t)(value) << RTC_CR_SC16P_SHIFT), RTC_CR_SC16P_SHIFT,
RTC_CR_SC16P_WIDTH))
/*@*/

/*!
* @name Register RTC_CR, field SC8P[11] (RW)
*
* Values:
* - 0b0 - Disable the load.
* - 0b1 - Enable the additional load.
*/
/*@*/
/*! @brief Read current value of the RTC_CR_SC8P field. */
#define RTC_RD_CR_SC8P(base) ((RTC_CR_REG(base) & RTC_CR_SC8P_MASK) >>
RTC_CR_SC8P_SHIFT)
#define RTC_BRD_CR_SC8P(base) (BME_UBFX32(&RTC_CR_REG(base),
RTC_CR_SC8P_SHIFT, RTC_CR_SC8P_WIDTH))

/*! @brief Set the SC8P field to a new value. */
#define RTC_WR_CR_SC8P(base, value) (RTC_RMW_CR(base, RTC_CR_SC8P_MASK,
RTC_CR_SC8P(value)))

```

```

#define RTC_BWR_CR_SC8P(base, value) (BME_BFI32(&RTC_CR_REG(base),  

((uint32_t)(value) << RTC_CR_SC8P_SHIFT), RTC_CR_SC8P_SHIFT,  

RTC_CR_SC8P_WIDTH))  

/*@}*/

/*!  

 * @name Register RTC_CR, field SC4P[12] (RW)  

 *  

 * Values:  

 * - 0b0 - Disable the load.  

 * - 0b1 - Enable the additional load.  

 */  

/*@{/**/  

/*! @brief Read current value of the RTC_CR_SC4P field. */  

#define RTC_RD_CR_SC4P(base) ((RTC_CR_REG(base) & RTC_CR_SC4P_MASK) >>  

RTC_CR_SC4P_SHIFT)  

#define RTC_BRD_CR_SC4P(base) (BME_UBFX32(&RTC_CR_REG(base),  

RTC_CR_SC4P_SHIFT, RTC_CR_SC4P_WIDTH))

/*! @brief Set the SC4P field to a new value. */  

#define RTC_WR_CR_SC4P(base, value) (RTC_RMW_CR(base, RTC_CR_SC4P_MASK,  

RTC_CR_SC4P(value)))  

#define RTC_BWR_CR_SC4P(base, value) (BME_BFI32(&RTC_CR_REG(base),  

((uint32_t)(value) << RTC_CR_SC4P_SHIFT), RTC_CR_SC4P_SHIFT,  

RTC_CR_SC4P_WIDTH))  

/*@*/}/

/*!  

 * @name Register RTC_CR, field SC2P[13] (RW)  

 *  

 * Values:  

 * - 0b0 - Disable the load.  

 * - 0b1 - Enable the additional load.  

 */  

/*@{/**/  

/*! @brief Read current value of the RTC_CR_SC2P field. */  

#define RTC_RD_CR_SC2P(base) ((RTC_CR_REG(base) & RTC_CR_SC2P_MASK) >>  

RTC_CR_SC2P_SHIFT)  

#define RTC_BRD_CR_SC2P(base) (BME_UBFX32(&RTC_CR_REG(base),  

RTC_CR_SC2P_SHIFT, RTC_CR_SC2P_WIDTH))

/*! @brief Set the SC2P field to a new value. */  

#define RTC_WR_CR_SC2P(base, value) (RTC_RMW_CR(base, RTC_CR_SC2P_MASK,  

RTC_CR_SC2P(value)))  

#define RTC_BWR_CR_SC2P(base, value) (BME_BFI32(&RTC_CR_REG(base),  

((uint32_t)(value) << RTC_CR_SC2P_SHIFT), RTC_CR_SC2P_SHIFT,  

RTC_CR_SC2P_WIDTH))  

/*@*/}/

*****  

* RTC_SR - RTC Status Register

```

```
*****
 ****/
/*!
 * @brief RTC_SR - RTC Status Register (RW)
 *
 * Reset value: 0x00000001U
 */
/*!!
 * @name Constants and macros for entire RTC_SR register
 */
/*@{*/
#define RTC_RD_SR(base)          (RTC_SR_REG(base))
#define RTC_WR_SR(base, value)    (RTC_SR_REG(base) = (value))
#define RTC_RMW_SR(base, mask, value) (RTC_WR_SR(base, (RTC_RD_SR(base) & ~mask) | (value)))
#define RTC_SET_SR(base, value)   (BME_OR32(&RTC_SR_REG(base), (uint32_t)(value)))
#define RTC_CLR_SR(base, value)   (BME_AND32(&RTC_SR_REG(base), (uint32_t)(~(value))))
#define RTC_TOG_SR(base, value)   (BME_XOR32(&RTC_SR_REG(base), (uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual RTC_SR bitfields
 */

/*!!
 * @name Register RTC_SR, field TIF[0] (RO)
 *
 * The time invalid flag is set on POR or software reset. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.
 *
 * Values:
 * - 0b0 - Time is valid.
 * - 0b1 - Time is invalid and time counter is read as zero.
 */
/*@{*/
/*! @brief Read current value of the RTC_SR_TIF field. */
#define RTC_RD_SR_TIF(base) ((RTC_SR_REG(base) & RTC_SR_TIF_MASK) >> RTC_SR_TIF_SHIFT)
#define RTC_BRD_SR_TIF(base) (BME_UBFX32(&RTC_SR_REG(base), RTC_SR_TIF_SHIFT, RTC_SR_TIF_WIDTH))
/*@}*/

/*!!
 * @name Register RTC_SR, field TOF[1] (RO)
 */
```

```

    * Time overflow flag is set when the time counter is enabled and
    overflows. The
    * TSR and TPR do not increment and read as zero when this bit is set.
This bit
    * is cleared by writing the TSR register when the time counter is
disabled.
    *
    * Values:
    * - 0b0 - Time overflow has not occurred.
    * - 0b1 - Time overflow has occurred and time counter is read as zero.
    */
/*@{*/
/*! @brief Read current value of the RTC_SR_TOF field. */
#define RTC_RD_SR_TOF(base) ((RTC_SR_REG(base) & RTC_SR_TOF_MASK) >>
RTC_SR_TOF_SHIFT)
#define RTC_BRD_SR_TOF(base) (BME_UBX32(&RTC_SR_REG(base)),
RTC_SR_TOF_SHIFT, RTC_SR_TOF_WIDTH)
/*}@*/ */

/*!
    * @name Register RTC_SR, field TAF[2] (RO)
    *
    * Time alarm flag is set when the TAR[TAR] equals the TSR[TSR] and the
    TSR[TSR]
    * increments. This bit is cleared by writing the TAR register.
    *
    * Values:
    * - 0b0 - Time alarm has not occurred.
    * - 0b1 - Time alarm has occurred.
    */
/*@{*/
/*! @brief Read current value of the RTC_SR_TAF field. */
#define RTC_RD_SR_TAF(base) ((RTC_SR_REG(base) & RTC_SR_TAF_MASK) >>
RTC_SR_TAF_SHIFT)
#define RTC_BRD_SR_TAF(base) (BME_UBX32(&RTC_SR_REG(base)),
RTC_SR_TAF_SHIFT, RTC_SR_TAF_WIDTH)
/*}@*/ */

/*!
    * @name Register RTC_SR, field TCE[4] (RW)
    *
    * When time counter is disabled the TSR register and TPR register are
    * writeable, but do not increment. When time counter is enabled the TSR
register and TPR
    * register are not writeable, but increment.
    *
    * Values:
    * - 0b0 - Time counter is disabled.
    * - 0b1 - Time counter is enabled.
    */
/*@{*/
/*! @brief Read current value of the RTC_SR_TCE field. */
#define RTC_RD_SR_TCE(base) ((RTC_SR_REG(base) & RTC_SR_TCE_MASK) >>
RTC_SR_TCE_SHIFT)

```

```

#define RTC_BRD_SR_TCE(base)  (BME_UBFX32(&RTC_SR_REG(base),  

RTC_SR_TCE_SHIFT, RTC_SR_TCE_WIDTH))

/*! @brief Set the TCE field to a new value. */  

#define RTC_WR_SR_TCE(base, value)  (RTC_RMW_SR(base, RTC_SR_TCE_MASK,  

RTC_SR_TCE(value)))  

#define RTC_BWR_SR_TCE(base, value)  (BME_BFI32(&RTC_SR_REG(base),  

(uint32_t)(value) << RTC_SR_TCE_SHIFT), RTC_SR_TCE_SHIFT,  

RTC_SR_TCE_WIDTH)  

/*@}*/

/*********************  

*****  

 * RTC_LR - RTC Lock Register  

*****  

/*********************  

****/  

  

/*!  

 * @brief RTC_LR - RTC Lock Register (RW)  

 *  

 * Reset value: 0x000000FFU  

 */  

/*!!  

 * @name Constants and macros for entire RTC_LR register  

 */  

/*@{*/  

#define RTC_RD_LR(base)          (RTC_LR_REG(base))  

#define RTC_WR_LR(base, value)    (RTC_LR_REG(base) = (value))  

#define RTC_RMW_LR(base, mask, value)  (RTC_WR_LR(base, (RTC_RD_LR(base) &  

~(mask)) | (value)))  

#define RTC_SET_LR(base, value)   (BME_OR32(&RTC_LR_REG(base),  

(uint32_t)(value)))  

#define RTC_CLR_LR(base, value)   (BME_AND32(&RTC_LR_REG(base),  

(uint32_t)(~(value))))  

#define RTC_TOG_LR(base, value)   (BME_XOR32(&RTC_LR_REG(base),  

(uint32_t)(value)))  

/*@*/  

  

/*  

 * Constants & macros for individual RTC_LR bitfields  

 */  

  

/*!  

 * @name Register RTC_LR, field TCL[3] (RW)  

 *  

 * After being cleared, this bit can be set only by POR or software  

reset.  

 *  

 * Values:  

 * - 0b0 - Time Compensation Register is locked and writes are ignored.  

 * - 0b1 - Time Compensation Register is not locked and writes complete  

as  

 *      normal.

```

```

/*
/*@{*/
/*! @brief Read current value of the RTC_LR_TCL field. */
#define RTC_RD_LR_TCL(base) ((RTC_LR_REG(base) & RTC_LR_TCL_MASK) >>
RTC_LR_TCL_SHIFT)
#define RTC_BRD_LR_TCL(base) (BME_UBFX32(&RTC_LR_REG(base),
RTC_LR_TCL_SHIFT, RTC_LR_TCL_WIDTH))

/*! @brief Set the TCL field to a new value. */
#define RTC_WR_LR_TCL(base, value) (RTC_RMW_LR(base, RTC_LR_TCL_MASK,
RTC_LR_TCL(value)))
#define RTC_BWR_LR_TCL(base, value) (BME_BFI32(&RTC_LR_REG(base),
((uint32_t)(value) << RTC_LR_TCL_SHIFT), RTC_LR_TCL_SHIFT,
RTC_LR_TCL_WIDTH))
/*}@*/ */

/*!
* @name Register RTC_LR, field CRL[4] (RW)
*
* After being cleared, this bit can only be set by POR.
*
* Values:
* - 0b0 - Control Register is locked and writes are ignored.
* - 0b1 - Control Register is not locked and writes complete as normal.
*/
/*@{*/
/*! @brief Read current value of the RTC_LR_CRL field. */
#define RTC_RD_LR_CRL(base) ((RTC_LR_REG(base) & RTC_LR_CRL_MASK) >>
RTC_LR_CRL_SHIFT)
#define RTC_BRD_LR_CRL(base) (BME_UBFX32(&RTC_LR_REG(base),
RTC_LR_CRL_SHIFT, RTC_LR_CRL_WIDTH))

/*! @brief Set the CRL field to a new value. */
#define RTC_WR_LR_CRL(base, value) (RTC_RMW_LR(base, RTC_LR_CRL_MASK,
RTC_LR_CRL(value)))
#define RTC_BWR_LR_CRL(base, value) (BME_BFI32(&RTC_LR_REG(base),
((uint32_t)(value) << RTC_LR_CRL_SHIFT), RTC_LR_CRL_SHIFT,
RTC_LR_CRL_WIDTH))
/*}@*/ */

/*!
* @name Register RTC_LR, field SRL[5] (RW)
*
* After being cleared, this bit can be set only by POR or software
reset.
*
* Values:
* - 0b0 - Status Register is locked and writes are ignored.
* - 0b1 - Status Register is not locked and writes complete as normal.
*/
/*@{*/
/*! @brief Read current value of the RTC_LR_SRL field. */
#define RTC_RD_LR_SRL(base) ((RTC_LR_REG(base) & RTC_LR_SRL_MASK) >>
RTC_LR_SRL_SHIFT)

```

```

#define RTC_BRD_LR_SRL(base)  (BME_UBFX32(&RTC_LR_REG(base),  

RTC_LR_SRL_SHIFT, RTC_LR_SRL_WIDTH))

/*! @brief Set the SRL field to a new value. */  

#define RTC_WR_LR_SRL(base, value)  (RTC_RMW_LR(base, RTC_LR_SRL_MASK,  

RTC_LR_SRL(value)))  

#define RTC_BWR_LR_SRL(base, value)  (BME_BFI32(&RTC_LR_REG(base),  

(uint32_t)(value) << RTC_LR_SRL_SHIFT), RTC_LR_SRL_SHIFT,  

RTC_LR_SRL_WIDTH)  

/*@}*/

/*!  

 * @name Register RTC_LR, field LRL[6] (RW)  

 *  

 * After being cleared, this bit can be set only by POR or software  

reset.  

 *  

 * Values:  

 * - 0b0 - Lock Register is locked and writes are ignored.  

 * - 0b1 - Lock Register is not locked and writes complete as normal.  

 */  

/*@*/  

/*! @brief Read current value of the RTC_LR_LRL field. */  

#define RTC_RD_LR_LRL(base)  ((RTC_LR_REG(base) & RTC_LR_LRL_MASK) >>  

RTC_LR_LRL_SHIFT)  

#define RTC_BRD_LR_LRL(base)  (BME_UBFX32(&RTC_LR_REG(base),  

RTC_LR_LRL_SHIFT, RTC_LR_LRL_WIDTH))

/*! @brief Set the LRL field to a new value. */  

#define RTC_WR_LR_LRL(base, value)  (RTC_RMW_LR(base, RTC_LR_LRL_MASK,  

RTC_LR_LRL(value)))  

#define RTC_BWR_LR_LRL(base, value)  (BME_BFI32(&RTC_LR_REG(base),  

(uint32_t)(value) << RTC_LR_LRL_SHIFT), RTC_LR_LRL_SHIFT,  

RTC_LR_LRL_WIDTH)  

/*@*/

*****  

*****  

 * RTC_IER - RTC Interrupt Enable Register  

*****  

*****/  

  

/*!  

 * @brief RTC_IER - RTC Interrupt Enable Register (RW)  

 *  

 * Reset value: 0x00000007U  

 */  

/*!  

 * @name Constants and macros for entire RTC_IER register  

 */  

/*@*/  

#define RTC_RD_IER(base)          (RTC_IER_REG(base))  

#define RTC_WR_IER(base, value)   (RTC_IER_REG(base) = (value))

```

```

#define RTC_RMW_IER(base, mask, value) (RTC_WR_IER(base,
(RTC_RD_IER(base) & ~mask) | (value)))
#define RTC_SET_IER(base, value) (BME_OR32(&RTC_IER_REG(base),
(uint32_t)(value)))
#define RTC_CLR_IER(base, value) (BME_AND32(&RTC_IER_REG(base),
(uint32_t)(~value)))
#define RTC_TOG_IER(base, value) (BME_XOR32(&RTC_IER_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual RTC_IER bitfields
 */

/*!!
 * @name Register RTC_IER, field TIIE[0] (RW)
 *
 * Values:
 * - 0b0 - Time invalid flag does not generate an interrupt.
 * - 0b1 - Time invalid flag does generate an interrupt.
 */
/*@{*/
/**! @brief Read current value of the RTC_IER_TIIE field. */
#define RTC_RD_IER_TIIE(base) ((RTC_IER_REG(base) & RTC_IER_TIIE_MASK) >>
RTC_IER_TIIE_SHIFT)
#define RTC_BRD_IER_TIIE(base) (BME_UBFX32(&RTC_IER_REG(base),
RTC_IER_TIIE_SHIFT, RTC_IER_TIIE_WIDTH))

/*!! @brief Set the TIIE field to a new value. */
#define RTC_WR_IER_TIIE(base, value) (RTC_RMW_IER(base,
RTC_IER_TIIE_MASK, RTC_IER_TIIE(value)))
#define RTC_BWR_IER_TIIE(base, value) (BME_BFI32(&RTC_IER_REG(base),
((uint32_t)(value) << RTC_IER_TIIE_SHIFT), RTC_IER_TIIE_SHIFT,
RTC_IER_TIIE_WIDTH))
/*@}*/

/*!!
 * @name Register RTC_IER, field TOIE[1] (RW)
 *
 * Values:
 * - 0b0 - Time overflow flag does not generate an interrupt.
 * - 0b1 - Time overflow flag does generate an interrupt.
 */
/*@{*/
/**! @brief Read current value of the RTC_IER_TOIE field. */
#define RTC_RD_IER_TOIE(base) ((RTC_IER_REG(base) & RTC_IER_TOIE_MASK) >>
RTC_IER_TOIE_SHIFT)
#define RTC_BRD_IER_TOIE(base) (BME_UBFX32(&RTC_IER_REG(base),
RTC_IER_TOIE_SHIFT, RTC_IER_TOIE_WIDTH))

/*!! @brief Set the TOIE field to a new value. */
#define RTC_WR_IER_TOIE(base, value) (RTC_RMW_IER(base,
RTC_IER_TOIE_MASK, RTC_IER_TOIE(value)))

```

```

#define RTC_BWR_IER_TOIE(base, value) (BME_BFI32(&RTC_IER_REG(base),  

((uint32_t)(value) << RTC_IER_TOIE_SHIFT), RTC_IER_TOIE_SHIFT,  

RTC_IER_TOIE_WIDTH))  

/*@}*/

/*!  

 * @name Register RTC_IER, field TAIE[2] (RW)  

 *  

 * Values:  

 * - 0b0 - Time alarm flag does not generate an interrupt.  

 * - 0b1 - Time alarm flag does generate an interrupt.  

 */  

/*@{/*/  

/*! @brief Read current value of the RTC_IER_TAIE field. */  

#define RTC_RD_IER_TAIE(base) ((RTC_IER_REG(base) & RTC_IER_TAIE_MASK) >>  

RTC_IER_TAIE_SHIFT)  

#define RTC_BRD_IER_TAIE(base) (BME_UBFX32(&RTC_IER_REG(base),  

RTC_IER_TAIE_SHIFT, RTC_IER_TAIE_WIDTH))

/*! @brief Set the TAIE field to a new value. */  

#define RTC_WR_IER_TAIE(base, value) (RTC_RMW_IER(base,  

RTC_IER_TAIE_MASK, RTC_IER_TAIE(value)))  

#define RTC_BWR_IER_TAIE(base, value) (BME_BFI32(&RTC_IER_REG(base),  

((uint32_t)(value) << RTC_IER_TAIE_SHIFT), RTC_IER_TAIE_SHIFT,  

RTC_IER_TAIE_WIDTH))  

/*@{/*/  

/*!  

 * @name Register RTC_IER, field TSIE[4] (RW)  

 *  

 * The seconds interrupt is an edge-sensitive interrupt with a dedicated  

 * interrupt vector. It is generated once a second and requires no  

software overhead  

 * (there is no corresponding status flag to clear).  

 *  

 * Values:  

 * - 0b0 - Seconds interrupt is disabled.  

 * - 0b1 - Seconds interrupt is enabled.  

 */  

/*@{/*/  

/*! @brief Read current value of the RTC_IER_TSIE field. */  

#define RTC_RD_IER_TSIE(base) ((RTC_IER_REG(base) & RTC_IER_TSIE_MASK) >>  

RTC_IER_TSIE_SHIFT)  

#define RTC_BRD_IER_TSIE(base) (BME_UBFX32(&RTC_IER_REG(base),  

RTC_IER_TSIE_SHIFT, RTC_IER_TSIE_WIDTH))

/*! @brief Set the TSIE field to a new value. */  

#define RTC_WR_IER_TSIE(base, value) (RTC_RMW_IER(base,  

RTC_IER_TSIE_MASK, RTC_IER_TSIE(value)))  

#define RTC_BWR_IER_TSIE(base, value) (BME_BFI32(&RTC_IER_REG(base),  

((uint32_t)(value) << RTC_IER_TSIE_SHIFT), RTC_IER_TSIE_SHIFT,  

RTC_IER_TSIE_WIDTH))  

/*@{/*/

```

```

/*!
 * @name Register RTC_IER, field WPON[7] (RW)
 *
 * The wakeup pin is optional and not available on all devices. Whenever
 * the
 *   wakeup pin is enabled and this bit is set, the wakeup pin will assert.
 *
 * Values:
 * - 0b0 - No effect.
 * - 0b1 - If the wakeup pin is enabled, then the wakeup pin will assert.
 */
/*@{ */

/*! @brief Read current value of the RTC_IER_WPON field. */
#define RTC_RD_IER_WPON(base) ((RTC_IER_REG(base) & RTC_IER_WPON_MASK) >>
RTC_IER_WPON_SHIFT)
#define RTC_BRD_IER_WPON(base) (BME_UBFX32(&RTC_IER_REG(base),
RTC_IER_WPON_SHIFT, RTC_IER_WPON_WIDTH))

/*! @brief Set the WPON field to a new value. */
#define RTC_WR_IER_WPON(base, value) (RTC_RMW_IER(base,
RTC_IER_WPON_MASK, RTC_IER_WPON(value)))
#define RTC_BWR_IER_WPON(base, value) (BME_BFI32(&RTC_IER_REG(base),
((uint32_t)(value) << RTC_IER_WPON_SHIFT), RTC_IER_WPON_SHIFT,
RTC_IER_WPON_WIDTH))
/*}@*/}

/*
 * MKL25Z4 SIM
 *
 * System Integration Module
 *
 * Registers defined in this header file:
 * - SIM_SOPT1 - System Options Register 1
 * - SIM_SOPT1CFG - SOPT1 Configuration Register
 * - SIM_SOPT2 - System Options Register 2
 * - SIM_SOPT4 - System Options Register 4
 * - SIM_SOPT5 - System Options Register 5
 * - SIM_SOPT7 - System Options Register 7
 * - SIM_SDID - System Device Identification Register
 * - SIM_SCGC4 - System Clock Gating Control Register 4
 * - SIM_SCGC5 - System Clock Gating Control Register 5
 * - SIM_SCGC6 - System Clock Gating Control Register 6
 * - SIM_SCGC7 - System Clock Gating Control Register 7
 * - SIM_CLKDIV1 - System Clock Divider Register 1
 * - SIM_FCFG1 - Flash Configuration Register 1
 * - SIM_FCFG2 - Flash Configuration Register 2
 * - SIM_UIDMH - Unique Identification Register Mid-High
 * - SIM_UIDML - Unique Identification Register Mid Low
 * - SIM_UIDL - Unique Identification Register Low
 * - SIM_COPC - COP Control Register
 * - SIM_SRVCOP - Service COP Register
*/

```

```

#define SIM_INSTANCE_COUNT (1U) /*!< Number of instances of the SIM
module. */
#define SIM_IDX (0U) /*!< Instance number for SIM. */

/********************* SIM_SOPT1 - System Options Register 1 *****/
****

 * SIM_SOPT1 - System Options Register 1

***** */

/*!
 * @brief SIM_SOPT1 - System Options Register 1 (RW)
 *
 * Reset value: 0x80000000U
 *
 * The SOPT1 register is only reset on POR or LVD.
 */
/*!
 * @name Constants and macros for entire SIM_SOPT1 register
 */
/*@{ */

#define SIM_RD_SOPT1(base) (SIM_SOPT1_REG(base))
#define SIM_WR_SOPT1(base, value) (SIM_SOPT1_REG(base) = (value))
#define SIM_RMW_SOPT1(base, mask, value) (SIM_WR_SOPT1(base,
(SIM_RD_SOPT1(base) & ~mask) | (value)))
#define SIM_SET_SOPT1(base, value) (BME_OR32(&SIM_SOPT1_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT1(base, value) (BME_AND32(&SIM_SOPT1_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT1(base, value) (BME_XOR32(&SIM_SOPT1_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual SIM_SOPT1 bitfields
 */

/*!
 * @name Register SIM_SOPT1, field OSC32KSEL[19:18] (RW)
 *
 * Selects the 32 kHz clock source (ERCLK32K) for RTC and LPTMR. This bit
is
 * reset only on POR/LVD.
 *
 * Values:
 * - 0b00 - System oscillator (OSC32KCLK)
 * - 0b01 - Reserved
 * - 0b10 - RTC_CLKIN
 * - 0b11 - LPO 1kHz
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT1_OSC32KSEL field. */

```

```

#define SIM_RD_SOPT1_OSC32KSEL(base) ((SIM_SOPT1_REG(base) &
SIM_SOPT1_OSC32KSEL_MASK) >> SIM_SOPT1_OSC32KSEL_SHIFT)
#define SIM_BRD_SOPT1_OSC32KSEL(base) (BME_UBFX32(&SIM_SOPT1_REG(base)),
SIM_SOPT1_OSC32KSEL_SHIFT, SIM_SOPT1_OSC32KSEL_WIDTH))

/*! @brief Set the OSC32KSEL field to a new value. */
#define SIM_WR_SOPT1_OSC32KSEL(base, value) (SIM_RMW_SOPT1(base,
SIM_SOPT1_OSC32KSEL_MASK, SIM_SOPT1_OSC32KSEL(value)))
#define SIM_BWR_SOPT1_OSC32KSEL(base, value)
(BME_BFI32(&SIM_SOPT1_REG(base), ((uint32_t)(value) <<
SIM_SOPT1_OSC32KSEL_SHIFT), SIM_SOPT1_OSC32KSEL_SHIFT,
SIM_SOPT1_OSC32KSEL_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SOPT1, field USBVSTBY[29] (RW)
 *
 * Controls whether the USB voltage regulator is placed in standby mode
during
 * VLPR and VLPW modes.
 *
 * Values:
 * - 0b0 - USB voltage regulator not in standby during VLPR and VLPW
modes.
 * - 0b1 - USB voltage regulator in standby during VLPR and VLPW modes.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT1_USBVSTBY field. */
#define SIM_RD_SOPT1_USBVSTBY(base) ((SIM_SOPT1_REG(base) &
SIM_SOPT1_USBVSTBY_MASK) >> SIM_SOPT1_USBVSTBY_SHIFT)
#define SIM_BRD_SOPT1_USBVSTBY(base) (BME_UBFX32(&SIM_SOPT1_REG(base),
SIM_SOPT1_USBVSTBY_SHIFT, SIM_SOPT1_USBVSTBY_WIDTH))

/*! @brief Set the USBVSTBY field to a new value. */
#define SIM_WR_SOPT1_USBVSTBY(base, value) (SIM_RMW_SOPT1(base,
SIM_SOPT1_USBVSTBY_MASK, SIM_SOPT1_USBVSTBY(value)))
#define SIM_BWR_SOPT1_USBVSTBY(base, value)
(BME_BFI32(&SIM_SOPT1_REG(base), ((uint32_t)(value) <<
SIM_SOPT1_USBVSTBY_SHIFT), SIM_SOPT1_USBVSTBY_SHIFT,
SIM_SOPT1_USBVSTBY_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SOPT1, field USBSSTBY[30] (RW)
 *
 * Controls whether the USB voltage regulator is placed in standby mode
during
 * Stop, VLPS, LLS and VLLS modes.
 *
 * Values:
 * - 0b0 - USB voltage regulator not in standby during Stop, VLPS, LLS
and VLLS
 *      modes.
 */

```

```

 * - 0b1 - USB voltage regulator in standby during Stop, VLPS, LLS and
VLLS
 *      modes.
 */
/*@{ */
/*! @brief Read current value of the SIM_SOPT1_USBSSTBY field. */
#define SIM_RD_SOPT1_USBSSTBY(base) ((SIM_SOPT1_REG(base) &
SIM_SOPT1_USBSSTBY_MASK) >> SIM_SOPT1_USBSSTBY_SHIFT)
#define SIM_BRD_SOPT1_USBSSTBY(base) (BME_UBFX32(&SIM_SOPT1_REG(base),
SIM_SOPT1_USBSSTBY_SHIFT, SIM_SOPT1_USBSSTBY_WIDTH))

/*! @brief Set the USBSSTBY field to a new value. */
#define SIM_WR_SOPT1_USBSSTBY(base, value) (SIM_RMW_SOPT1(base,
SIM_SOPT1_USBSSTBY_MASK, SIM_SOPT1_USBSSTBY(value)))
#define SIM_BWR_SOPT1_USBSSTBY(base, value)
(BME_BFI32(&SIM_SOPT1_REG(base), ((uint32_t)(value) <<
SIM_SOPT1_USBSSTBY_SHIFT), SIM_SOPT1_USBSSTBY_SHIFT,
SIM_SOPT1_USBSSTBY_WIDTH))
/*@} */

/*!
 * @name Register SIM_SOPT1, field USBREGEN[31] (RW)
 *
 * Controls whether the USB voltage regulator is enabled.
 *
 * Values:
 * - 0b0 - USB voltage regulator is disabled.
 * - 0b1 - USB voltage regulator is enabled.
 */
/*@{ */
/*! @brief Read current value of the SIM_SOPT1_USBREGEN field. */
#define SIM_RD_SOPT1_USBREGEN(base) ((SIM_SOPT1_REG(base) &
SIM_SOPT1_USBREGEN_MASK) >> SIM_SOPT1_USBREGEN_SHIFT)
#define SIM_BRD_SOPT1_USBREGEN(base) (BME_UBFX32(&SIM_SOPT1_REG(base),
SIM_SOPT1_USBREGEN_SHIFT, SIM_SOPT1_USBREGEN_WIDTH))

/*! @brief Set the USBREGEN field to a new value. */
#define SIM_WR_SOPT1_USBREGEN(base, value) (SIM_RMW_SOPT1(base,
SIM_SOPT1_USBREGEN_MASK, SIM_SOPT1_USBREGEN(value)))
#define SIM_BWR_SOPT1_USBREGEN(base, value)
(BME_BFI32(&SIM_SOPT1_REG(base), ((uint32_t)(value) <<
SIM_SOPT1_USBREGEN_SHIFT), SIM_SOPT1_USBREGEN_SHIFT,
SIM_SOPT1_USBREGEN_WIDTH))
/*@} */

/******************
*****
 * SIM_SOPT1CFG - SOPT1 Configuration Register
*****
******************/

/*!
 * @brief SIM_SOPT1CFG - SOPT1 Configuration Register (RW)

```

```

/*
 * Reset value: 0x00000000U
 *
 * The SOPT1CFG register is reset on System Reset not VLLS.
 */
/*!
 * @name Constants and macros for entire SIM_SOPT1CFG register
 */
/*@{*/
#define SIM_RD_SOPT1CFG(base)      (SIM_SOPT1CFG_REG(base))
#define SIM_WR_SOPT1CFG(base, value) (SIM_SOPT1CFG_REG(base) = (value))
#define SIM_RMW_SOPT1CFG(base, mask, value) (SIM_WR_SOPT1CFG(base,
(SIM_RD_SOPT1CFG(base) & ~mask) | (value)))
#define SIM_SET_SOPT1CFG(base, value) (BME_OR32(&SIM_SOPT1CFG_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT1CFG(base, value) (BME_AND32(&SIM_SOPT1CFG_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT1CFG(base, value) (BME_XOR32(&SIM_SOPT1CFG_REG(base),
(uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual SIM_SOPT1CFG bitfields
 */
/*!
 * @name Register SIM_SOPT1CFG, field URWE[24] (RW)
 *
 * Writing one to the URWE bit allows the SOPT1 USBREGEN bit to be
written. This
 * register bit clears after a write to USBREGEN.
 *
 * Values:
 * - 0b0 - SOPT1 USBREGEN cannot be written.
 * - 0b1 - SOPT1 USBREGEN can be written.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT1CFG_URWE field. */
#define SIM_RD_SOPT1CFG_URWE(base) ((SIM_SOPT1CFG_REG(base) &
SIM_SOPT1CFG_URWE_MASK) >> SIM_SOPT1CFG_URWE_SHIFT)
#define SIM_BRD_SOPT1CFG_URWE(base) (BME_UBFX32(&SIM_SOPT1CFG_REG(base),
SIM_SOPT1CFG_URWE_SHIFT, SIM_SOPT1CFG_URWE_WIDTH))

/*! @brief Set the URWE field to a new value. */
#define SIM_WR_SOPT1CFG_URWE(base, value) (SIM_RMW_SOPT1CFG(base,
SIM_SOPT1CFG_URWE_MASK, SIM_SOPT1CFG_URWE(value)))
#define SIM_BWR_SOPT1CFG_URWE(base, value)
(BME_BFI32(&SIM_SOPT1CFG_REG(base), ((uint32_t)(value) <<
SIM_SOPT1CFG_URWE_SHIFT), SIM_SOPT1CFG_URWE_SHIFT,
SIM_SOPT1CFG_URWE_WIDTH))
/*}@*/ */

/*!
 * @name Register SIM_SOPT1CFG, field UVSWE[25] (RW)

```

```

/*
 * Writing one to the UVSWE bit allows the SOPT1 USBVSTBY bit to be
written.
 * This register bit clears after a write to USBVSTBY.
 *
 * Values:
 * - 0b0 - SOPT1 USBVSTB cannot be written.
 * - 0b1 - SOPT1 USBVSTB can be written.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT1CFG_UVSWE field. */
#define SIM_RD_SOPT1CFG_UVSWE(base) ((SIM_SOPT1CFG_REG(base) &
SIM_SOPT1CFG_UVSWE_MASK) >> SIM_SOPT1CFG_UVSWE_SHIFT)
#define SIM_BRD_SOPT1CFG_UVSWE(base) (BME_UBFX32(&SIM_SOPT1CFG_REG(base),
SIM_SOPT1CFG_UVSWE_SHIFT, SIM_SOPT1CFG_UVSWE_WIDTH))

/*! @brief Set the UVSWE field to a new value. */
#define SIM_WR_SOPT1CFG_UVSWE(base, value) (SIM_RMW_SOPT1CFG(base,
SIM_SOPT1CFG_UVSWE_MASK, SIM_SOPT1CFG_UVSWE(value)))
#define SIM_BWR_SOPT1CFG_UVSWE(base, value)
(BME_BFI32(&SIM_SOPT1CFG_REG(base), ((uint32_t)(value) <<
SIM_SOPT1CFG_UVSWE_SHIFT), SIM_SOPT1CFG_UVSWE_SHIFT,
SIM_SOPT1CFG_UVSWE_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register SIM_SOPT1CFG, field USSWE[26] (RW)
 *
 * Writing one to the USSWE bit allows the SOPT1 USBSSTBY bit to be
written.
 * This register bit clears after a write to USBSSTBY.
 *
 * Values:
 * - 0b0 - SOPT1 USBSSTB cannot be written.
 * - 0b1 - SOPT1 USBSSTB can be written.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT1CFG_USSWE field. */
#define SIM_RD_SOPT1CFG_USSWE(base) ((SIM_SOPT1CFG_REG(base) &
SIM_SOPT1CFG_USSWE_MASK) >> SIM_SOPT1CFG_USSWE_SHIFT)
#define SIM_BRD_SOPT1CFG_USSWE(base) (BME_UBFX32(&SIM_SOPT1CFG_REG(base),
SIM_SOPT1CFG_USSWE_SHIFT, SIM_SOPT1CFG_USSWE_WIDTH))

/*! @brief Set the USSWE field to a new value. */
#define SIM_WR_SOPT1CFG_USSWE(base, value) (SIM_RMW_SOPT1CFG(base,
SIM_SOPT1CFG_USSWE_MASK, SIM_SOPT1CFG_USSWE(value)))
#define SIM_BWR_SOPT1CFG_USSWE(base, value)
(BME_BFI32(&SIM_SOPT1CFG_REG(base), ((uint32_t)(value) <<
SIM_SOPT1CFG_USSWE_SHIFT), SIM_SOPT1CFG_USSWE_SHIFT,
SIM_SOPT1CFG_USSWE_WIDTH))
/*}@*/
```

  

```

/*****
*****
```

```

 * SIM_SOPT2 - System Options Register 2
 ****
 **** */

/*!
 * @brief SIM_SOPT2 - System Options Register 2 (RW)
 *
 * Reset value: 0x00000000U
 *
 * SOPT2 contains the controls for selecting many of the module clock
 * source
 * options on this device. See the Clock Distribution chapter for more
 * information
 * including clocking diagrams and definitions of device clocks.
 */
/*!
 * @name Constants and macros for entire SIM_SOPT2 register
 */
/*@{*/
#define SIM_RD_SOPT2(base)      (SIM_SOPT2_REG(base))
#define SIM_WR_SOPT2(base, value) (SIM_SOPT2_REG(base) = (value))
#define SIM_RMW_SOPT2(base, mask, value) (SIM_WR_SOPT2(base,
(SIM_RD_SOPT2(base) & ~mask) | (value)))
#define SIM_SET_SOPT2(base, value) (BME_OR32(&SIM_SOPT2_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT2(base, value) (BME_AND32(&SIM_SOPT2_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT2(base, value) (BME_XOR32(&SIM_SOPT2_REG(base),
(uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual SIM_SOPT2 bitfields
 */
/*!
 * @name Register SIM_SOPT2, field RTCCLKOUTSEL[4] (RW)
 *
 * Selects either the RTC 1 Hz clock or the OSC clock to be output on the
 * RTC_CLKOUT pin.
 *
 * Values:
 * - 0b0 - RTC 1 Hz clock is output on the RTC_CLKOUT pin.
 * - 0b1 - OSCERCLK clock is output on the RTC_CLKOUT pin.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT2_RTCCLKOUTSEL field. */
#define SIM_RD_SOPT2_RTCCLKOUTSEL(base) ((SIM_SOPT2_REG(base) &
SIM_SOPT2_RTCCLKOUTSEL_MASK) >> SIM_SOPT2_RTCCLKOUTSEL_SHIFT)
#define SIM_BRD_SOPT2_RTCCLKOUTSEL(base)
(BME_UBFX32(&SIM_SOPT2_REG(base), SIM_SOPT2_RTCCLKOUTSEL_SHIFT,
SIM_SOPT2_RTCCLKOUTSEL_WIDTH))

```

```

/*! @brief Set the RTCCLKOUTSEL field to a new value. */
#define SIM_WR_SOPT2_RTCCLKOUTSEL(base, value) (SIM_RMW_SOPT2(base,
SIM_SOPT2_RTCCLKOUTSEL_MASK, SIM_SOPT2_RTCCLKOUTSEL(value)))
#define SIM_BWR_SOPT2_RTCCLKOUTSEL(base, value)
(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<
SIM_SOPT2_RTCCLKOUTSEL_SHIFT), SIM_SOPT2_RTCCLKOUTSEL_SHIFT,
SIM_SOPT2_RTCCLKOUTSEL_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SOPT2, field CLKOUTSEL[7:5] (RW)
 *
 * Selects the clock to output on the CLKOUT pin.
 *
 * Values:
 * - 0b00 - Reserved
 * - 0b01 - Reserved
 * - 0b10 - Bus clock
 * - 0b11 - LPO clock (1 kHz)
 * - 0b100 - MCGIRCLK
 * - 0b101 - Reserved
 * - 0b110 - OSCERCLK
 * - 0b111 - Reserved
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT2_CLKOUTSEL field. */
#define SIM_RD_SOPT2_CLKOUTSEL(base) ((SIM_SOPT2_REG(base) &
SIM_SOPT2_CLKOUTSEL_MASK) >> SIM_SOPT2_CLKOUTSEL_SHIFT)
#define SIM_BRD_SOPT2_CLKOUTSEL(base) (BME_UBFX32(&SIM_SOPT2_REG(base),
SIM_SOPT2_CLKOUTSEL_SHIFT, SIM_SOPT2_CLKOUTSEL_WIDTH))

/*! @brief Set the CLKOUTSEL field to a new value. */
#define SIM_WR_SOPT2_CLKOUTSEL(base, value) (SIM_RMW_SOPT2(base,
SIM_SOPT2_CLKOUTSEL_MASK, SIM_SOPT2_CLKOUTSEL(value)))
#define SIM_BWR_SOPT2_CLKOUTSEL(base, value)
(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<
SIM_SOPT2_CLKOUTSEL_SHIFT), SIM_SOPT2_CLKOUTSEL_SHIFT,
SIM_SOPT2_CLKOUTSEL_WIDTH))
/*@*/

/*!!
 * @name Register SIM_SOPT2, field PLLFLLSEL[16] (RW)
 *
 * Selects the MCGPLLCLK or MCGFLLCLK clock for various peripheral
clocking
 * options.
 *
 * Values:
 * - 0b0 - MCGFLLCLK clock
 * - 0b1 - MCGPLLCLK clock with fixed divide by two
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT2_PLLFLLSEL field. */

```

```

#define SIM_RD_SOPT2_PLLFLLSEL(base) ((SIM_SOPT2_REG(base) &
SIM_SOPT2_PLLFLLSEL_MASK) >> SIM_SOPT2_PLLFLLSEL_SHIFT)
#define SIM_BRD_SOPT2_PLLFLLSEL(base) (BME_UBFX32(&SIM_SOPT2_REG(base),
SIM_SOPT2_PLLFLLSEL_SHIFT, SIM_SOPT2_PLLFLLSEL_WIDTH))

/*! @brief Set the PLLFLLSEL field to a new value. */
#define SIM_WR_SOPT2_PLLFLLSEL(base, value) (SIM_RMW_SOPT2(base,
SIM_SOPT2_PLLFLLSEL_MASK, SIM_SOPT2_PLLFLLSEL(value)))
#define SIM_BWR_SOPT2_PLLFLLSEL(base, value)
(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<
SIM_SOPT2_PLLFLLSEL_SHIFT), SIM_SOPT2_PLLFLLSEL_SHIFT,
SIM_SOPT2_PLLFLLSEL_WIDTH))
/*@}*/

/*!!
* @name Register SIM_SOPT2, field USBSRC[18] (RW)
*
* Selects the clock source for the USB 48 MHz clock.
*
* Values:
* - 0b0 - External bypass clock (USB_CLKIN).
* - 0b1 - MCGPLLCLK/2 or MCGFLLCLK clock
*/
/*@{*/
/*! @brief Read current value of the SIM_SOPT2_USBSRC field. */
#define SIM_RD_SOPT2_USBSRC(base) ((SIM_SOPT2_REG(base) &
SIM_SOPT2_USBSRC_MASK) >> SIM_SOPT2_USBSRC_SHIFT)
#define SIM_BRD_SOPT2_USBSRC(base) (BME_UBFX32(&SIM_SOPT2_REG(base),
SIM_SOPT2_USBSRC_SHIFT, SIM_SOPT2_USBSRC_WIDTH))

/*! @brief Set the USBSRC field to a new value. */
#define SIM_WR_SOPT2_USBSRC(base, value) (SIM_RMW_SOPT2(base,
SIM_SOPT2_USBSRC_MASK, SIM_SOPT2_USBSRC(value)))
#define SIM_BWR_SOPT2_USBSRC(base, value)
(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<
SIM_SOPT2_USBSRC_SHIFT), SIM_SOPT2_USBSRC_SHIFT, SIM_SOPT2_USBSRC_WIDTH))
/*@}*/

/*!!
* @name Register SIM_SOPT2, field TPMSRC[25:24] (RW)
*
* Selects the clock source for the TPM counter clock
*
* Values:
* - 0b00 - Clock disabled
* - 0b01 - MCGFLLCLK clock or MCGPLLCLK/2
* - 0b10 - OSCERCLK clock
* - 0b11 - MCGIRCLK clock
*/
/*@{*/
/*! @brief Read current value of the SIM_SOPT2_TPMSRC field. */
#define SIM_RD_SOPT2_TPMSRC(base) ((SIM_SOPT2_REG(base) &
SIM_SOPT2_TPMSRC_MASK) >> SIM_SOPT2_TPMSRC_SHIFT)

```

```

#define SIM_BRD_SOPT2_TPMSRC(base) (BME_UBFX32(&SIM_SOPT2_REG(base),  

SIM_SOPT2_TPMSRC_SHIFT, SIM_SOPT2_TPMSRC_WIDTH))

/*! @brief Set the TPMSRC field to a new value. */  

#define SIM_WR_SOPT2_TPMSRC(base, value) (SIM_RMW_SOPT2(base,  

SIM_SOPT2_TPMSRC_MASK, SIM_SOPT2_TPMSRC(value)))  

#define SIM_BWR_SOPT2_TPMSRC(base, value)  

(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<  

SIM_SOPT2_TPMSRC_SHIFT), SIM_SOPT2_TPMSRC_SHIFT, SIM_SOPT2_TPMSRC_WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_SOPT2, field UART0SRC[27:26] (RW)  

 *  

 * Selects the clock source for the UART0 transmit and receive clock.  

 *  

 * Values:  

 * - 0b00 - Clock disabled  

 * - 0b01 - MCGFLLCLK clock or MCGPLLCLK/2 clock  

 * - 0b10 - OSCERCLK clock  

 * - 0b11 - MCGIRCLK clock  

 */  

/*@{ */  

/*! @brief Read current value of the SIM_SOPT2_UART0SRC field. */  

#define SIM_RD_SOPT2_UART0SRC(base) ((SIM_SOPT2_REG(base) &  

SIM_SOPT2_UART0SRC_MASK) >> SIM_SOPT2_UART0SRC_SHIFT)  

#define SIM_BRD_SOPT2_UART0SRC(base) (BME_UBFX32(&SIM_SOPT2_REG(base),  

SIM_SOPT2_UART0SRC_SHIFT, SIM_SOPT2_UART0SRC_WIDTH))

/*! @brief Set the UART0SRC field to a new value. */  

#define SIM_WR_SOPT2_UART0SRC(base, value) (SIM_RMW_SOPT2(base,  

SIM_SOPT2_UART0SRC_MASK, SIM_SOPT2_UART0SRC(value)))  

#define SIM_BWR_SOPT2_UART0SRC(base, value)  

(BME_BFI32(&SIM_SOPT2_REG(base), ((uint32_t)(value) <<  

SIM_SOPT2_UART0SRC_SHIFT), SIM_SOPT2_UART0SRC_SHIFT,  

SIM_SOPT2_UART0SRC_WIDTH))  

/*@}*/

*****  

*****  

* SIM_SOPT4 - System Options Register 4  

*****  

***** /
```

```

/*!  

 * @brief SIM_SOPT4 - System Options Register 4 (RW)  

 *  

 * Reset value: 0x00000000U  

 */  

/*!  

 * @name Constants and macros for entire SIM_SOPT4 register  

 */  

/*@{ */

```

```

#define SIM_RD_SOPT4(base)          (SIM_SOPT4_REG(base))
#define SIM_WR_SOPT4(base, value)   (SIM_SOPT4_REG(base) = (value))
#define SIM_RMW_SOPT4(base, mask, value) (SIM_WR_SOPT4(base,
(SIM_RD_SOPT4(base) & ~mask)) | (value)))
#define SIM_SET_SOPT4(base, value)  (BME_OR32(&SIM_SOPT4_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT4(base, value)  (BME_AND32(&SIM_SOPT4_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT4(base, value)  (BME_XOR32(&SIM_SOPT4_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual SIM_SOPT4 bitfields
 */

/*!!
 * @name Register SIM_SOPT4, field TPM1CH0SRC[18] (RW)
 *
 * Selects the source for TPM1 channel 0 input capture. When TPM1 is not
in
 * input capture mode, clear this field.
 *
 * Values:
 * - 0b0 - TPM1_CH0 signal
 * - 0b1 - CMP0 output
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT4_TPM1CH0SRC field. */
#define SIM_RD_SOPT4_TPM1CH0SRC(base) ((SIM_SOPT4_REG(base) &
SIM_SOPT4_TPM1CH0SRC_MASK) >> SIM_SOPT4_TPM1CH0SRC_SHIFT)
#define SIM_BRD_SOPT4_TPM1CH0SRC(base) (BME_UBFX32(&SIM_SOPT4_REG(base),
SIM_SOPT4_TPM1CH0SRC_SHIFT, SIM_SOPT4_TPM1CH0SRC_WIDTH))

/*! @brief Set the TPM1CH0SRC field to a new value. */
#define SIM_WR_SOPT4_TPM1CH0SRC(base, value) (SIM_RMW_SOPT4(base,
SIM_SOPT4_TPM1CH0SRC_MASK, SIM_SOPT4_TPM1CH0SRC(value)))
#define SIM_BWR_SOPT4_TPM1CH0SRC(base, value)
(BME_BFI32(&SIM_SOPT4_REG(base), ((uint32_t)(value) <<
SIM_SOPT4_TPM1CH0SRC_SHIFT), SIM_SOPT4_TPM1CH0SRC_SHIFT,
SIM_SOPT4_TPM1CH0SRC_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SOPT4, field TPM2CH0SRC[20] (RW)
 *
 * Selects the source for TPM2 channel 0 input capture. When TPM2 is not
in
 * input capture mode, clear this field.
 *
 * Values:
 * - 0b0 - TPM2_CH0 signal
 * - 0b1 - CMP0 output
 */

```

```

/*@{*/
/*! @brief Read current value of the SIM_SOPT4_TPM2CH0SRC field. */
#define SIM_RD_SOPT4_TPM2CH0SRC(base) ((SIM_SOPT4_REG(base) &
SIM_SOPT4_TPM2CH0SRC_MASK) >> SIM_SOPT4_TPM2CH0SRC_SHIFT)
#define SIM_BRD_SOPT4_TPM2CH0SRC(base) (BME_UBFX32(&SIM_SOPT4_REG(base),
SIM_SOPT4_TPM2CH0SRC_SHIFT, SIM_SOPT4_TPM2CH0SRC_WIDTH))

/*! @brief Set the TPM2CH0SRC field to a new value. */
#define SIM_WR_SOPT4_TPM2CH0SRC(base, value) (SIM_RMW_SOPT4(base,
SIM_SOPT4_TPM2CH0SRC_MASK, SIM_SOPT4_TPM2CH0SRC(value)))
#define SIM_BWR_SOPT4_TPM2CH0SRC(base, value)
(BME_BFI32(&SIM_SOPT4_REG(base), ((uint32_t)(value) <<
SIM_SOPT4_TPM2CH0SRC_SHIFT), SIM_SOPT4_TPM2CH0SRC_SHIFT,
SIM_SOPT4_TPM2CH0SRC_WIDTH))
/*}@*/
```

/\*!

- \* @name Register SIM\_SOPT4, field TPM0CLKSEL[24] (RW)
- \*
- \* Selects the external pin used to drive the clock to the TPM0 module.

The

- \* selected pin must also be configured for the TPM external clock function through
- \* the appropriate pin control register in the port control module.
- \*
- \* Values:
- \* - 0b0 - TPM0 external clock driven by TPM\_CLKIN0 pin.
- \* - 0b1 - TPM0 external clock driven by TPM\_CLKIN1 pin.
- \*/

```
/*@{*/
/*! @brief Read current value of the SIM_SOPT4_TPM0CLKSEL field. */
#define SIM_RD_SOPT4_TPM0CLKSEL(base) ((SIM_SOPT4_REG(base) &
SIM_SOPT4_TPM0CLKSEL_MASK) >> SIM_SOPT4_TPM0CLKSEL_SHIFT)
#define SIM_BRD_SOPT4_TPM0CLKSEL(base) (BME_UBFX32(&SIM_SOPT4_REG(base),
SIM_SOPT4_TPM0CLKSEL_SHIFT, SIM_SOPT4_TPM0CLKSEL_WIDTH))

/*! @brief Set the TPM0CLKSEL field to a new value. */
#define SIM_WR_SOPT4_TPM0CLKSEL(base, value) (SIM_RMW_SOPT4(base,
SIM_SOPT4_TPM0CLKSEL_MASK, SIM_SOPT4_TPM0CLKSEL(value)))
#define SIM_BWR_SOPT4_TPM0CLKSEL(base, value)
(BME_BFI32(&SIM_SOPT4_REG(base), ((uint32_t)(value) <<
SIM_SOPT4_TPM0CLKSEL_SHIFT), SIM_SOPT4_TPM0CLKSEL_SHIFT,
SIM_SOPT4_TPM0CLKSEL_WIDTH))
/*}@*/
```

/\*!

- \* @name Register SIM\_SOPT4, field TPM1CLKSEL[25] (RW)
- \*
- \* Selects the external pin used to drive the clock to the TPM1 module.

The

- \* selected pin must also be configured for the TPM external clock function through
- \* the appropriate pin control register in the port control module.
- \*

```

* Values:
* - 0b0 - TPM1 external clock driven by TPM_CLKIN0 pin.
* - 0b1 - TPM1 external clock driven by TPM_CLKIN1 pin.
*/
/*@{*/
/*! @brief Read current value of the SIM_SOPT4_TPM1CLKSEL field. */
#define SIM_RD_SOPT4_TPM1CLKSEL(base) ((SIM_SOPT4_REG(base) &
SIM_SOPT4_TPM1CLKSEL_MASK) >> SIM_SOPT4_TPM1CLKSEL_SHIFT)
#define SIM_BRD_SOPT4_TPM1CLKSEL(base) (BME_UBX32(&SIM_SOPT4_REG(base),
SIM_SOPT4_TPM1CLKSEL_SHIFT, SIM_SOPT4_TPM1CLKSEL_WIDTH))

/*! @brief Set the TPM1CLKSEL field to a new value. */
#define SIM_WR_SOPT4_TPM1CLKSEL(base, value) (SIM_RMW_SOPT4(base,
SIM_SOPT4_TPM1CLKSEL_MASK, SIM_SOPT4_TPM1CLKSEL(value)))
#define SIM_BWR_SOPT4_TPM1CLKSEL(base, value)
(BME_BFI32(&SIM_SOPT4_REG(base), ((uint32_t)(value) <<
SIM_SOPT4_TPM1CLKSEL_SHIFT), SIM_SOPT4_TPM1CLKSEL_SHIFT,
SIM_SOPT4_TPM1CLKSEL_WIDTH))
/*}@*/ */

/*!
* @name Register SIM_SOPT4, field TPM2CLKSEL[26] (RW)
*
* Selects the external pin used to drive the clock to the TPM2 module.
The
* selected pin must also be configured for the TPM external clock
function through
* the appropriate pin control register in the port control module.
*
* Values:
* - 0b0 - TPM2 external clock driven by TPM_CLKIN0 pin.
* - 0b1 - TPM2 external clock driven by TPM_CLKIN1 pin.
*/
/*@{*/
/*! @brief Read current value of the SIM_SOPT4_TPM2CLKSEL field. */
#define SIM_RD_SOPT4_TPM2CLKSEL(base) ((SIM_SOPT4_REG(base) &
SIM_SOPT4_TPM2CLKSEL_MASK) >> SIM_SOPT4_TPM2CLKSEL_SHIFT)
#define SIM_BRD_SOPT4_TPM2CLKSEL(base) (BME_UBX32(&SIM_SOPT4_REG(base),
SIM_SOPT4_TPM2CLKSEL_SHIFT, SIM_SOPT4_TPM2CLKSEL_WIDTH))

/*! @brief Set the TPM2CLKSEL field to a new value. */
#define SIM_WR_SOPT4_TPM2CLKSEL(base, value) (SIM_RMW_SOPT4(base,
SIM_SOPT4_TPM2CLKSEL_MASK, SIM_SOPT4_TPM2CLKSEL(value)))
#define SIM_BWR_SOPT4_TPM2CLKSEL(base, value)
(BME_BFI32(&SIM_SOPT4_REG(base), ((uint32_t)(value) <<
SIM_SOPT4_TPM2CLKSEL_SHIFT), SIM_SOPT4_TPM2CLKSEL_SHIFT,
SIM_SOPT4_TPM2CLKSEL_WIDTH))
/*}@*/ */

*****  

*****  

* SIM_SOPT5 - System Options Register 5

```

```
*****
**** */

/*!
 * @brief SIM_SOPT5 - System Options Register 5 (RW)
 *
 * Reset value: 0x00000000U
 */
/*!!
 * @name Constants and macros for entire SIM_SOPT5 register
 */
/*@{*/
#define SIM_RD_SOPT5(base)          (SIM_SOPT5_REG(base))
#define SIM_WR_SOPT5(base, value)   (SIM_SOPT5_REG(base) = (value))
#define SIM_RMW_SOPT5(base, mask, value) (SIM_WR_SOPT5(base,
(SIM_RD_SOPT5(base) & ~mask) | (value)))
#define SIM_SET_SOPT5(base, value)  (BME_OR32(&SIM_SOPT5_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT5(base, value)  (BME_AND32(&SIM_SOPT5_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT5(base, value)  (BME_XOR32(&SIM_SOPT5_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual SIM_SOPT5 bitfields
 */
/*!
 * @name Register SIM_SOPT5, field UART0TXSRC[1:0] (RW)
 *
 * Selects the source for the UART0 transmit data.
 *
 * Values:
 * - 0b00 - UART0_TX pin
 * - 0b01 - UART0_TX pin modulated with TPM1 channel 0 output
 * - 0b10 - UART0_TX pin modulated with TPM2 channel 0 output
 * - 0b11 - Reserved
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT5_UART0TXSRC field. */
#define SIM_RD_SOPT5_UART0TXSRC(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART0TXSRC_MASK) >> SIM_SOPT5_UART0TXSRC_SHIFT)
#define SIM_BRD_SOPT5_UART0TXSRC(base) (BME_UBFX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART0TXSRC_SHIFT, SIM_SOPT5_UART0TXSRC_WIDTH))

/*! @brief Set the UART0TXSRC field to a new value. */
#define SIM_WR_SOPT5_UART0TXSRC(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART0TXSRC_MASK, SIM_SOPT5_UART0TXSRC(value)))
#define SIM_BWR_SOPT5_UART0TXSRC(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART0TXSRC_SHIFT), SIM_SOPT5_UART0TXSRC_SHIFT,
SIM_SOPT5_UART0TXSRC_WIDTH))
```

```

/*@}*/

/*
 * @name Register SIM_SOPT5, field UART0RXSRC[2] (RW)
 *
 * Selects the source for the UART0 receive data.
 *
 * Values:
 * - 0b0 - UART0_RX pin
 * - 0b1 - CMP0 output
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT5_UART0RXSRC field. */
#define SIM_RD_SOPT5_UART0RXSRC(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART0RXSRC_MASK) >> SIM_SOPT5_UART0RXSRC_SHIFT)
#define SIM_BRD_SOPT5_UART0RXSRC(base) (BME_UBX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART0RXSRC_SHIFT, SIM_SOPT5_UART0RXSRC_WIDTH))

/*! @brief Set the UART0RXSRC field to a new value. */
#define SIM_WR_SOPT5_UART0RXSRC(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART0RXSRC_MASK, SIM_SOPT5_UART0RXSRC(value)))
#define SIM_BWR_SOPT5_UART0RXSRC(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART0RXSRC_SHIFT), SIM_SOPT5_UART0RXSRC_SHIFT,
SIM_SOPT5_UART0RXSRC_WIDTH))
/*@}/

/*
 * @name Register SIM_SOPT5, field UART1TXSRC[5:4] (RW)
 *
 * Selects the source for the UART1 transmit data.
 *
 * Values:
 * - 0b00 - UART1_TX pin
 * - 0b01 - UART1_TX pin modulated with TPM1 channel 0 output
 * - 0b10 - UART1_TX pin modulated with TPM2 channel 0 output
 * - 0b11 - Reserved
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT5_UART1TXSRC field. */
#define SIM_RD_SOPT5_UART1TXSRC(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART1TXSRC_MASK) >> SIM_SOPT5_UART1TXSRC_SHIFT)
#define SIM_BRD_SOPT5_UART1TXSRC(base) (BME_UBX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART1TXSRC_SHIFT, SIM_SOPT5_UART1TXSRC_WIDTH))

/*! @brief Set the UART1TXSRC field to a new value. */
#define SIM_WR_SOPT5_UART1TXSRC(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART1TXSRC_MASK, SIM_SOPT5_UART1TXSRC(value)))
#define SIM_BWR_SOPT5_UART1TXSRC(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART1TXSRC_SHIFT), SIM_SOPT5_UART1TXSRC_SHIFT,
SIM_SOPT5_UART1TXSRC_WIDTH))
/*@}/

```

```

/*!
 * @name Register SIM_SOPT5, field UART1RXSRC[6] (RW)
 *
 * Selects the source for the UART1 receive data.
 *
 * Values:
 * - 0b0 - UART1_RX pin
 * - 0b1 - CMP0 output
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT5_UART1RXSRC field. */
#define SIM_RD_SOPT5_UART1RXSRC(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART1RXSRC_MASK) >> SIM_SOPT5_UART1RXSRC_SHIFT)
#define SIM_BRD_SOPT5_UART1RXSRC(base) (BME_UBX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART1RXSRC_SHIFT, SIM_SOPT5_UART1RXSRC_WIDTH))

/*! @brief Set the UART1RXSRC field to a new value. */
#define SIM_WR_SOPT5_UART1RXSRC(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART1RXSRC_MASK, SIM_SOPT5_UART1RXSRC(value)))
#define SIM_BWR_SOPT5_UART1RXSRC(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART1RXSRC_SHIFT), SIM_SOPT5_UART1RXSRC_SHIFT,
SIM_SOPT5_UART1RXSRC_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register SIM_SOPT5, field UART0ODE[16] (RW)
 *
 * Values:
 * - 0b0 - Open drain is disabled on UART0
 * - 0b1 - Open drain is enabled on UART0
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT5_UART0ODE field. */
#define SIM_RD_SOPT5_UART0ODE(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART0ODE_MASK) >> SIM_SOPT5_UART0ODE_SHIFT)
#define SIM_BRD_SOPT5_UART0ODE(base) (BME_UBX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART0ODE_SHIFT, SIM_SOPT5_UART0ODE_WIDTH))

/*! @brief Set the UART0ODE field to a new value. */
#define SIM_WR_SOPT5_UART0ODE(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART0ODE_MASK, SIM_SOPT5_UART0ODE(value)))
#define SIM_BWR_SOPT5_UART0ODE(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART0ODE_SHIFT), SIM_SOPT5_UART0ODE_SHIFT,
SIM_SOPT5_UART0ODE_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register SIM_SOPT5, field UART1ODE[17] (RW)
 *
 * Values:
 * - 0b0 - Open drain is disabled on UART1
 * - 0b1 - Open drain is enabled on UART1
*/

```

```

/*
/*@{*/
/*! @brief Read current value of the SIM_SOPT5_UART1ODE field. */
#define SIM_RD_SOPT5_UART1ODE(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART1ODE_MASK) >> SIM_SOPT5_UART1ODE_SHIFT)
#define SIM_BRD_SOPT5_UART1ODE(base) (BME_UBFX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART1ODE_SHIFT, SIM_SOPT5_UART1ODE_WIDTH))

/*! @brief Set the UART1ODE field to a new value. */
#define SIM_WR_SOPT5_UART1ODE(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART1ODE_MASK, SIM_SOPT5_UART1ODE(value)))
#define SIM_BWR_SOPT5_UART1ODE(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART1ODE_SHIFT), SIM_SOPT5_UART1ODE_SHIFT,
SIM_SOPT5_UART1ODE_WIDTH))
/*@}*/

/*
 * @name Register SIM_SOPT5, field UART2ODE[18] (RW)
 *
 * Values:
 * - 0b0 - Open drain is disabled on UART2
 * - 0b1 - Open drain is enabled on UART2
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT5_UART2ODE field. */
#define SIM_RD_SOPT5_UART2ODE(base) ((SIM_SOPT5_REG(base) &
SIM_SOPT5_UART2ODE_MASK) >> SIM_SOPT5_UART2ODE_SHIFT)
#define SIM_BRD_SOPT5_UART2ODE(base) (BME_UBFX32(&SIM_SOPT5_REG(base),
SIM_SOPT5_UART2ODE_SHIFT, SIM_SOPT5_UART2ODE_WIDTH))

/*! @brief Set the UART2ODE field to a new value. */
#define SIM_WR_SOPT5_UART2ODE(base, value) (SIM_RMW_SOPT5(base,
SIM_SOPT5_UART2ODE_MASK, SIM_SOPT5_UART2ODE(value)))
#define SIM_BWR_SOPT5_UART2ODE(base, value)
(BME_BFI32(&SIM_SOPT5_REG(base), ((uint32_t)(value) <<
SIM_SOPT5_UART2ODE_SHIFT), SIM_SOPT5_UART2ODE_SHIFT,
SIM_SOPT5_UART2ODE_WIDTH))
/*@*/

*****  

*****  

* SIM_SOPT7 - System Options Register 7  

*****  

*****/  

  

/*!  

* @brief SIM_SOPT7 - System Options Register 7 (RW)
*  

* Reset value: 0x00000000U
*/
/*!  

* @name Constants and macros for entire SIM_SOPT7 register

```

```

/*
/*@{*/
#define SIM_RD_SOPT7(base)      (SIM_SOPT7_REG(base))
#define SIM_WR_SOPT7(base, value) (SIM_SOPT7_REG(base) = (value))
#define SIM_RMW_SOPT7(base, mask, value) (SIM_WR_SOPT7(base,
(SIM_RD_SOPT7(base) & ~mask) | (value)))
#define SIM_SET_SOPT7(base, value) (BME_OR32(&SIM_SOPT7_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SOPT7(base, value) (BME_AND32(&SIM_SOPT7_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SOPT7(base, value) (BME_XOR32(&SIM_SOPT7_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual SIM_SOPT7 bitfields
 */

/*!
 * @name Register SIM_SOPT7, field ADC0TRGSEL[3:0] (RW)
 *
 * Selects the ADC0 trigger source when alternative triggers are
functional in
 * stop and VLPS modes. .
 *
 * Values:
 * - 0b0000 - External trigger pin input (EXTRG_IN)
 * - 0b0001 - CMP0 output
 * - 0b0010 - Reserved
 * - 0b0011 - Reserved
 * - 0b0100 - PIT trigger 0
 * - 0b0101 - PIT trigger 1
 * - 0b0110 - Reserved
 * - 0b0111 - Reserved
 * - 0b1000 - TPM0 overflow
 * - 0b1001 - TPM1 overflow
 * - 0b1010 - TPM2 overflow
 * - 0b1011 - Reserved
 * - 0b1100 - RTC alarm
 * - 0b1101 - RTC seconds
 * - 0b1110 - LPTMR0 trigger
 * - 0b1111 - Reserved
 */
/*@*/
/*! @brief Read current value of the SIM_SOPT7_ADC0TRGSEL field. */
#define SIM_RD_SOPT7_ADC0TRGSEL(base) ((SIM_SOPT7_REG(base) &
SIM_SOPT7_ADC0TRGSEL_MASK) >> SIM_SOPT7_ADC0TRGSEL_SHIFT)
#define SIM_BRD_SOPT7_ADC0TRGSEL(base) (BME_UBFX32(&SIM_SOPT7_REG(base),
SIM_SOPT7_ADC0TRGSEL_SHIFT, SIM_SOPT7_ADC0TRGSEL_WIDTH))

/*! @brief Set the ADC0TRGSEL field to a new value. */
#define SIM_WR_SOPT7_ADC0TRGSEL(base, value) (SIM_RMW_SOPT7(base,
SIM_SOPT7_ADC0TRGSEL_MASK, SIM_SOPT7_ADC0TRGSEL(value)))

```

```

#define SIM_BWR_SOPT7_ADC0TRGSEL(base, value)
(BME_BFI32(&SIM_SOPT7_REG(base), ((uint32_t)(value) <<
SIM_SOPT7_ADC0TRGSEL_SHIFT), SIM_SOPT7_ADC0TRGSEL_SHIFT,
SIM_SOPT7_ADC0TRGSEL_WIDTH))
/*@}*/

/*
 * @name Register SIM_SOPT7, field ADC0PRETRGSEL[4] (RW)
 *
 * Selects the ADC0 pre-trigger source when alternative triggers are
enabled
 * through ADC0ALTTRGEN.
 *
 * Values:
 * - 0b0 - Pre-trigger A
 * - 0b1 - Pre-trigger B
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT7_ADC0PRETRGSEL field. */
#define SIM_RD_SOPT7_ADC0PRETRGSEL(base) ((SIM_SOPT7_REG(base) &
SIM_SOPT7_ADC0PRETRGSEL_MASK) >> SIM_SOPT7_ADC0PRETRGSEL_SHIFT)
#define SIM_BRD_SOPT7_ADC0PRETRGSEL(base)
(BME_UBFX32(&SIM_SOPT7_REG(base), SIM_SOPT7_ADC0PRETRGSEL_SHIFT,
SIM_SOPT7_ADC0PRETRGSEL_WIDTH))

/*! @brief Set the ADC0PRETRGSEL field to a new value. */
#define SIM_WR_SOPT7_ADC0PRETRGSEL(base, value) (SIM_RMW_SOPT7(base,
SIM_SOPT7_ADC0PRETRGSEL_MASK, SIM_SOPT7_ADC0PRETRGSEL(value)))
#define SIM_BWR_SOPT7_ADC0PRETRGSEL(base, value)
(BME_BFI32(&SIM_SOPT7_REG(base), ((uint32_t)(value) <<
SIM_SOPT7_ADC0PRETRGSEL_SHIFT), SIM_SOPT7_ADC0PRETRGSEL_SHIFT,
SIM_SOPT7_ADC0PRETRGSEL_WIDTH))
/*@}*/

/*
 * @name Register SIM_SOPT7, field ADC0ALTTRGEN[7] (RW)
 *
 * Enable alternative conversion triggers for ADC0.
 *
 * Values:
 * - 0b0 - TPM1 channel 0 (A) and channel 1 (B) triggers selected for
ADC0.
 * - 0b1 - Alternate trigger selected for ADC0.
 */
/*@{*/
/*! @brief Read current value of the SIM_SOPT7_ADC0ALTTRGEN field. */
#define SIM_RD_SOPT7_ADC0ALTTRGEN(base) ((SIM_SOPT7_REG(base) &
SIM_SOPT7_ADC0ALTTRGEN_MASK) >> SIM_SOPT7_ADC0ALTTRGEN_SHIFT)
#define SIM_BRD_SOPT7_ADC0ALTTRGEN(base)
(BME_UBFX32(&SIM_SOPT7_REG(base), SIM_SOPT7_ADC0ALTTRGEN_SHIFT,
SIM_SOPT7_ADC0ALTTRGEN_WIDTH))

/*! @brief Set the ADC0ALTTRGEN field to a new value. */

```

```

#define SIM_WR_SOPT7_ADC0ALTRGEN(base, value) (SIM_RMW_SOPT7(base,
SIM_SOPT7_ADC0ALTRGEN_MASK, SIM_SOPT7_ADC0ALTRGEN(value)))
#define SIM_BWR_SOPT7_ADC0ALTRGEN(base, value)
(BME_BFI32(&SIM_SOPT7_REG(base), ((uint32_t)(value) <<
SIM_SOPT7_ADC0ALTRGEN_SHIFT), SIM_SOPT7_ADC0ALTRGEN_SHIFT,
SIM_SOPT7_ADC0ALTRGEN_WIDTH))
/*@}*/

/***** * SIM_SDID - System Device Identification Register *****/
***** */

/*!
* @brief SIM_SDID - System Device Identification Register (RO)
*
* Reset value: 0x00100480U
*/
/*!
* @name Constants and macros for entire SIM_SDID register
*/
/*@{ */
#define SIM_RD_SDID(base) (SIM_SDID_REG(base))
/*@}*/

/*
* Constants & macros for individual SIM_SDID bitfields
*/

/*!
* @name Register SIM_SDID, field PINID[3:0] (RO)
*
* Specifies the pincount of the device.
*
* Values:
* - 0b0000 - 16-pin
* - 0b0001 - 24-pin
* - 0b0010 - 32-pin
* - 0b0011 - Reserved
* - 0b0100 - 48-pin
* - 0b0101 - 64-pin
* - 0b0110 - 80-pin
* - 0b0111 - Reserved
* - 0b1000 - 100-pin
* - 0b1001 - Reserved
* - 0b1010 - Reserved
* - 0b1011 - Reserved
* - 0b1100 - Reserved
* - 0b1101 - Reserved
* - 0b1110 - Reserved
* - 0b1111 - Reserved
*/

```

```

/*@{*/
/*! @brief Read current value of the SIM_SDID_PINID field. */
#define SIM_RD_SDID_PINID(base) ((SIM_SDID_REG(base) &
SIM_SDID_PINID_MASK) >> SIM_SDID_PINID_SHIFT)
#define SIM_BRD_SDID_PINID(base) (BME_UBFX32(&SIM_SDID_REG(base),
SIM_SDID_PINID_SHIFT, SIM_SDID_PINID_WIDTH))
/*}@*/



/*!
* @name Register SIM_SDID, field DIEID[11:7] (RO)
*
* Specifies the silicon implementation number for the device.
*/
/*@{*/
/*! @brief Read current value of the SIM_SDID_DIEID field. */
#define SIM_RD_SDID_DIEID(base) ((SIM_SDID_REG(base) &
SIM_SDID_DIEID_MASK) >> SIM_SDID_DIEID_SHIFT)
#define SIM_BRD_SDID_DIEID(base) (BME_UBFX32(&SIM_SDID_REG(base),
SIM_SDID_DIEID_SHIFT, SIM_SDID_DIEID_WIDTH))
/*}@*/



/*!
* @name Register SIM_SDID, field REVID[15:12] (RO)
*
* Specifies the silicon implementation number for the device.
*/
/*@{*/
/*! @brief Read current value of the SIM_SDID_REVID field. */
#define SIM_RD_SDID_REVID(base) ((SIM_SDID_REG(base) &
SIM_SDID_REVID_MASK) >> SIM_SDID_REVID_SHIFT)
#define SIM_BRD_SDID_REVID(base) (BME_UBFX32(&SIM_SDID_REG(base),
SIM_SDID_REVID_SHIFT, SIM_SDID_REVID_WIDTH))
/*}@*/



/*!
* @name Register SIM_SDID, field SRAMSIZE[19:16] (RO)
*
* Specifies the size of the System SRAM
*
* Values:
* - 0b0000 - 0.5 KB
* - 0b0001 - 1 KB
* - 0b0010 - 2 KB
* - 0b0011 - 4 KB
* - 0b0100 - 8 KB
* - 0b0101 - 16 KB
* - 0b0110 - 32 KB
* - 0b0111 - 64 KB
*/
/*@{*/
/*! @brief Read current value of the SIM_SDID_SRAMSIZE field. */
#define SIM_RD_SDID_SRAMSIZE(base) ((SIM_SDID_REG(base) &
SIM_SDID_SRAMSIZE_MASK) >> SIM_SDID_SRAMSIZE_SHIFT)

```

```

#define SIM_BRD_SDID_SRAMSIZE(base)  (BME_UBFX32(&SIM_SDID_REG(base),  

SIM_SDID_SRAMSIZE_SHIFT, SIM_SDID_SRAMSIZE_WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_SDID, field SERIESID[23:20] (RO)  

 *  

 * Specifies the Kinetis family of the device.  

 *  

 * Values:  

 * - 0b0001 - KL family  

 */  

/*@*/  

/*! @brief Read current value of the SIM_SDID_SERIESID field. */  

#define SIM_RD_SDID_SERIESID(base) ((SIM_SDID_REG(base) &  

SIM_SDID_SERIESID_MASK) >> SIM_SDID_SERIESID_SHIFT)  

#define SIM_BRD_SDID_SERIESID(base) (BME_UBFX32(&SIM_SDID_REG(base),  

SIM_SDID_SERIESID_SHIFT, SIM_SDID_SERIESID_WIDTH))  

/*@*/  

/*!  

 * @name Register SIM_SDID, field SUBFAMID[27:24] (RO)  

 *  

 * Specifies the Kinetis sub-family of the device.  

 *  

 * Values:  

 * - 0b0010 - KLx2 Subfamily (low end)  

 * - 0b0100 - KLx4 Subfamily (basic analog)  

 * - 0b0101 - KLx5 Subfamily (advanced analog)  

 * - 0b0110 - KLx6 Subfamily (advanced analog with I2S)  

 */  

/*@*/  

/*! @brief Read current value of the SIM_SDID_SUBFAMID field. */  

#define SIM_RD_SDID_SUBFAMID(base) ((SIM_SDID_REG(base) &  

SIM_SDID_SUBFAMID_MASK) >> SIM_SDID_SUBFAMID_SHIFT)  

#define SIM_BRD_SDID_SUBFAMID(base) (BME_UBFX32(&SIM_SDID_REG(base),  

SIM_SDID_SUBFAMID_SHIFT, SIM_SDID_SUBFAMID_WIDTH))  

/*@*/  

/*!  

 * @name Register SIM_SDID, field FAMID[31:28] (RO)  

 *  

 * Specifies the Kinetis family of the device.  

 *  

 * Values:  

 * - 0b0000 - KL0x Family (low end)  

 * - 0b0001 - KL1x Family (basic)  

 * - 0b0010 - KL2x Family (USB)  

 * - 0b0011 - KL3x Family (Segment LCD)  

 * - 0b0100 - KL4x Family (USB and Segment LCD)  

 */  

/*@*/  

/*! @brief Read current value of the SIM_SDID_FAMID field. */

```

```

#define SIM_RD_SDID_FAMID(base) ((SIM_SDID_REG(base) &
SIM_SDID_FAMID_MASK) >> SIM_SDID_FAMID_SHIFT)
#define SIM_BRD_SDID_FAMID(base) (BME_UBFX32(&SIM_SDID_REG(base),
SIM_SDID_FAMID_SHIFT, SIM_SDID_FAMID_WIDTH))
/*@}*/

/********************* System Clock Gating Control Register 4 *****/
***** */
* SIM_SCGC4 - System Clock Gating Control Register 4
***** */

/*!
* @brief SIM_SCGC4 - System Clock Gating Control Register 4 (RW)
*
* Reset value: 0xF0000030U
*/
/*!
* @name Constants and macros for entire SIM_SCGC4 register
*/
/*@{*/
#define SIM_RD_SCGC4(base) (SIM_SCGC4_REG(base))
#define SIM_WR_SCGC4(base, value) (SIM_SCGC4_REG(base) = (value))
#define SIM_RMW_SCGC4(base, mask, value) (SIM_WR_SCGC4(base,
(SIM_RD_SCGC4(base) & ~mask) | (value)))
#define SIM_SET_SCGC4(base, value) (BME_OR32(&SIM_SCGC4_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SCGC4(base, value) (BME_AND32(&SIM_SCGC4_REG(base),
(uint32_t)(~value)))
#define SIM_TOG_SCGC4(base, value) (BME_XOR32(&SIM_SCGC4_REG(base),
(uint32_t)(value)))
/*@}*/

/* Unified clock gate bit access macros */
#define SIM_SCGC_BIT_REG(base, index) (*((volatile uint32_t
*)&SIM_SCGC4_REG(base) + (((uint32_t)(index) >> 5) - 3U)))
#define SIM_SCGC_BIT_SHIFT(index) ((uint32_t)(index) & ((1U <<
5) - 1U))
#define SIM_RD_SCGC_BIT(base, index) (SIM_SCGC_BIT_REG((base),
(index)) & (1U << SIM_SCGC_BIT_SHIFT(index)))
#define SIM_BRD_SCGC_BIT(base, index)
(BME_UBFX32(&SIM_SCGC_BIT_REG((base)), (index)),
SIM_SCGC_BIT_SHIFT(index), 1))
#define SIM_WR_SCGC_BIT(base, index, value) (SIM_SCGC_BIT_REG((base),
(index)) = (SIM_SCGC_BIT_REG((base), (index)) & ~((1U <<
SIM_SCGC_BIT_SHIFT(index))) | ((uint32_t)(value) <<
SIM_SCGC_BIT_SHIFT(index)))
#define SIM_BWR_SCGC_BIT(base, index, value)
(BME_BFI32(&SIM_SCGC_BIT_REG((base)), (index)), ((uint32_t)(value) <<
SIM_SCGC_BIT_SHIFT(index)), SIM_SCGC_BIT_SHIFT(index), 1))

/*
* Constants & macros for individual SIM_SCGC4 bitfields

```

```

*/
/*!
 * @name Register SIM_SCGC4, field I2C0[6] (RW)
 *
 * This bit controls the clock gate to the I2C0 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{ */
/*! @brief Read current value of the SIM_SCGC4_I2C0 field. */
#define SIM_RD_SCGC4_I2C0(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_I2C0_MASK) >> SIM_SCGC4_I2C0_SHIFT)
#define SIM_BRD_SCGC4_I2C0(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_I2C0_SHIFT, SIM_SCGC4_I2C0_WIDTH))

/*! @brief Set the I2C0 field to a new value. */
#define SIM_WR_SCGC4_I2C0(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_I2C0_MASK, SIM_SCGC4_I2C0(value)))
#define SIM_BWR_SCGC4_I2C0(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
(uint32_t)(value) << SIM_SCGC4_I2C0_SHIFT), SIM_SCGC4_I2C0_SHIFT,
SIM_SCGC4_I2C0_WIDTH)
/*}@*/ */

/*
 * @name Register SIM_SCGC4, field I2C1[7] (RW)
 *
 * This bit controls the clock gate to the I2C1 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{ */
/*! @brief Read current value of the SIM_SCGC4_I2C1 field. */
#define SIM_RD_SCGC4_I2C1(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_I2C1_MASK) >> SIM_SCGC4_I2C1_SHIFT)
#define SIM_BRD_SCGC4_I2C1(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_I2C1_SHIFT, SIM_SCGC4_I2C1_WIDTH))

/*! @brief Set the I2C1 field to a new value. */
#define SIM_WR_SCGC4_I2C1(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_I2C1_MASK, SIM_SCGC4_I2C1(value)))
#define SIM_BWR_SCGC4_I2C1(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
(uint32_t)(value) << SIM_SCGC4_I2C1_SHIFT), SIM_SCGC4_I2C1_SHIFT,
SIM_SCGC4_I2C1_WIDTH)
/*}@*/ */

/*
 * @name Register SIM_SCGC4, field UART0[10] (RW)
 *
 * This bit controls the clock gate to the UART0 module.

```

```

/*
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_UART0 field. */
#define SIM_RD_SCGC4_UART0(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_UART0_MASK) >> SIM_SCGC4_UART0_SHIFT)
#define SIM_BRD_SCGC4_UART0(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_UART0_SHIFT, SIM_SCGC4_UART0_WIDTH))

/*! @brief Set the UART0 field to a new value. */
#define SIM_WR_SCGC4_UART0(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_UART0_MASK, SIM_SCGC4_UART0(value)))
#define SIM_BWR_SCGC4_UART0(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
(uint32_t)(value) << SIM_SCGC4_UART0_SHIFT), SIM_SCGC4_UART0_SHIFT,
SIM_SCGC4_UART0_WIDTH)
/*}@*/ */

/*!
 * @name Register SIM_SCGC4, field UART1[11] (RW)
 *
 * This bit controls the clock gate to the UART1 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_UART1 field. */
#define SIM_RD_SCGC4_UART1(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_UART1_MASK) >> SIM_SCGC4_UART1_SHIFT)
#define SIM_BRD_SCGC4_UART1(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_UART1_SHIFT, SIM_SCGC4_UART1_WIDTH))

/*! @brief Set the UART1 field to a new value. */
#define SIM_WR_SCGC4_UART1(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_UART1_MASK, SIM_SCGC4_UART1(value)))
#define SIM_BWR_SCGC4_UART1(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
(uint32_t)(value) << SIM_SCGC4_UART1_SHIFT), SIM_SCGC4_UART1_SHIFT,
SIM_SCGC4_UART1_WIDTH)
/*}@*/ */

/*!
 * @name Register SIM_SCGC4, field UART2[12] (RW)
 *
 * This bit controls the clock gate to the UART2 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/

```

```

/*! @brief Read current value of the SIM_SCGC4_UART2 field. */
#define SIM_RD_SCGC4_UART2(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_UART2_MASK) >> SIM_SCGC4_UART2_SHIFT)
#define SIM_BRD_SCGC4_UART2(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_UART2_SHIFT, SIM_SCGC4_UART2_WIDTH))

/*! @brief Set the UART2 field to a new value. */
#define SIM_WR_SCGC4_UART2(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_UART2_MASK, SIM_SCGC4_UART2(value)))
#define SIM_BWR_SCGC4_UART2(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
((uint32_t)(value) << SIM_SCGC4_UART2_SHIFT), SIM_SCGC4_UART2_SHIFT,
SIM_SCGC4_UART2_WIDTH))
/*@}*/

/*!
 * @name Register SIM_SCGC4, field USBOTG[18] (RW)
 *
 * This bit controls the clock gate to the USB module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_USBOTG field. */
#define SIM_RD_SCGC4_USBOTG(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_USBOTG_MASK) >> SIM_SCGC4_USBOTG_SHIFT)
#define SIM_BRD_SCGC4_USBOTG(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_USBOTG_SHIFT, SIM_SCGC4_USBOTG_WIDTH))

/*! @brief Set the USBOTG field to a new value. */
#define SIM_WR_SCGC4_USBOTG(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_USBOTG_MASK, SIM_SCGC4_USBOTG(value)))
#define SIM_BWR_SCGC4_USBOTG(base, value)
(BME_BFI32(&SIM_SCGC4_REG(base), ((uint32_t)(value) <<
SIM_SCGC4_USBOTG_SHIFT), SIM_SCGC4_USBOTG_SHIFT, SIM_SCGC4_USBOTG_WIDTH))
/*@}*/

/*!
 * @name Register SIM_SCGC4, field CMP[19] (RW)
 *
 * This bit controls the clock gate to the comparator module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_CMP field. */
#define SIM_RD_SCGC4_CMP(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_CMP_MASK) >> SIM_SCGC4_CMP_SHIFT)
#define SIM_BRD_SCGC4_CMP(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_CMP_SHIFT, SIM_SCGC4_CMP_WIDTH))

```

```

/*! @brief Set the CMP field to a new value. */
#define SIM_WR_SCGC4_CMP(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_CMP_MASK, SIM_SCGC4_CMP(value)))
#define SIM_BWR_SCGC4_CMP(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
((uint32_t)(value) << SIM_SCGC4_CMP_SHIFT), SIM_SCGC4_CMP_SHIFT,
SIM_SCGC4_CMP_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SCGC4, field SPI0[22] (RW)
 *
 * This bit controls the clock gate to the SPI0 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_SPI0 field. */
#define SIM_RD_SCGC4_SPI0(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_SPI0_MASK) >> SIM_SCGC4_SPI0_SHIFT)
#define SIM_BRD_SCGC4_SPI0(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_SPI0_SHIFT, SIM_SCGC4_SPI0_WIDTH))

/*! @brief Set the SPI0 field to a new value. */
#define SIM_WR_SCGC4_SPI0(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_SPI0_MASK, SIM_SCGC4_SPI0(value)))
#define SIM_BWR_SCGC4_SPI0(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
((uint32_t)(value) << SIM_SCGC4_SPI0_SHIFT), SIM_SCGC4_SPI0_SHIFT,
SIM_SCGC4_SPI0_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_SCGC4, field SPI1[23] (RW)
 *
 * This bit controls the clock gate to the SPI1 module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC4_SPI1 field. */
#define SIM_RD_SCGC4_SPI1(base) ((SIM_SCGC4_REG(base) &
SIM_SCGC4_SPI1_MASK) >> SIM_SCGC4_SPI1_SHIFT)
#define SIM_BRD_SCGC4_SPI1(base) (BME_UBFX32(&SIM_SCGC4_REG(base),
SIM_SCGC4_SPI1_SHIFT, SIM_SCGC4_SPI1_WIDTH))

/*! @brief Set the SPI1 field to a new value. */
#define SIM_WR_SCGC4_SPI1(base, value) (SIM_RMW_SCGC4(base,
SIM_SCGC4_SPI1_MASK, SIM_SCGC4_SPI1(value)))
#define SIM_BWR_SCGC4_SPI1(base, value) (BME_BFI32(&SIM_SCGC4_REG(base),
((uint32_t)(value) << SIM_SCGC4_SPI1_SHIFT), SIM_SCGC4_SPI1_SHIFT,
SIM_SCGC4_SPI1_WIDTH))

```

```

/*@}*/

/*****
 * SIM_SCGC5 - System Clock Gating Control Register 5
 *****/
/* */

/*!
 * @brief SIM_SCGC5 - System Clock Gating Control Register 5 (RW)
 *
 * Reset value: 0x00000180U
 */
/*!
 * @name Constants and macros for entire SIM_SCGC5 register
 */
/*@{*/
#define SIM_RD_SCGC5(base)          (SIM_SCGC5_REG(base))
#define SIM_WR_SCGC5(base, value)   (SIM_SCGC5_REG(base) = (value))
#define SIM_RMW_SCGC5(base, mask, value) (SIM_WR_SCGC5(base,
(SIM_RD_SCGC5(base) & ~mask) | (value)))
#define SIM_SET_SCGC5(base, value)  (BME_OR32(&SIM_SCGC5_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SCGC5(base, value)  (BME_AND32(&SIM_SCGC5_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SCGC5(base, value)  (BME_XOR32(&SIM_SCGC5_REG(base),
(uint32_t)(value)))
/*@}*/
/* */

/*!
 * Constants & macros for individual SIM_SCGC5 bitfields
 */
/*!

 * @name Register SIM_SCGC5, field LPTMR[0] (RW)
 *
 * This bit controls software access to the Low Power Timer module.
 *
 * Values:
 * - 0b0 - Access disabled
 * - 0b1 - Access enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC5_LPTMR field. */
#define SIM_RD_SCGC5_LPTMR(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_LPTMR_MASK) >> SIM_SCGC5_LPTMR_SHIFT)
#define SIM_BRD_SCGC5_LPTMR(base) (BME_UBFX32(&SIM_SCGC5_REG(base),
SIM_SCGC5_LPTMR_SHIFT, SIM_SCGC5_LPTMR_WIDTH))

/*! @brief Set the LPTMR field to a new value. */
#define SIM_WR_SCGC5_LPTMR(base, value) (SIM_RMW_SCGC5(base,
SIM_SCGC5_LPTMR_MASK, SIM_SCGC5_LPTMR(value)))

```

```

#define SIM_BWR_SCGC5_LPTMR(base, value) (BME_BFI32(&SIM_SCGC5_REG(base), \
((uint32_t)(value) << SIM_SCGC5_LPTMR_SHIFT), SIM_SCGC5_LPTMR_SHIFT, \
SIM_SCGC5_LPTMR_WIDTH))
/*@}*/

/*!
 * @name Register SIM_SCGC5, field TSI[5] (RW)
 *
 * This bit controls software access to the TSI module.
 *
 * Values:
 * - 0b0 - Access disabled
 * - 0b1 - Access enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC5_TSI field. */
#define SIM_RD_SCGC5_TSI(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_TSI_MASK) >> SIM_SCGC5_TSI_SHIFT)
#define SIM_BRD_SCGC5_TSI(base) (BME_UBFX32(&SIM_SCGC5_REG(base), \
SIM_SCGC5_TSI_SHIFT, SIM_SCGC5_TSI_WIDTH))

/*! @brief Set the TSI field to a new value. */
#define SIM_WR_SCGC5_TSI(base, value) (SIM_RMW_SCGC5(base, \
SIM_SCGC5_TSI_MASK, SIM_SCGC5_TSI(value)))
#define SIM_BWR_SCGC5_TSI(base, value) (BME_BFI32(&SIM_SCGC5_REG(base), \
((uint32_t)(value) << SIM_SCGC5_TSI_SHIFT), SIM_SCGC5_TSI_SHIFT, \
SIM_SCGC5_TSI_WIDTH))
/*@}*/

/*!
 * @name Register SIM_SCGC5, field PORTA[9] (RW)
 *
 * This bit controls the clock gate to the Port A module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC5_PORTA field. */
#define SIM_RD_SCGC5_PORTA(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_PORTA_MASK) >> SIM_SCGC5_PORTA_SHIFT)
#define SIM_BRD_SCGC5_PORTA(base) (BME_UBFX32(&SIM_SCGC5_REG(base), \
SIM_SCGC5_PORTA_SHIFT, SIM_SCGC5_PORTA_WIDTH))

/*! @brief Set the PORTA field to a new value. */
#define SIM_WR_SCGC5_PORTA(base, value) (SIM_RMW_SCGC5(base, \
SIM_SCGC5_PORTA_MASK, SIM_SCGC5_PORTA(value)))
#define SIM_BWR_SCGC5_PORTA(base, value) (BME_BFI32(&SIM_SCGC5_REG(base), \
((uint32_t)(value) << SIM_SCGC5_PORTA_SHIFT), SIM_SCGC5_PORTA_SHIFT, \
SIM_SCGC5_PORTA_WIDTH))
/*@}*/

/*

```

```

* @name Register SIM_SCGC5, field PORTB[10] (RW)
*
* This bit controls the clock gate to the Port B module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{/ */
/*! @brief Read current value of the SIM_SCGC5_PORTB field. */
#define SIM_RD_SCGC5_PORTB(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_PORTB_MASK) >> SIM_SCGC5_PORTB_SHIFT)
#define SIM_BRD_SCGC5_PORTB(base) (BME_UBFX32(&SIM_SCGC5_REG(base),
SIM_SCGC5_PORTB_SHIFT, SIM_SCGC5_PORTB_WIDTH))

/*! @brief Set the PORTB field to a new value. */
#define SIM_WR_SCGC5_PORTB(base, value) (SIM_RMW_SCGC5(base,
SIM_SCGC5_PORTB_MASK, SIM_SCGC5_PORTB(value)))
#define SIM_BWR_SCGC5_PORTB(base, value) (BME_BFI32(&SIM_SCGC5_REG(base),
((uint32_t)(value) << SIM_SCGC5_PORTB_SHIFT), SIM_SCGC5_PORTB_SHIFT,
SIM_SCGC5_PORTB_WIDTH))
/*@{/ */

/*!
* @name Register SIM_SCGC5, field PORTC[11] (RW)
*
* This bit controls the clock gate to the Port C module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{/ */
/*! @brief Read current value of the SIM_SCGC5_PORTC field. */
#define SIM_RD_SCGC5_PORTC(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_PORTC_MASK) >> SIM_SCGC5_PORTC_SHIFT)
#define SIM_BRD_SCGC5_PORTC(base) (BME_UBFX32(&SIM_SCGC5_REG(base),
SIM_SCGC5_PORTC_SHIFT, SIM_SCGC5_PORTC_WIDTH))

/*! @brief Set the PORTC field to a new value. */
#define SIM_WR_SCGC5_PORTC(base, value) (SIM_RMW_SCGC5(base,
SIM_SCGC5_PORTC_MASK, SIM_SCGC5_PORTC(value)))
#define SIM_BWR_SCGC5_PORTC(base, value) (BME_BFI32(&SIM_SCGC5_REG(base),
((uint32_t)(value) << SIM_SCGC5_PORTC_SHIFT), SIM_SCGC5_PORTC_SHIFT,
SIM_SCGC5_PORTC_WIDTH))
/*@{/ */

/*!
* @name Register SIM_SCGC5, field PORTD[12] (RW)
*
* This bit controls the clock gate to the Port D module.
*
* Values:
* - 0b0 - Clock disabled

```

```

* - 0b1 - Clock enabled
*/
/*@{ */
/*! @brief Read current value of the SIM_SCGC5_PORTD field. */
#define SIM_RD_SCGC5_PORTD(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_PORTD_MASK) >> SIM_SCGC5_PORTD_SHIFT)
#define SIM_BRD_SCGC5_PORTD(base) (BME_UBFX32(&SIM_SCGC5_REG(base),
SIM_SCGC5_PORTD_SHIFT, SIM_SCGC5_PORTD_WIDTH))

/*! @brief Set the PORTD field to a new value. */
#define SIM_WR_SCGC5_PORTD(base, value) (SIM_RMW_SCGC5(base,
SIM_SCGC5_PORTD_MASK, SIM_SCGC5_PORTD(value)))
#define SIM_BWR_SCGC5_PORTD(base, value) (BME_BFI32(&SIM_SCGC5_REG(base),
((uint32_t)(value) << SIM_SCGC5_PORTD_SHIFT), SIM_SCGC5_PORTD_SHIFT,
SIM_SCGC5_PORTD_WIDTH))
/*}@ */

/*!
* @name Register SIM_SCGC5, field PORTE[13] (RW)
*
* This bit controls the clock gate to the Port E module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{ */
/*! @brief Read current value of the SIM_SCGC5_PORTE field. */
#define SIM_RD_SCGC5_PORTE(base) ((SIM_SCGC5_REG(base) &
SIM_SCGC5_PORTE_MASK) >> SIM_SCGC5_PORTE_SHIFT)
#define SIM_BRD_SCGC5_PORTE(base) (BME_UBFX32(&SIM_SCGC5_REG(base),
SIM_SCGC5_PORTE_SHIFT, SIM_SCGC5_PORTE_WIDTH))

/*! @brief Set the PORTE field to a new value. */
#define SIM_WR_SCGC5_PORTE(base, value) (SIM_RMW_SCGC5(base,
SIM_SCGC5_PORTE_MASK, SIM_SCGC5_PORTE(value)))
#define SIM_BWR_SCGC5_PORTE(base, value) (BME_BFI32(&SIM_SCGC5_REG(base),
((uint32_t)(value) << SIM_SCGC5_PORTE_SHIFT), SIM_SCGC5_PORTE_SHIFT,
SIM_SCGC5_PORTE_WIDTH))
/*}@ */

*****  

*****  

* SIM_SCGC6 - System Clock Gating Control Register 6  

*****  

*****/  

/*!  

* @brief SIM_SCGC6 - System Clock Gating Control Register 6 (RW)
*
* Reset value: 0x00000001U
*/
/*!
```

```

 * @name Constants and macros for entire SIM_SCGC6 register
 */
/*@{*/
#define SIM_RD_SCGC6(base)          (SIM_SCGC6_REG(base))
#define SIM_WR_SCGC6(base, value)   (SIM_SCGC6_REG(base) = (value))
#define SIM_RMW_SCGC6(base, mask, value) (SIM_WR_SCGC6(base,
(SIM_RD_SCGC6(base) & ~mask)) | (value)))
#define SIM_SET_SCGC6(base, value)  (BME_OR32(&SIM_SCGC6_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SCGC6(base, value)  (BME_AND32(&SIM_SCGC6_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SCGC6(base, value)  (BME_XOR32(&SIM_SCGC6_REG(base),
(uint32_t)(value)))
/*}@*/
```

```

/*
 * Constants & macros for individual SIM_SCGC6 bitfields
 */

/*!
 * @name Register SIM_SCGC6, field FTF[0] (RW)
 *
 * This bit controls the clock gate to the flash memory. Flash reads are
still
 * supported while the flash memory is clock gated, but entry into low
power modes
 * is blocked.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC6_FTF field. */
#define SIM_RD_SCGC6_FTF(base) ((SIM_SCGC6_REG(base) &
SIM_SCGC6_FTF_MASK) >> SIM_SCGC6_FTF_SHIFT)
#define SIM_BRD_SCGC6_FTF(base) (BME_UBFX32(&SIM_SCGC6_REG(base),
SIM_SCGC6_FTF_SHIFT, SIM_SCGC6_FTF_WIDTH))

/*! @brief Set the FTF field to a new value. */
#define SIM_WR_SCGC6_FTF(base, value) (SIM_RMW_SCGC6(base,
SIM_SCGC6_FTF_MASK, SIM_SCGC6_FTF(value)))
#define SIM_BWR_SCGC6_FTF(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),
((uint32_t)(value) << SIM_SCGC6_FTF_SHIFT), SIM_SCGC6_FTF_SHIFT,
SIM_SCGC6_FTF_WIDTH))
/*}@*/
```

```

/*!
 * @name Register SIM_SCGC6, field DMAMUX[1] (RW)
 *
 * This bit controls the clock gate to the DMA Mux module.
 *
 * Values:
 * - 0b0 - Clock disabled
```

```

* - 0b1 - Clock enabled
*/
/*@{ */
/*! @brief Read current value of the SIM_SCGC6_DMAMUX field. */
#define SIM_RD_SCGC6_DMAMUX(base) ((SIM_SCGC6_REG(base) &
SIM_SCGC6_DMAMUX_MASK) >> SIM_SCGC6_DMAMUX_SHIFT)
#define SIM_BRD_SCGC6_DMAMUX(base) (BME_UBFX32(&SIM_SCGC6_REG(base),
SIM_SCGC6_DMAMUX_SHIFT, SIM_SCGC6_DMAMUX_WIDTH))

/*! @brief Set the DMAMUX field to a new value. */
#define SIM_WR_SCGC6_DMAMUX(base, value) (SIM_RMW_SCGC6(base,
SIM_SCGC6_DMAMUX_MASK, SIM_SCGC6_DMAMUX(value)))
#define SIM_BWR_SCGC6_DMAMUX(base, value)
(BME_BFI32(&SIM_SCGC6_REG(base), ((uint32_t)(value) <<
SIM_SCGC6_DMAMUX_SHIFT), SIM_SCGC6_DMAMUX_SHIFT, SIM_SCGC6_DMAMUX_WIDTH))
/*@} */

/*!!
* @name Register SIM_SCGC6, field PIT[23] (RW)
*
* This bit controls the clock gate to the PIT module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{ */
/*! @brief Read current value of the SIM_SCGC6_PIT field. */
#define SIM_RD_SCGC6_PIT(base) ((SIM_SCGC6_REG(base) &
SIM_SCGC6_PIT_MASK) >> SIM_SCGC6_PIT_SHIFT)
#define SIM_BRD_SCGC6_PIT(base) (BME_UBFX32(&SIM_SCGC6_REG(base),
SIM_SCGC6_PIT_SHIFT, SIM_SCGC6_PIT_WIDTH))

/*! @brief Set the PIT field to a new value. */
#define SIM_WR_SCGC6_PIT(base, value) (SIM_RMW_SCGC6(base,
SIM_SCGC6_PIT_MASK, SIM_SCGC6_PIT(value)))
#define SIM_BWR_SCGC6_PIT(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),
((uint32_t)(value) << SIM_SCGC6_PIT_SHIFT), SIM_SCGC6_PIT_SHIFT,
SIM_SCGC6_PIT_WIDTH))
/*@} */

/*!!
* @name Register SIM_SCGC6, field TPM0[24] (RW)
*
* This bit controls the clock gate to the TPM0 module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{ */
/*! @brief Read current value of the SIM_SCGC6_TPM0 field. */
#define SIM_RD_SCGC6_TPM0(base) ((SIM_SCGC6_REG(base) &
SIM_SCGC6_TPM0_MASK) >> SIM_SCGC6_TPM0_SHIFT)

```

```

#define SIM_BRD_SCGC6 TPM0(base) (BME_UBFX32(&SIM_SCGC6_REG(base),  

SIM_SCGC6_TPM0_SHIFT, SIM_SCGC6_TPM0_WIDTH))

/*! @brief Set the TPM0 field to a new value. */  

#define SIM_WR_SCGC6_TPM0(base, value) (SIM_RMW_SCGC6(base,  

SIM_SCGC6_TPM0_MASK, SIM_SCGC6_TPM0(value)))  

#define SIM_BWR_SCGC6_TPM0(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),  

(uint32_t)(value) << SIM_SCGC6_TPM0_SHIFT, SIM_SCGC6_TPM0_SHIFT,  

SIM_SCGC6_TPM0_WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_SCGC6, field TPM1[25] (RW)  

 *  

 * This bit controls the clock gate to the TPM1 module.  

 *  

 * Values:  

 * - 0b0 - Clock disabled  

 * - 0b1 - Clock enabled  

 */  

/*@*/  

/*! @brief Read current value of the SIM_SCGC6_TPM1 field. */  

#define SIM_RD_SCGC6_TPM1(base) ((SIM_SCGC6_REG(base) &  

SIM_SCGC6_TPM1_MASK) >> SIM_SCGC6_TPM1_SHIFT)  

#define SIM_BRD_SCGC6_TPM1(base) (BME_UBFX32(&SIM_SCGC6_REG(base),  

SIM_SCGC6_TPM1_SHIFT, SIM_SCGC6_TPM1_WIDTH))

/*! @brief Set the TPM1 field to a new value. */  

#define SIM_WR_SCGC6_TPM1(base, value) (SIM_RMW_SCGC6(base,  

SIM_SCGC6_TPM1_MASK, SIM_SCGC6_TPM1(value)))  

#define SIM_BWR_SCGC6_TPM1(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),  

(uint32_t)(value) << SIM_SCGC6_TPM1_SHIFT, SIM_SCGC6_TPM1_SHIFT,  

SIM_SCGC6_TPM1_WIDTH))  

/*@*/

/*!  

 * @name Register SIM_SCGC6, field TPM2[26] (RW)  

 *  

 * This bit controls the clock gate to the TPM2 module.  

 *  

 * Values:  

 * - 0b0 - Clock disabled  

 * - 0b1 - Clock enabled  

 */  

/*@*/  

/*! @brief Read current value of the SIM_SCGC6_TPM2 field. */  

#define SIM_RD_SCGC6_TPM2(base) ((SIM_SCGC6_REG(base) &  

SIM_SCGC6_TPM2_MASK) >> SIM_SCGC6_TPM2_SHIFT)  

#define SIM_BRD_SCGC6_TPM2(base) (BME_UBFX32(&SIM_SCGC6_REG(base),  

SIM_SCGC6_TPM2_SHIFT, SIM_SCGC6_TPM2_WIDTH))

/*! @brief Set the TPM2 field to a new value. */  

#define SIM_WR_SCGC6_TPM2(base, value) (SIM_RMW_SCGC6(base,  

SIM_SCGC6_TPM2_MASK, SIM_SCGC6_TPM2(value)))

```

```

#define SIM_BWR_SCGC6 TPM2(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),  

((uint32_t)(value) << SIM_SCGC6 TPM2 SHIFT), SIM_SCGC6 TPM2 SHIFT,  

SIM_SCGC6 TPM2 WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_SCGC6, field ADC0[27] (RW)  

 *  

 * This bit controls the clock gate to the ADC0 module.  

 *  

 * Values:  

 * - 0b0 - Clock disabled  

 * - 0b1 - Clock enabled  

 */  

/*@{*/  

/*! @brief Read current value of the SIM_SCGC6_ADC0 field. */  

#define SIM_RD_SCGC6_ADC0(base) ((SIM_SCGC6_REG(base) &  

SIM_SCGC6_ADC0_MASK) >> SIM_SCGC6_ADC0_SHIFT)  

#define SIM_BRD_SCGC6_ADC0(base) (BME_UBFX32(&SIM_SCGC6_REG(base),  

SIM_SCGC6_ADC0_SHIFT, SIM_SCGC6_ADC0_WIDTH))

/*! @brief Set the ADC0 field to a new value. */  

#define SIM_WR_SCGC6_ADC0(base, value) (SIM_RMW_SCGC6(base,  

SIM_SCGC6_ADC0_MASK, SIM_SCGC6_ADC0(value)))  

#define SIM_BWR_SCGC6_ADC0(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),  

((uint32_t)(value) << SIM_SCGC6_ADC0_SHIFT), SIM_SCGC6_ADC0_SHIFT,  

SIM_SCGC6_ADC0_WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_SCGC6, field RTC[29] (RW)  

 *  

 * This bit controls software access and interrupts to the RTC module.  

 *  

 * Values:  

 * - 0b0 - Access and interrupts disabled  

 * - 0b1 - Access and interrupts enabled  

 */  

/*@{*/  

/*! @brief Read current value of the SIM_SCGC6_RTC field. */  

#define SIM_RD_SCGC6_RTC(base) ((SIM_SCGC6_REG(base) &  

SIM_SCGC6_RTC_MASK) >> SIM_SCGC6_RTC_SHIFT)  

#define SIM_BRD_SCGC6_RTC(base) (BME_UBFX32(&SIM_SCGC6_REG(base),  

SIM_SCGC6_RTC_SHIFT, SIM_SCGC6_RTC_WIDTH))

/*! @brief Set the RTC field to a new value. */  

#define SIM_WR_SCGC6_RTC(base, value) (SIM_RMW_SCGC6(base,  

SIM_SCGC6_RTC_MASK, SIM_SCGC6_RTC(value)))  

#define SIM_BWR_SCGC6_RTC(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),  

((uint32_t)(value) << SIM_SCGC6_RTC_SHIFT), SIM_SCGC6_RTC_SHIFT,  

SIM_SCGC6_RTC_WIDTH))  

/*@}*/

/*!

```

```

* @name Register SIM_SCGC6, field DAC0[31] (RW)
*
* This bit controls the clock gate to the DAC0 module.
*
* Values:
* - 0b0 - Clock disabled
* - 0b1 - Clock enabled
*/
/*@{*/
/*! @brief Read current value of the SIM_SCGC6_DAC0 field. */
#define SIM_RD_SCGC6_DAC0(base) ((SIM_SCGC6_REG(base) &
SIM_SCGC6_DAC0_MASK) >> SIM_SCGC6_DAC0_SHIFT)
#define SIM_BRD_SCGC6_DAC0(base) (BME_UBFX32(&SIM_SCGC6_REG(base),
SIM_SCGC6_DAC0_SHIFT, SIM_SCGC6_DAC0_WIDTH))

/*! @brief Set the DAC0 field to a new value. */
#define SIM_WR_SCGC6_DAC0(base, value) (SIM_RMW_SCGC6(base,
SIM_SCGC6_DAC0_MASK, SIM_SCGC6_DAC0(value)))
#define SIM_BWR_SCGC6_DAC0(base, value) (BME_BFI32(&SIM_SCGC6_REG(base),
((uint32_t)(value) << SIM_SCGC6_DAC0_SHIFT), SIM_SCGC6_DAC0_SHIFT,
SIM_SCGC6_DAC0_WIDTH))
/*}@*/



/********************* **** SIM_SCGC7 - System Clock Gating Control Register 7 **** *****************/
***** /


/*!
* @brief SIM_SCGC7 - System Clock Gating Control Register 7 (RW)
*
* Reset value: 0x00000100U
*/
/*!
* @name Constants and macros for entire SIM_SCGC7 register
*/
/*@{*/
#define SIM_RD_SCGC7(base) (SIM_SCGC7_REG(base))
#define SIM_WR_SCGC7(base, value) (SIM_SCGC7_REG(base) = (value))
#define SIM_RMW_SCGC7(base, mask, value) (SIM_WR_SCGC7(base,
(SIM_RD_SCGC7(base) & ~mask)) | (value)))
#define SIM_SET_SCGC7(base, value) (BME_OR32(&SIM_SCGC7_REG(base),
(uint32_t)(value)))
#define SIM_CLR_SCGC7(base, value) (BME_AND32(&SIM_SCGC7_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_SCGC7(base, value) (BME_XOR32(&SIM_SCGC7_REG(base),
(uint32_t)(value)))
/*}@*/



/*
* Constants & macros for individual SIM_SCGC7 bitfields
*/

```

```

/*!
 * @name Register SIM_SCGC7, field DMA[8] (RW)
 *
 * This bit controls the clock gate to the DMA module.
 *
 * Values:
 * - 0b0 - Clock disabled
 * - 0b1 - Clock enabled
 */
/*@{*/
/*! @brief Read current value of the SIM_SCGC7_DMA field. */
#define SIM_RD_SCGC7_DMA(base) ((SIM_SCGC7_REG(base) &
SIM_SCGC7_DMA_MASK) >> SIM_SCGC7_DMA_SHIFT)
#define SIM_BRD_SCGC7_DMA(base) (BME_UBFX32(&SIM_SCGC7_REG(base),
SIM_SCGC7_DMA_SHIFT, SIM_SCGC7_DMA_WIDTH))

/*! @brief Set the DMA field to a new value. */
#define SIM_WR_SCGC7_DMA(base, value) (SIM_RMW_SCGC7(base,
SIM_SCGC7_DMA_MASK, SIM_SCGC7_DMA(value)))
#define SIM_BWR_SCGC7_DMA(base, value) (BME_BFI32(&SIM_SCGC7_REG(base),
((uint32_t)(value) << SIM_SCGC7_DMA_SHIFT), SIM_SCGC7_DMA_SHIFT,
SIM_SCGC7_DMA_WIDTH))
/*}@*/



/*****************
*****/
 * SIM_CLKDIV1 - System Clock Divider Register 1
*****/


/*!
 * @brief SIM_CLKDIV1 - System Clock Divider Register 1 (RW)
 *
 * Reset value: 0x00010000U
 *
 * The CLKDIV1 register cannot be written to when the device is in VLPR
mode.
 * Reset value loaded during System Reset from FTF_FOPT[LPBOOT].
 */
/*!
 * @name Constants and macros for entire SIM_CLKDIV1 register
 */
/*@{*/
#define SIM_RD_CLKDIV1(base) (SIM_CLKDIV1_REG(base))
#define SIM_WR_CLKDIV1(base, value) (SIM_CLKDIV1_REG(base) = (value))
#define SIM_RMW_CLKDIV1(base, mask, value) (SIM_WR_CLKDIV1(base,
(SIM_RD_CLKDIV1(base) & ~mask) | (value)))
#define SIM_SET_CLKDIV1(base, value) (BME_OR32(&SIM_CLKDIV1_REG(base),
(uint32_t)(value)))
#define SIM_CLR_CLKDIV1(base, value) (BME_AND32(&SIM_CLKDIV1_REG(base),
(uint32_t)(~(value))))

```

```

#define SIM_TOG_CLKDIV1(base, value) (BME_XOR32(&SIM_CLKDIV1_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual SIM_CLKDIV1 bitfields
 */

/*!  

 * @name Register SIM_CLKDIV1, field OUTDIV4[18:16] (RW)  

 *  

 * This field sets the divide value for the bus and flash clock and is in  

 * addition to the System clock divide ratio. At the end of reset, it is  

loaded with  

 * 0001 (divide by two).  

 *  

 * Values:  

 * - 0b000 - Divide-by-1.  

 * - 0b001 - Divide-by-2.  

 * - 0b010 - Divide-by-3.  

 * - 0b011 - Divide-by-4.  

 * - 0b100 - Divide-by-5.  

 * - 0b101 - Divide-by-6.  

 * - 0b110 - Divide-by-7.  

 * - 0b111 - Divide-by-8.  

 */
/*@{*/
/*! @brief Read current value of the SIM_CLKDIV1_OUTDIV4 field. */  

#define SIM_RD_CLKDIV1_OUTDIV4(base) ((SIM_CLKDIV1_REG(base) &  

SIM_CLKDIV1_OUTDIV4_MASK) >> SIM_CLKDIV1_OUTDIV4_SHIFT)  

#define SIM_BRD_CLKDIV1_OUTDIV4(base) (BME_UBFX32(&SIM_CLKDIV1_REG(base),  

SIM_CLKDIV1_OUTDIV4_SHIFT, SIM_CLKDIV1_OUTDIV4_WIDTH))  

/*! @brief Set the OUTDIV4 field to a new value. */  

#define SIM_WR_CLKDIV1_OUTDIV4(base, value) (SIM_RMW_CLKDIV1(base,  

SIM_CLKDIV1_OUTDIV4_MASK, SIM_CLKDIV1_OUTDIV4(value)))  

#define SIM_BWR_CLKDIV1_OUTDIV4(base, value)  

(BME_BFI32(&SIM_CLKDIV1_REG(base), ((uint32_t)(value) <<  

SIM_CLKDIV1_OUTDIV4_SHIFT), SIM_CLKDIV1_OUTDIV4_SHIFT,  

SIM_CLKDIV1_OUTDIV4_WIDTH))  

/*@}*/

/*!  

 * @name Register SIM_CLKDIV1, field OUTDIV1[31:28] (RW)  

 *  

 * This field sets the divide value for the core/system clock, as well as  

the  

 * bus/flash clocks. At the end of reset, it is loaded with 0000 (divide  

by one),  

 * 0001 (divide by two), 0011 (divide by four), or 0111 (divide by eight)  

 * depending on the setting of the two FTF_FOPT[LPBOOT] configuration  

bits.  

 *  

 * Values:

```

```

* - 0b0000 - Divide-by-1.
* - 0b0001 - Divide-by-2.
* - 0b0010 - Divide-by-3.
* - 0b0011 - Divide-by-4.
* - 0b0100 - Divide-by-5.
* - 0b0101 - Divide-by-6.
* - 0b0110 - Divide-by-7.
* - 0b0111 - Divide-by-8.
* - 0b1000 - Divide-by-9.
* - 0b1001 - Divide-by-10.
* - 0b1010 - Divide-by-11.
* - 0b1011 - Divide-by-12.
* - 0b1100 - Divide-by-13.
* - 0b1101 - Divide-by-14.
* - 0b1110 - Divide-by-15.
* - 0b1111 - Divide-by-16.
*/
/*@{*/
/*! @brief Read current value of the SIM_CLKDIV1_OUTDIV1 field. */
#define SIM_RD_CLKDIV1_OUTDIV1(base) ((SIM_CLKDIV1_REG(base) &
SIM_CLKDIV1_OUTDIV1_MASK) >> SIM_CLKDIV1_OUTDIV1_SHIFT)
#define SIM_BRD_CLKDIV1_OUTDIV1(base) (BME_UBFX32(&SIM_CLKDIV1_REG(base),
SIM_CLKDIV1_OUTDIV1_SHIFT, SIM_CLKDIV1_OUTDIV1_WIDTH))

/*! @brief Set the OUTDIV1 field to a new value. */
#define SIM_WR_CLKDIV1_OUTDIV1(base, value) (SIM_RMW_CLKDIV1(base,
SIM_CLKDIV1_OUTDIV1_MASK, SIM_CLKDIV1_OUTDIV1(value)))
#define SIM_BWR_CLKDIV1_OUTDIV1(base, value)
(BME_BFI32(&SIM_CLKDIV1_REG(base), ((uint32_t)(value) <<
SIM_CLKDIV1_OUTDIV1_SHIFT), SIM_CLKDIV1_OUTDIV1_SHIFT,
SIM_CLKDIV1_OUTDIV1_WIDTH))
/*}@*/
*****  

* SIM_FCFG1 - Flash Configuration Register 1  

*****  

/*!  

* @brief SIM_FCFG1 - Flash Configuration Register 1 (RW)  

*  

* Reset value: 0x0F000000U  

*/  

/*!  

* @name Constants and macros for entire SIM_FCFG1 register  

*/  

/*@{*/
#define SIM_RD_FCFG1(base) (SIM_FCFG1_REG(base))
#define SIM_WR_FCFG1(base, value) (SIM_FCFG1_REG(base) = (value))
#define SIM_RMW_FCFG1(base, mask, value) (SIM_WR_FCFG1(base,
(SIM_RD_FCFG1(base) & ~mask) | (value)))

```

```

#define SIM_SET_FCFG1(base, value) (BME_OR32(&SIM_FCFG1_REG(base),  

(uint32_t)(value)))  

#define SIM_CLR_FCFG1(base, value) (BME_AND32(&SIM_FCFG1_REG(base),  

(uint32_t)(~(value))))  

#define SIM_TOG_FCFG1(base, value) (BME_XOR32(&SIM_FCFG1_REG(base),  

(uint32_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual SIM_FCFG1 bitfields
 */

/*!
 * @name Register SIM_FCFG1, field FLASHDIS[0] (RW)
 *
 * Flash accesses are disabled (and generate a bus error) and the Flash
memory
 * is placed in a low power state. This bit should not be changed during
VLP
 * modes. Relocate the interrupt vectors out of Flash memory before
disabling the
 * Flash.
 *
 * Values:
 * - 0b0 - Flash is enabled
 * - 0b1 - Flash is disabled
 */
/*@{*/
/*! @brief Read current value of the SIM_FCFG1_FLASHDIS field. */  

#define SIM_RD_FCFG1_FLASHDIS(base) ((SIM_FCFG1_REG(base) &  

SIM_FCFG1_FLASHDIS_MASK) >> SIM_FCFG1_FLASHDIS_SHIFT)  

#define SIM_BRD_FCFG1_FLASHDIS(base) (BME_UBFX32(&SIM_FCFG1_REG(base),  

SIM_FCFG1_FLASHDIS_SHIFT, SIM_FCFG1_FLASHDIS_WIDTH))

/*! @brief Set the FLASHDIS field to a new value. */  

#define SIM_WR_FCFG1_FLASHDIS(base, value) (SIM_RMW_FCFG1(base,  

SIM_FCFG1_FLASHDIS_MASK, SIM_FCFG1_FLASHDIS(value)))  

#define SIM_BWR_FCFG1_FLASHDIS(base, value)  

(BME_BFI32(&SIM_FCFG1_REG(base), ((uint32_t)(value) <<  

SIM_FCFG1_FLASHDIS_SHIFT), SIM_FCFG1_FLASHDIS_SHIFT,  

SIM_FCFG1_FLASHDIS_WIDTH))  

/*@}*/

/*
 * @name Register SIM_FCFG1, field FLASHDOZE[1] (RW)
 *
 * When set, Flash memory is disabled for the duration of Doze mode. This
bit
 * should be clear during VLP modes. The Flash will be automatically
enabled again
 * at the end of Doze mode so interrupt vectors do not need to be
relocated out
 * of Flash memory. The wakeup time from Doze mode is extended when this
bit is

```

```

 * set. An attempt by the DMA or other bus master to access the Flash
when the
 * Flash is disabled will result in a bus error.
 *
 * Values:
 * - 0b0 - Flash remains enabled during Doze mode
 * - 0b1 - Flash is disabled for the duration of Doze mode
 */
/*@{*/
/*! @brief Read current value of the SIM_FCFG1_FLASHDOZE field. */
#define SIM_RD_FCFG1_FLASHDOZE(base) ((SIM_FCFG1_REG(base) &
SIM_FCFG1_FLASHDOZE_MASK) >> SIM_FCFG1_FLASHDOZE_SHIFT)
#define SIM_BRD_FCFG1_FLASHDOZE(base) (BME_UBFX32(&SIM_FCFG1_REG(base),
SIM_FCFG1_FLASHDOZE_SHIFT, SIM_FCFG1_FLASHDOZE_WIDTH))

/*! @brief Set the FLASHDOZE field to a new value. */
#define SIM_WR_FCFG1_FLASHDOZE(base, value) (SIM_RMW_FCFG1(base,
SIM_FCFG1_FLASHDOZE_MASK, SIM_FCFG1_FLASHDOZE(value)))
#define SIM_BWR_FCFG1_FLASHDOZE(base, value)
(BME_BFI32(&SIM_FCFG1_REG(base), ((uint32_t)(value) <<
SIM_FCFG1_FLASHDOZE_SHIFT), SIM_FCFG1_FLASHDOZE_SHIFT,
SIM_FCFG1_FLASHDOZE_WIDTH))
/*}@*/ */

/*!
 * @name Register SIM_FCFG1, field PFSIZE[27:24] (RO)
 *
 * This field specifies the amount of program flash memory available on
the
 * device . Undefined values are reserved.
 *
 * Values:
 * - 0b0000 - 8 KB of program flash memory, 0.25 KB protection region
 * - 0b0001 - 16 KB of program flash memory, 0.5 KB protection region
 * - 0b0011 - 32 KB of program flash memory, 1 KB protection region
 * - 0b0101 - 64 KB of program flash memory, 2 KB protection region
 * - 0b0111 - 128 KB of program flash memory, 4 KB protection region
 * - 0b1001 - 256 KB of program flash memory, 8 KB protection region
 * - 0b1111 - 128 KB of program flash memory, 4 KB protection region
 */
/*@{*/
/*! @brief Read current value of the SIM_FCFG1_PFSIZE field. */
#define SIM_RD_FCFG1_PFSIZE(base) ((SIM_FCFG1_REG(base) &
SIM_FCFG1_PFSIZE_MASK) >> SIM_FCFG1_PFSIZE_SHIFT)
#define SIM_BRD_FCFG1_PFSIZE(base) (BME_UBFX32(&SIM_FCFG1_REG(base),
SIM_FCFG1_PFSIZE_SHIFT, SIM_FCFG1_PFSIZE_WIDTH))
/*}@*/ */

*****  

* SIM_FCFG2 - Flash Configuration Register 2  

*****  

*****/

```

```

/*!
 * @brief SIM_FCFG2 - Flash Configuration Register 2 (RO)
 *
 * Reset value: 0x7F800000U
 */
/*!
 * @name Constants and macros for entire SIM_FCFG2 register
 */
/*@{*/
#define SIM_RD_FCFG2(base)          (SIM_FCFG2_REG(base))
/*}@*/



/*
 * Constants & macros for individual SIM_FCFG2 bitfields
 */

/*!
 * @name Register SIM_FCFG2, field MAXADDR0[30:24] (RO)
 *
 * This field concatenated with leading zeros indicates the first invalid
 * address of program flash. For example, if MAXADDR0 = 0x10 the first
invalid address
 * of program flash is 0x0002_0000. This would be the MAXADDR0 value for
a device
 * with 128 KB program flash.
 */
/*@{*/
/*! @brief Read current value of the SIM_FCFG2_MAXADDR0 field. */
#define SIM_RD_FCFG2_MAXADDR0(base) ((SIM_FCFG2_REG(base) &
SIM_FCFG2_MAXADDR0_MASK) >> SIM_FCFG2_MAXADDR0_SHIFT)
#define SIM_BRD_FCFG2_MAXADDR0(base) (BME_UBFX32(&SIM_FCFG2_REG(base),
SIM_FCFG2_MAXADDR0_SHIFT, SIM_FCFG2_MAXADDR0_WIDTH))
/*}@*/



*****  

*****/



 * SIM_UIDMH - Unique Identification Register Mid-High
*****  

****/



/*!
 * @brief SIM_UIDMH - Unique Identification Register Mid-High (RO)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire SIM_UIDMH register
 */
/*@{*/
#define SIM_RD_UIDMH(base)          (SIM_UIDMH_REG(base))
/*}@*/

```

```

/*
 * Constants & macros for individual SIM_UIDMH bitfields
 */

/*!
 * @name Register SIM_UIDMH, field UID[15:0] (RO)
 *
 * Unique identification for the device.
 */
/*@{*/
/*! @brief Read current value of the SIM_UIDMH_UID field. */
#define SIM_RD_UIDMH_UID(base) ((SIM_UIDMH_REG(base) &
SIM_UIDMH_UID_MASK) >> SIM_UIDMH_UID_SHIFT)
#define SIM_BRD_UIDMH_UID(base) (BME_UBFX32(&SIM_UIDMH_REG(base),
SIM_UIDMH_UID_SHIFT, SIM_UIDMH_UID_WIDTH))
/*}@*/



/*********************  

*****  

 * SIM_UIDML - Unique Identification Register Mid Low  

*****  

/*********************  

****/  

  

/*!
 * @brief SIM_UIDML - Unique Identification Register Mid Low (RO)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire SIM_UIDML register
 */
/*@{*/
#define SIM_RD_UIDML(base) (SIM_UIDML_REG(base))
/*}@*/



/*********************  

*****  

 * SIM_UIDL - Unique Identification Register Low  

*****  

/*********************  

****/  

  

/*!
 * @brief SIM_UIDL - Unique Identification Register Low (RO)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire SIM_UIDL register
 */
/*@{*/
#define SIM_RD_UIDL(base) (SIM_UIDL_REG(base))
/*}@*/

```

```

/*****
 * SIM_COPC - COP Control Register
 *****/
/*!
 * @brief SIM_COPC - COP Control Register (RW)
 *
 * Reset value: 0x0000000CU
 *
 * All of the bits in this register can be written only once after a
reset.
 */
/*!
 * @name Constants and macros for entire SIM_COPC register
 */
/*@{ */
#define SIM_RD_COPC(base)          (SIM_COPC_REG(base))
#define SIM_WR_COPC(base, value)   (SIM_COPC_REG(base) = (value))
#define SIM_RMW_COPC(base, mask, value) (SIM_WR_COPC(base,
(SIM_RD_COPC(base) & ~(mask)) | (value)))
#define SIM_SET_COPC(base, value)  (BME_OR32(&SIM_COPC_REG(base),
(uint32_t)(value)))
#define SIM_CLR_COPC(base, value)  (BME_AND32(&SIM_COPC_REG(base),
(uint32_t)(~(value))))
#define SIM_TOG_COPC(base, value)  (BME_XOR32(&SIM_COPC_REG(base),
(uint32_t)(value)))
/*@} */

/*
 * Constants & macros for individual SIM_COPC bitfields
 */
/*!
 * @name Register SIM_COPC, field COPW[0] (RW)
 *
 * Windowed mode is only supported when COP is running from the bus
clock. The
 * COP window is opened three quarters through the timeout period.
 *
 * Values:
 * - 0b0 - Normal mode
 * - 0b1 - Windowed mode
 */
/*@{ */
/*! @brief Read current value of the SIM_COPC_COPW field. */
#define SIM_RD_COPC_COPW(base) ((SIM_COPC_REG(base) & SIM_COPC_COPW_MASK)
>> SIM_COPC_COPW_SHIFT)
#define SIM_BRD_COPC_COPW(base) (BME_UBFX32(&SIM_COPC_REG(base),
SIM_COPC_COPW_SHIFT, SIM_COPC_COPW_WIDTH))

```

```

/*! @brief Set the COPW field to a new value. */
#define SIM_WR_COPC_COPW(base, value) (SIM_RMW_COPC(base,
SIM_COPC_COPW_MASK, SIM_COPC_COPW(value)))
#define SIM_BWR_COPC_COPW(base, value) (BME_BFI32(&SIM_COPC_REG(base),
((uint32_t)(value) << SIM_COPC_COPW_SHIFT), SIM_COPC_COPW_SHIFT,
SIM_COPC_COPW_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_COPC, field COPCLKS[1] (RW)
 *
 * This write-once bit selects the clock source of the COP watchdog.
 *
 * Values:
 * - 0b0 - Internal 1 kHz clock is source to COP
 * - 0b1 - Bus clock is source to COP
 */
/*@{*/
/*! @brief Read current value of the SIM_COPC_COPCLKS field. */
#define SIM_RD_COPC_COPCLKS(base) ((SIM_COPC_REG(base) &
SIM_COPC_COPCLKS_MASK) >> SIM_COPC_COPCLKS_SHIFT)
#define SIM_BRD_COPC_COPCLKS(base) (BME_UBFX32(&SIM_COPC_REG(base),
SIM_COPC_COPCLKS_SHIFT, SIM_COPC_COPCLKS_WIDTH))

/*! @brief Set the COPCLKS field to a new value. */
#define SIM_WR_COPC_COPCLKS(base, value) (SIM_RMW_COPC(base,
SIM_COPC_COPCLKS_MASK, SIM_COPC_COPCLKS(value)))
#define SIM_BWR_COPC_COPCLKS(base, value) (BME_BFI32(&SIM_COPC_REG(base),
((uint32_t)(value) << SIM_COPC_COPCLKS_SHIFT), SIM_COPC_COPCLKS_SHIFT,
SIM_COPC_COPCLKS_WIDTH))
/*@}*/

/*!!
 * @name Register SIM_COPC, field COPT[3:2] (RW)
 *
 * These write-once bits select the timeout period of the COP. The COPT
field
 * along with the COPCLKS bit define the COP timeout period.
 *
 * Values:
 * - 0b00 - COP disabled
 * - 0b01 - COP timeout after 2^5 LPO cycles or 213 bus clock cycles
 * - 0b10 - COP timeout after 2^8 LPO cycles or 216 bus clock cycles
 * - 0b11 - COP timeout after 2^10 LPO cycles or 218 bus clock cycles
 */
/*@{*/
/*! @brief Read current value of the SIM_COPC_COPT field. */
#define SIM_RD_COPC_COPT(base) ((SIM_COPC_REG(base) & SIM_COPC_COPT_MASK)
>> SIM_COPC_COPT_SHIFT)
#define SIM_BRD_COPC_COPT(base) (BME_UBFX32(&SIM_COPC_REG(base),
SIM_COPC_COPT_SHIFT, SIM_COPC_COPT_WIDTH))

/*! @brief Set the COPT field to a new value. */

```

```

#define SIM_WR_COPC_COPT(base, value) (SIM_RMW_COPC(base,
SIM_COPC_COPT_MASK, SIM_COPC_COPT(value)))
#define SIM_BWR_COPC_COPT(base, value) (BME_BFI32(&SIM_COPC_REG(base),
((uint32_t)(value) << SIM_COPC_COPT_SHIFT), SIM_COPC_COPT_SHIFT,
SIM_COPC_COPT_WIDTH))
/*@}*/

/********************* Service COP Register ********************
****

 * SIM_SRVCOP - Service COP Register

***** */

/*!
 * @brief SIM_SRVCOP - Service COP Register (WO)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire SIM_SRVCOP register
 */
/*@{ */
#define SIM_WR_SRVCOP(base, value) (SIM_SRVCOP_REG(base) = (value))
/*@}*/

/*
 * Constants & macros for individual SIM_SRVCOP bitfields
 */

/*!
 * @name Register SIM_SRVCOP, field SRVCOP[7:0] (WO)
 *
 * Write 0x55 and then 0xAA (in that order) to reset the COP timeout
counter.
 */
/*@{ */
/*! @brief Set the SRVCOP field to a new value. */
#define SIM_WR_SRVCOP_SRVCOP(base, value) (SIM_WR_SRVCOP(base,
SIM_SRVCOP_SRVCOP(value)))
#define SIM_BWR_SRVCOP_SRVCOP(base, value) (SIM_WR_SRVCOP_SRVCOP(base,
value))
/*@}*/

/*
 * MKL25Z4 SMC
 *
 * System Mode Controller
 *
 * Registers defined in this header file:
 * - SMC_PMPROT - Power Mode Protection register
 * - SMC_PMCTRL - Power Mode Control register
 * - SMC_STOPCTRL - Stop Control Register
 * - SMC_PMSTAT - Power Mode Status register

```

```

*/



#define SMC_INSTANCE_COUNT (1U) /*!< Number of instances of the SMC
module. */
#define SMC_IDX (0U) /*!< Instance number for SMC. */

/********************* SMC_PMPROT - Power Mode Protection register *****/
****

 * SMC_PMPROT - Power Mode Protection register

*****/


/*!
 * @brief SMC_PMPROT - Power Mode Protection register (RW)
 *
 * Reset value: 0x00U
 *
 * This register provides protection for entry into any low-power run or
stop
 * mode. The enabling of the low-power run or stop mode occurs by
configuring the
 * Power Mode Control register (PMCTRL). The PMPROT register can be
written only
 * once after any system reset. If the MCU is configured for a disallowed
or
 * reserved power mode, the MCU remains in its current power mode. For
example, if the
 * MCU is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode
using
 * PMCTRL[RUNM] is blocked and the RUNM bits remain 00b, indicating the
MCU is
 * still in Normal Run mode. This register is reset on Chip Reset not
VLLS and by
 * reset types that trigger Chip Reset not VLLS. It is unaffected by
reset types
 * that do not trigger Chip Reset not VLLS. See the Reset section details
for more
 * information.
 */
/*!
 * @name Constants and macros for entire SMC_PMPROT register
 */
/*@{ */

#define SMC_RD_PMPROT(base)      (SMC_PMPROT_REG(base))
#define SMC_WR_PMPROT(base, value) (SMC_PMPROT_REG(base) = (value))
#define SMC_RMW_PMPROT(base, mask, value) (SMC_WR_PMPROT(base,
(SMC_RD_PMPROT(base) & ~mask) | (value)))
#define SMC_SET_PMPROT(base, value) (BME_OR8(&SMC_PMPROT_REG(base),
(uint8_t)(value)))
#define SMC_CLR_PMPROT(base, value) (BME_AND8(&SMC_PMPROT_REG(base),
(uint8_t)(~(value))))
#define SMC_TOG_PMPROT(base, value) (BME_XOR8(&SMC_PMPROT_REG(base),
(uint8_t)(value)))

```

```

/*@}*/

/*
 * Constants & macros for individual SMC_PMPROT bitfields
 */

/*!
 * @name Register SMC_PMPROT, field AVLLS[1] (RW)
 *
 * Provided the appropriate control bits are set up in PMCTRL, this write
once
 * bit allows the MCU to enter any very-low-leakage stop mode (VLLSx).
*
 * Values:
 * - 0b0 - Any VLLSx mode is not allowed
 * - 0b1 - Any VLLSx mode is allowed
 */
/*@{*/
/*! @brief Read current value of the SMC_PMPROT_AVLLS field. */
#define SMC_RD_PMPROT_AVLLS(base) ((SMC_PMPROT_REG(base) &
SMC_PMPROT_AVLLS_MASK) >> SMC_PMPROT_AVLLS_SHIFT)
#define SMC_BRD_PMPROT_AVLLS(base) (BME_UBFX8(&SMC_PMPROT_REG(base),
SMC_PMPROT_AVLLS_SHIFT, SMC_PMPROT_AVLLS_WIDTH))

/*! @brief Set the AVLLS field to a new value. */
#define SMC_WR_PMPROT_AVLLS(base, value) (SMC_RMW_PMPROT(base,
SMC_PMPROT_AVLLS_MASK, SMC_PMPROT_AVLLS(value)))
#define SMC_BWR_PMPROT_AVLLS(base, value)
(BME_BFI8(&SMC_PMPROT_REG(base), ((uint8_t)(value) <<
SMC_PMPROT_AVLLS_SHIFT), SMC_PMPROT_AVLLS_SHIFT, SMC_PMPROT_AVLLS_WIDTH))
/*@*/ */

/*!
 * @name Register SMC_PMPROT, field ALLS[3] (RW)
 *
 * This write once bit allows the MCU to enter any low-leakage stop mode
(LLS),
 * provided the appropriate control bits are set up in PMCTRL.
*
 * Values:
 * - 0b0 - LLS is not allowed
 * - 0b1 - LLS is allowed
 */
/*@{*/
/*! @brief Read current value of the SMC_PMPROT_ALLS field. */
#define SMC_RD_PMPROT_ALLS(base) ((SMC_PMPROT_REG(base) &
SMC_PMPROT_ALLS_MASK) >> SMC_PMPROT_ALLS_SHIFT)
#define SMC_BRD_PMPROT_ALLS(base) (BME_UBFX8(&SMC_PMPROT_REG(base),
SMC_PMPROT_ALLS_SHIFT, SMC_PMPROT_ALLS_WIDTH))

/*! @brief Set the ALLS field to a new value. */
#define SMC_WR_PMPROT_ALLS(base, value) (SMC_RMW_PMPROT(base,
SMC_PMPROT_ALLS_MASK, SMC_PMPROT_ALLS(value)))

```

```

#define SMC_BWR_PMPROT_ALLS(base, value) (BME_BFI8(&SMC_PMPROT_REG(base),  

((uint8_t)(value) << SMC_PMPROT_ALLS_SHIFT), SMC_PMPROT_ALLS_SHIFT,  

SMC_PMPROT_ALLS_WIDTH))  

/*@}*/

/*!  

 * @name Register SMC_PMPROT, field AVLP[5] (RW)  

 *  

 * Provided the appropriate control bits are set up in PMCTRL, this  

write-once  

 * bit allows the MCU to enter any very-low-power modes: VLPR, VLPW, and  

VLPS.  

 *  

 * Values:  

 * - 0b0 - VLPR, VLPW and VLPS are not allowed  

 * - 0b1 - VLPR, VLPW and VLPS are allowed  

 */  

/*@{/**/  

/*! @brief Read current value of the SMC_PMPROT_AVLP field. */  

#define SMC_RD_PMPROT_AVLP(base) ((SMC_PMPROT_REG(base) &  

SMC_PMPROT_AVLP_MASK) >> SMC_PMPROT_AVLP_SHIFT)  

#define SMC_BRD_PMPROT_AVLP(base) (BME_UBFX8(&SMC_PMPROT_REG(base),  

SMC_PMPROT_AVLP_SHIFT, SMC_PMPROT_AVLP_WIDTH))

/*! @brief Set the AVLP field to a new value. */  

#define SMC_WR_PMPROT_AVLP(base, value) (SMC_RMW_PMPROT(base,  

SMC_PMPROT_AVLP_MASK, SMC_PMPROT_AVLP(value)))  

#define SMC_BWR_PMPROT_AVLP(base, value) (BME_BFI8(&SMC_PMPROT_REG(base),  

((uint8_t)(value) << SMC_PMPROT_AVLP_SHIFT), SMC_PMPROT_AVLP_SHIFT,  

SMC_PMPROT_AVLP_WIDTH))  

/*@*/
```

\*\*\*\*\*  
\*\*\*\*\*  
\* SMC\_PMCTRL - Power Mode Control register  
\*\*\*\*\*  
\*\*\*\*\*

```

/*!  

 * @brief SMC_PMCTRL - Power Mode Control register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The PMCTRL register controls entry into low-power run and stop modes,  

 * provided that the selected power mode is allowed via an appropriate  

setting of the  

 * protection (PMPROT) register. This register is reset on Chip POR not  

VLLS and by  

 * reset types that trigger Chip POR not VLLS. It is unaffected by reset  

types  

 * that do not trigger Chip POR not VLLS. See the Reset section details  

for more  

 * information.
```

```

/*
/*!
 * @name Constants and macros for entire SMC_PMCTRL register
 */
/*@{ */
#define SMC_RD_PMCTRL(base)      (SMC_PMCTRL_REG(base))
#define SMC_WR_PMCTRL(base, value) (SMC_PMCTRL_REG(base) = (value))
#define SMC_RMW_PMCTRL(base, mask, value) (SMC_WR_PMCTRL(base,
(SMC_RD_PMCTRL(base) & ~mask)) | (value)))
#define SMC_SET_PMCTRL(base, value) (BME_OR8(&SMC_PMCTRL_REG(base),
(uint8_t)(value)))
#define SMC_CLR_PMCTRL(base, value) (BME_AND8(&SMC_PMCTRL_REG(base),
(uint8_t)(~(value))))
#define SMC_TOG_PMCTRL(base, value) (BME_XOR8(&SMC_PMCTRL_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual SMC_PMCTRL bitfields
 */

/*!
 * @name Register SMC_PMCTRL, field STOPM[2:0] (RW)
 *
 * When written, controls entry into the selected stop mode when Sleep-Now or
 * Sleep-On-Exit mode is entered with SLEEPDEEP=1 . Writes to this field are
 * blocked if the protection level has not been enabled using the PMPROT register.
 * After any system reset, this field is cleared by hardware on any successful write
 * to the PMPROT register. When set to VLLSm, the VLLSM bits in the STOPCTRL
 * register is used to further select the particular VLLS submode which will be
 * entered. When set to STOP, the PSTOPO bits in the STOPCTRL register can be used to
 * select a Partial Stop mode if desired.
 *
 * Values:
 * - 0b000 - Normal Stop (STOP)
 * - 0b001 - Reserved
 * - 0b010 - Very-Low-Power Stop (VLPS)
 * - 0b011 - Low-Leakage Stop (LLS)
 * - 0b100 - Very-Low-Leakage Stop (VLLSm)
 * - 0b101 - Reserved
 * - 0b110 - Reserved
 * - 0b111 - Reserved
 */
/*@{ */
/*! @brief Read current value of the SMC_PMCTRL_STOPM field. */
#define SMC_RD_PMCTRL_STOPM(base) ((SMC_PMCTRL_REG(base) &
SMC_PMCTRL_STOPM_MASK) >> SMC_PMCTRL_STOPM_SHIFT)

```

```

#define SMC_BRD_PMCTRL_STOPM(base) (BME_UBFX8(&SMC_PMCTRL_REG(base),  

SMC_PMCTRL_STOPM_SHIFT, SMC_PMCTRL_STOPM_WIDTH))

/*! @brief Set the STOPM field to a new value. */  

#define SMC_WR_PMCTRL_STOPM(base, value) (SMC_RMW_PMCTRL(base,  

SMC_PMCTRL_STOPM_MASK, SMC_PMCTRL_STOPM(value)))  

#define SMC_BWR_PMCTRL_STOPM(base, value)  

(BME_BFI8(&SMC_PMCTRL_REG(base), ((uint8_t)(value) <<  

SMC_PMCTRL_STOPM_SHIFT), SMC_PMCTRL_STOPM_SHIFT, SMC_PMCTRL_STOPM_WIDTH))  

/*@*/
```

/\*!

- \* @name Register SMC\_PMCTRL, field STOPA[3] (RO)
- \*
- \* When set, this read-only status bit indicates an interrupt or reset occurred
- \* during the previous stop mode entry sequence, preventing the system from
- \* entering that mode. This bit is cleared by hardware at the beginning of any stop
- \* mode entry sequence and is set if the sequence was aborted.
- \*
- \* Values:
- \* - 0b0 - The previous stop mode entry was successful.
- \* - 0b1 - The previous stop mode entry was aborted.
- \*/

```
/*@*/
/*! @brief Read current value of the SMC_PMCTRL_STOPA field. */  

#define SMC_RD_PMCTRL_STOPA(base) ((SMC_PMCTRL_REG(base) &  

SMC_PMCTRL_STOPA_MASK) >> SMC_PMCTRL_STOPA_SHIFT)  

#define SMC_BRD_PMCTRL_STOPA(base) (BME_UBFX8(&SMC_PMCTRL_REG(base),  

SMC_PMCTRL_STOPA_SHIFT, SMC_PMCTRL_STOPA_WIDTH))  

/*@*/
```

/\*!

- \* @name Register SMC\_PMCTRL, field RUNM[6:5] (RW)
- \*
- \* When written, causes entry into the selected run mode. Writes to this field
- \* are blocked if the protection level has not been enabled using the PMPROT
- \* register. This field is cleared by hardware on any exit to normal RUN mode. RUNM
- \* must be set to VLPR only when PMSTAT=RUN. After being written to VLPR, RUNM
- \* should not be written back to RUN until PMSTAT=VLPR. RUNM must be set to RUN only
- \* when PMSTAT=VLPR. After being written to RUN, RUNM should not be written back
- \* to VLPR until PMSTAT=RUN.
- \*
- \* Values:
- \* - 0b00 - Normal Run mode (RUN)
- \* - 0b01 - Reserved

```

* - 0b10 - Very-Low-Power Run mode (VLPR)
* - 0b11 - Reserved
*/
/*@{ */
/*! @brief Read current value of the SMC_PMCTRL_RUNM field. */
#define SMC_RD_PMCTRL_RUNM(base) ((SMC_PMCTRL_REG(base) &
SMC_PMCTRL_RUNM_MASK) >> SMC_PMCTRL_RUNM_SHIFT)
#define SMC_BRD_PMCTRL_RUNM(base) (BME_UBFX8(&SMC_PMCTRL_REG(base),
SMC_PMCTRL_RUNM_SHIFT, SMC_PMCTRL_RUNM_WIDTH))

/*! @brief Set the RUNM field to a new value. */
#define SMC_WR_PMCTRL_RUNM(base, value) (SMC_RMW_PMCTRL(base,
SMC_PMCTRL_RUNM_MASK, SMC_PMCTRL_RUNM(value)))
#define SMC_BWR_PMCTRL_RUNM(base, value) (BME_BFI8(&SMC_PMCTRL_REG(base),
((uint8_t)(value) << SMC_PMCTRL_RUNM_SHIFT), SMC_PMCTRL_RUNM_SHIFT,
SMC_PMCTRL_RUNM_WIDTH))
/*}@ */

/********************* Stop Control Register ********************
*****
 * SMC_STOPCTRL - Stop Control Register
*****
***** */

/*!
* @brief SMC_STOPCTRL - Stop Control Register (RW)
*
* Reset value: 0x03U
*
* The STOPCTRL register provides various control bits allowing the user
to fine
* tune power consumption during the stop mode selected by the STOPM
field. This
* register is reset on Chip POR not VLLS and by reset types that trigger
Chip
* POR not VLLS. It is unaffected by reset types that do not trigger Chip
POR not
* VLLS. See the Reset section details for more information.
*/
/*!
* @name Constants and macros for entire SMC_STOPCTRL register
*/
/*@{ */
#define SMC_RD_STOPCTRL(base) (SMC_STOPCTRL_REG(base))
#define SMC_WR_STOPCTRL(base, value) (SMC_STOPCTRL_REG(base) = (value))
#define SMC_RMW_STOPCTRL(base, mask, value) (SMC_WR_STOPCTRL(base,
(SMC_RD_STOPCTRL(base) & ~mask) | value)))
#define SMC_SET_STOPCTRL(base, value) (BME_OR8(&SMC_STOPCTRL_REG(base),
(uint8_t)(value)))
#define SMC_CLR_STOPCTRL(base, value) (BME_AND8(&SMC_STOPCTRL_REG(base),
(uint8_t)(~value)))
#define SMC_TOG_STOPCTRL(base, value) (BME_XOR8(&SMC_STOPCTRL_REG(base),
(uint8_t)(value)))

```

```

/*@}*/

/*
 * Constants & macros for individual SMC_STOPCTRL bitfields
 */

/*!
 * @name Register SMC_STOPCTRL, field VLLSM[2:0] (RW)
 *
 * This field controls which VLLS sub-mode to enter if STOPM=VLLS.
 *
 * Values:
 * - 0b000 - VLLS0
 * - 0b001 - VLLS1
 * - 0b010 - Reserved
 * - 0b011 - VLLS3
 * - 0b100 - Reserved
 * - 0b101 - Reserved
 * - 0b110 - Reserved
 * - 0b111 - Reserved
 */
/*@{*/
/*! @brief Read current value of the SMC_STOPCTRL_VLLSM field. */
#define SMC_RD_STOPCTRL_VLLSM(base) ((SMC_STOPCTRL_REG(base) &
SMC_STOPCTRL_VLLSM_MASK) >> SMC_STOPCTRL_VLLSM_SHIFT)
#define SMC_BRD_STOPCTRL_VLLSM(base) (BME_UBFX8(&SMC_STOPCTRL_REG(base),
SMC_STOPCTRL_VLLSM_SHIFT, SMC_STOPCTRL_VLLSM_WIDTH))

/*! @brief Set the VLLSM field to a new value. */
#define SMC_WR_STOPCTRL_VLLSM(base, value) (SMC_RMW_STOPCTRL(base,
SMC_STOPCTRL_VLLSM_MASK, SMC_STOPCTRL_VLLSM(value)))
#define SMC_BWR_STOPCTRL_VLLSM(base, value)
(BME_BFI8(&SMC_STOPCTRL_REG(base), ((uint8_t)(value) <<
SMC_STOPCTRL_VLLSM_SHIFT), SMC_STOPCTRL_VLLSM_SHIFT,
SMC_STOPCTRL_VLLSM_WIDTH))
/*@}*/
/*!
 * @name Register SMC_STOPCTRL, field PORPO[5] (RW)
 *
 * This bit controls whether the POR detect circuit is enabled in VLLS0
mode.
 *
 * Values:
 * - 0b0 - POR detect circuit is enabled in VLLS0
 * - 0b1 - POR detect circuit is disabled in VLLS0
 */
/*@{*/
/*! @brief Read current value of the SMC_STOPCTRL_PORPO field. */
#define SMC_RD_STOPCTRL_PORPO(base) ((SMC_STOPCTRL_REG(base) &
SMC_STOPCTRL_PORPO_MASK) >> SMC_STOPCTRL_PORPO_SHIFT)
#define SMC_BRD_STOPCTRL_PORPO(base) (BME_UBFX8(&SMC_STOPCTRL_REG(base),
SMC_STOPCTRL_PORPO_SHIFT, SMC_STOPCTRL_PORPO_WIDTH))

```

```

/*! @brief Set the PORPO field to a new value. */
#define SMC_WR_STOPCTRL_PORPO(base, value) (SMC_RMW_STOPCTRL(base,
SMC_STOPCTRL_PORPO_MASK, SMC_STOPCTRL_PORPO(value)))
#define SMC_BWR_STOPCTRL_PORPO(base, value)
(BME_BFI8(&SMC_STOPCTRL_REG(base), ((uint8_t)(value) <<
SMC_STOPCTRL_PORPO_SHIFT), SMC_STOPCTRL_PORPO_SHIFT,
SMC_STOPCTRL_PORPO_WIDTH))
/*@}*/

/*!!
 * @name Register SMC_STOPCTRL, field PSTOPO[7:6] (RW)
 *
 * These bits control whether a Partial Stop mode is entered when
STOPM=STOP.
 * When entering a Partial Stop mode from RUN mode, the PMC, MCG and
flash remain
 * fully powered, allowing the device to wakeup almost instantaneously at
the
 * expense of higher power consumption. In PSTOP2, only system clocks are
gated
 * allowing peripherals running on bus clock to remain fully functional.
In PSTOP1,
 * both system and bus clocks are gated.
*
* Values:
* - 0b00 - STOP - Normal Stop mode
* - 0b01 - PSTOP1 - Partial Stop with both system and bus clocks
disabled
* - 0b10 - PSTOP2 - Partial Stop with system clock disabled and bus
clock
*     enabled
* - 0b11 - Reserved
*/
/*@*/
/*! @brief Read current value of the SMC_STOPCTRL_PSTOPO field. */
#define SMC_RD_STOPCTRL_PSTOPO(base) ((SMC_STOPCTRL_REG(base) &
SMC_STOPCTRL_PSTOPO_MASK) >> SMC_STOPCTRL_PSTOPO_SHIFT)
#define SMC_BRD_STOPCTRL_PSTOPO(base) (BME_UBFX8(&SMC_STOPCTRL_REG(base),
SMC_STOPCTRL_PSTOPO_SHIFT, SMC_STOPCTRL_PSTOPO_WIDTH))

/*!! @brief Set the PSTOPO field to a new value. */
#define SMC_WR_STOPCTRL_PSTOPO(base, value) (SMC_RMW_STOPCTRL(base,
SMC_STOPCTRL_PSTOPO_MASK, SMC_STOPCTRL_PSTOPO(value)))
#define SMC_BWR_STOPCTRL_PSTOPO(base, value)
(BME_BFI8(&SMC_STOPCTRL_REG(base), ((uint8_t)(value) <<
SMC_STOPCTRL_PSTOPO_SHIFT), SMC_STOPCTRL_PSTOPO_SHIFT,
SMC_STOPCTRL_PSTOPO_WIDTH))
/*@*/

*****  

* SMC_PMSTAT - Power Mode Status register

```

```
*****
****/


/*!
 * @brief SMC_PMSTAT - Power Mode Status register (RO)
 *
 * Reset value: 0x01U
 *
 * PMSTAT is a read-only, one-hot register which indicates the current
power
 * mode of the system. This register is reset on Chip POR not VLLS and by
reset
 * types that trigger Chip POR not VLLS. It is unaffected by reset types
that do not
 * trigger Chip POR not VLLS. See the Reset section details for more
information.
 */
/*!!
 * @name Constants and macros for entire SMC_PMSTAT register
 */
/*@{*/
#define SMC_RD_PMSTAT(base)          (SMC_PMSTAT_REG(base))
/*}@*/



/*
 * Constants & macros for individual SMC_PMSTAT bitfields
 */
/*!
 * @name Register SMC_PMSTAT, field PMSTAT[6:0] (RO)
 *
 * When debug is enabled, the PMSTAT will not update to STOP or VLPS When
a
 * PSTOP mode is enabled, the PMSTAT will not update to STOP or VLPS
 */
/*@{*/
/*! @brief Read current value of the SMC_PMSTAT_PMSTAT field. */
#define SMC_RD_PMSTAT_PMSTAT(base) ((SMC_PMSTAT_REG(base) &
SMC_PMSTAT_PMSTAT_MASK) >> SMC_PMSTAT_PMSTAT_SHIFT)
#define SMC_BRD_PMSTAT_PMSTAT(base) (BME_UBFX8(&SMC_PMSTAT_REG(base),
SMC_PMSTAT_PMSTAT_SHIFT, SMC_PMSTAT_PMSTAT_WIDTH))
/*}@*/



/*
 * MKL25Z4 SPI
 *
 * Serial Peripheral Interface
 *
 * Registers defined in this header file:
 * - SPI_C1 - SPI control register 1
 * - SPI_C2 - SPI control register 2
 * - SPI_BR - SPI baud rate register
 * - SPI_S - SPI status register

```

```

* - SPI_D - SPI data register
* - SPI_M - SPI match register
*/
#define SPI_INSTANCE_COUNT (2U) /*!< Number of instances of the SPI
module. */
#define SPI0_IDX (0U) /*!< Instance number for SPI0. */
#define SPI1_IDX (1U) /*!< Instance number for SPI1. */

/*****!
 * SPI_C1 - SPI control register 1
*****
 * @brief SPI_C1 - SPI control register 1 (RW)
 *
 * Reset value: 0x04U
 *
 * This read/write register includes the SPI enable control, interrupt
enables,
 * and configuration options.
 */
/*!
 * @name Constants and macros for entire SPI_C1 register
 */
/*@{ */
#define SPI_RD_C1(base)          (SPI_C1_REG(base))
#define SPI_WR_C1(base, value)    (SPI_C1_REG(base) = (value))
#define SPI_RMW_C1(base, mask, value) (SPI_WR_C1(base, (SPI_RD_C1(base) &
~(mask)) | (value)))
#define SPI_SET_C1(base, value)   (BME_OR8(&SPI_C1_REG(base),
(uint8_t)(value)))
#define SPI_CLR_C1(base, value)   (BME_AND8(&SPI_C1_REG(base),
(uint8_t)(~(value))))
#define SPI_TOG_C1(base, value)   (BME_XOR8(&SPI_C1_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual SPI_C1 bitfields
 */
/*!
 * @name Register SPI_C1, field LSBFE[0] (RW)
 *
 * This bit does not affect the position of the MSB and LSB in the data
 * register. Reads and writes of the data register always have the MSB in
bit 7.
 *
 * Values:
 * - 0b0 - SPI serial data transfers start with most significant bit

```

```

* - 0b1 - SPI serial data transfers start with least significant bit
*/
/*@{*/
/*! @brief Read current value of the SPI_C1_LSBFE field. */
#define SPI_RD_C1_LSBFE(base) ((SPI_C1_REG(base) & SPI_C1_LSBFE_MASK) >>
SPI_C1_LSBFE_SHIFT)
#define SPI_BRD_C1_LSBFE(base) (BME_UBFX8(&SPI_C1_REG(base),
SPI_C1_LSBFE_SHIFT, SPI_C1_LSBFE_WIDTH))

/*! @brief Set the LSBFE field to a new value. */
#define SPI_WR_C1_LSBFE(base, value) (SPI_RMW_C1(base, SPI_C1_LSBFE_MASK,
SPI_C1_LSBFE(value)))
#define SPI_BWR_C1_LSBFE(base, value) (BME_BFI8(&SPI_C1_REG(base),
(uint8_t)(value) << SPI_C1_LSBFE_SHIFT), SPI_C1_LSBFE_SHIFT,
SPI_C1_LSBFE_WIDTH)
/*}@*/
```

/\*!

\* @name Register SPI\_C1, field SSOE[1] (RW)

\*

\* This bit is used in combination with the mode fault enable (MODFEN) bit in

\* the C2 register and the master/slave (MSTR) control bit to determine the

\* function of the SS pin.

\*

\* Values:

\* - 0b0 - When MODFEN is 0: In master mode, SS pin function is general-purpose

\* I/O (not SPI). In slave mode, SS pin function is slave select input. When

\* MODFEN is 1: In master mode, SS pin function is SS input for mode fault.

\* In slave mode, SS pin function is slave select input.

\* - 0b1 - When MODFEN is 0: In master mode, SS pin function is general-purpose

\* I/O (not SPI). In slave mode, SS pin function is slave select input. When

\* MODFEN is 1: In master mode, SS pin function is automatic SS output. In

\* slave mode: SS pin function is slave select input.

\*/

/\*@{\*/
/\*! @brief Read current value of the SPI\_C1\_SSOE field. \*/
#define SPI\_RD\_C1\_SSOE(base) ((SPI\_C1\_REG(base) & SPI\_C1\_SSOE\_MASK) >>
SPI\_C1\_SSOE\_SHIFT)
#define SPI\_BRD\_C1\_SSOE(base) (BME\_UBFX8(&SPI\_C1\_REG(base),
SPI\_C1\_SSOE\_SHIFT, SPI\_C1\_SSOE\_WIDTH))

/\*! @brief Set the SSOE field to a new value. \*/
#define SPI\_WR\_C1\_SSOE(base, value) (SPI\_RMW\_C1(base, SPI\_C1\_SSOE\_MASK,
SPI\_C1\_SSOE(value)))

```

#define SPI_BWR_C1_SSOE(base, value) (BME_BFI8(&SPI_C1_REG(base),  

((uint8_t)(value) << SPI_C1_SSOE_SHIFT), SPI_C1_SSOE_SHIFT,  

SPI_C1_SSOE_WIDTH))  

/*@}*/

/*!  

 * @name Register SPI_C1, field CPHA[2] (RW)  

 *  

 * This bit selects one of two clock formats for different kinds of  

 * synchronous  

 * serial peripheral devices. Refer to the description of "SPI Clock  

 * Formats" for  

 * details.  

 *  

 * Values:  

 * - 0b0 - First edge on SPSCK occurs at the middle of the first cycle of  

 * a data  

 *   transfer  

 * - 0b1 - First edge on SPSCK occurs at the start of the first cycle of  

 * a data  

 *   transfer  

 */  

/*@{*/  

/*! @brief Read current value of the SPI_C1_CPHA field. */  

#define SPI_RD_C1_CPHA(base) ((SPI_C1_REG(base) & SPI_C1_CPHA_MASK) >>  

SPI_C1_CPHA_SHIFT)  

#define SPI_BRD_C1_CPHA(base) (BME_UBFX8(&SPI_C1_REG(base),  

SPI_C1_CPHA_SHIFT, SPI_C1_CPHA_WIDTH))

/*! @brief Set the CPHA field to a new value. */  

#define SPI_WR_C1_CPHA(base, value) (SPI_RMW_C1(base, SPI_C1_CPHA_MASK,  

SPI_C1_CPHA(value)))  

#define SPI_BWR_C1_CPHA(base, value) (BME_BFI8(&SPI_C1_REG(base),  

((uint8_t)(value) << SPI_C1_CPHA_SHIFT), SPI_C1_CPHA_SHIFT,  

SPI_C1_CPHA_WIDTH))  

/*@}*/

/*!  

 * @name Register SPI_C1, field CPOL[3] (RW)  

 *  

 * This bit selects an inverted or non-inverted SPI clock. To transmit  

 * data  

 * between SPI modules, the SPI modules must have identical CPOL values.  

 * This bit  

 * effectively places an inverter in series with the clock signal either  

 * from a  

 * master SPI device or to a slave SPI device. Refer to the description  

 * of "SPI Clock  

 * Formats" for details.  

 *  

 * Values:  

 * - 0b0 - Active-high SPI clock (idles low)  

 * - 0b1 - Active-low SPI clock (idles high)  

 */

```

```

/*@{*/
/*! @brief Read current value of the SPI_C1_CPOL field. */
#define SPI_RD_C1_CPOL(base) ((SPI_C1_REG(base) & SPI_C1_CPOL_MASK) >>
SPI_C1_CPOL_SHIFT)
#define SPI_BRD_C1_CPOL(base) (BME_UBFX8(&SPI_C1_REG(base),
SPI_C1_CPOL_SHIFT, SPI_C1_CPOL_WIDTH))

/*! @brief Set the CPOL field to a new value. */
#define SPI_WR_C1_CPOL(base, value) (SPI_RMW_C1(base, SPI_C1_CPOL_MASK,
SPI_C1_CPOL(value)))
#define SPI_BWR_C1_CPOL(base, value) (BME_BFI8(&SPI_C1_REG(base),
((uint8_t)(value) << SPI_C1_CPOL_SHIFT), SPI_C1_CPOL_SHIFT,
SPI_C1_CPOL_WIDTH))
/*}@*/
```

```

/*!
 * @name Register SPI_C1, field MSTR[4] (RW)
 *
 * This bit selects master or slave mode operation.
 *
 * Values:
 * - 0b0 - SPI module configured as a slave SPI device
 * - 0b1 - SPI module configured as a master SPI device
 */
/*@{*/
/*! @brief Read current value of the SPI_C1_MSTR field. */
#define SPI_RD_C1_MSTR(base) ((SPI_C1_REG(base) & SPI_C1_MSTR_MASK) >>
SPI_C1_MSTR_SHIFT)
#define SPI_BRD_C1_MSTR(base) (BME_UBFX8(&SPI_C1_REG(base),
SPI_C1_MSTR_SHIFT, SPI_C1_MSTR_WIDTH))

/*! @brief Set the MSTR field to a new value. */
#define SPI_WR_C1_MSTR(base, value) (SPI_RMW_C1(base, SPI_C1_MSTR_MASK,
SPI_C1_MSTR(value)))
#define SPI_BWR_C1_MSTR(base, value) (BME_BFI8(&SPI_C1_REG(base),
((uint8_t)(value) << SPI_C1_MSTR_SHIFT), SPI_C1_MSTR_SHIFT,
SPI_C1_MSTR_WIDTH))
/*}@*/
```

```

/*!
 * @name Register SPI_C1, field SPTIE[5] (RW)
 *
 * This is the interrupt enable bit for SPI transmit buffer empty
(SPTEF). An
 * interrupt occurs when the SPI transmit buffer is empty (SPTEF is set).
 *
 * Values:
 * - 0b0 - Interrupts from SPTEF inhibited (use polling)
 * - 0b1 - When SPTEF is 1, hardware interrupt requested
 */
/*@{*/
/*! @brief Read current value of the SPI_C1_SPTIE field. */
#define SPI_RD_C1_SPTIE(base) ((SPI_C1_REG(base) & SPI_C1_SPTIE_MASK) >>
SPI_C1_SPTIE_SHIFT)
```

```

#define SPI_BRD_C1_SPTIE(base)  (BME_UBFX8(&SPI_C1_REG(base),  

SPI_C1_SPTIE_SHIFT, SPI_C1_SPTIE_WIDTH))

/*! @brief Set the SPTIE field to a new value. */  

#define SPI_WR_C1_SPTIE(base, value)  (SPI_RMW_C1(base, SPI_C1_SPTIE_MASK,  

SPI_C1_SPTIE(value)))  

#define SPI_BWR_C1_SPTIE(base, value)  (BME_BFI8(&SPI_C1_REG(base),  

((uint8_t)(value) << SPI_C1_SPTIE_SHIFT), SPI_C1_SPTIE_SHIFT,  

SPI_C1_SPTIE_WIDTH))  

/*@}*/

/*!  

 * @name Register SPI_C1, field SPE[6] (RW)  

 *  

 * This bit enables the SPI system and dedicates the SPI port pins to SPI  

system  

 * functions. If SPE is cleared, the SPI is disabled and forced into an  

idle  

 * state, and all status bits in the S register are reset.  

 *  

 * Values:  

 * - 0b0 - SPI system inactive  

 * - 0b1 - SPI system enabled  

 */  

/*@*/  

/*! @brief Read current value of the SPI_C1_SPE field. */  

#define SPI_RD_C1_SPE(base)  ((SPI_C1_REG(base) & SPI_C1_SPE_MASK) >>  

SPI_C1_SPE_SHIFT)  

#define SPI_BRD_C1_SPE(base)  (BME_UBFX8(&SPI_C1_REG(base),  

SPI_C1_SPE_SHIFT, SPI_C1_SPE_WIDTH))

/*! @brief Set the SPE field to a new value. */  

#define SPI_WR_C1_SPE(base, value)  (SPI_RMW_C1(base, SPI_C1_SPE_MASK,  

SPI_C1_SPE(value)))  

#define SPI_BWR_C1_SPE(base, value)  (BME_BFI8(&SPI_C1_REG(base),  

((uint8_t)(value) << SPI_C1_SPE_SHIFT), SPI_C1_SPE_SHIFT,  

SPI_C1_SPE_WIDTH))  

/*@*/

/*!  

 * @name Register SPI_C1, field SPIE[7] (RW)  

 *  

 * This bit enables the interrupt for SPI receive buffer full (SPRF) and  

mode  

 * fault (MODF) events.  

 *  

 * Values:  

 * - 0b0 - Interrupts from SPRF and MODF are inhibited-use polling  

 * - 0b1 - Request a hardware interrupt when SPRF or MODF is 1  

 */  

/*@*/  

/*! @brief Read current value of the SPI_C1_SPIE field. */  

#define SPI_RD_C1_SPIE(base)  ((SPI_C1_REG(base) & SPI_C1_SPIE_MASK) >>  

SPI_C1_SPIE_SHIFT)

```

```

#define SPI_BRD_C1_SPIE(base)  (BME_UBFX8(&SPI_C1_REG(base),  

SPI_C1_SPIE_SHIFT, SPI_C1_SPIE_WIDTH))

/*! @brief Set the SPIE field to a new value. */  

#define SPI_WR_C1_SPIE(base, value)  (SPI_RMW_C1(base, SPI_C1_SPIE_MASK,  

SPI_C1_SPIE(value)))  

#define SPI_BWR_C1_SPIE(base, value)  (BME_BFI8(&SPI_C1_REG(base),  

(uint8_t)(value) << SPI_C1_SPIE_SHIFT), SPI_C1_SPIE_SHIFT,  

SPI_C1_SPIE_WIDTH)  

/*@}*/

/*********************  

*****  

 * SPI_C2 - SPI control register 2  

*****  

/*********************  

****/  

  

/*!  

 * @brief SPI_C2 - SPI control register 2 (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * This read/write register is used to control optional features of the  

SPI  

 * system. Bit 6 is not implemented and always reads 0.  

 */  

/*!  

 * @name Constants and macros for entire SPI_C2 register  

 */  

/*@*/  

#define SPI_RD_C2(base)          (SPI_C2_REG(base))  

#define SPI_WR_C2(base, value)    (SPI_C2_REG(base) = (value))  

#define SPI_RMW_C2(base, mask, value)  (SPI_WR_C2(base, (SPI_RD_C2(base) &  

~(mask)) | (value)))  

#define SPI_SET_C2(base, value)   (BME_OR8(&SPI_C2_REG(base),  

(uint8_t)(value)))  

#define SPI_CLR_C2(base, value)   (BME_AND8(&SPI_C2_REG(base),  

(uint8_t)(~(value))))  

#define SPI_TOG_C2(base, value)   (BME_XOR8(&SPI_C2_REG(base),  

(uint8_t)(value)))  

/*@*/  

  

/*  

 * Constants & macros for individual SPI_C2 bitfields  

 */  

  

/*!  

 * @name Register SPI_C2, field SPC0[0] (RW)  

 *  

 * This bit enables bidirectional pin configurations.  

 *  

 * Values:  


```

```

* - 0b0 - SPI uses separate pins for data input and data output (pin
mode is
*      normal). In master mode of operation: MISO is master in and MOSI
is master
*      out. In slave mode of operation: MISO is slave out and MOSI is
slave in.
* - 0b1 - SPI configured for single-wire bidirectional operation (pin
mode is
*      bidirectional). In master mode of operation: MISO is not used by
SPI; MOSI
*      is master in when BIDIROE is 0 or master I/O when BIDIROE is 1. In
slave
*      mode of operation: MISO is slave in when BIDIROE is 0 or slave I/O
when
*      BIDIROE is 1; MOSI is not used by SPI.
*/
/*@{*/
/*! @brief Read current value of the SPI_C2_SPC0 field. */
#define SPI_RD_C2_SPC0(base) ((SPI_C2_REG(base) & SPI_C2_SPC0_MASK) >>
SPI_C2_SPC0_SHIFT)
#define SPI_BRD_C2_SPC0(base) (BME_UBXF8(&SPI_C2_REG(base),
SPI_C2_SPC0_SHIFT, SPI_C2_SPC0_WIDTH))

/*! @brief Set the SPC0 field to a new value. */
#define SPI_WR_C2_SPC0(base, value) (SPI_RMW_C2(base, SPI_C2_SPC0_MASK,
SPI_C2_SPC0(value)))
#define SPI_BWR_C2_SPC0(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_SPC0_SHIFT), SPI_C2_SPC0_SHIFT,
SPI_C2_SPC0_WIDTH))
/*@}*/
/*!
* @name Register SPI_C2, field SPISWAI[1] (RW)
*
* This bit is used for power conservation while the device is in wait
mode.
*
* Values:
* - 0b0 - SPI clocks continue to operate in wait mode
* - 0b1 - SPI clocks stop when the MCU enters wait mode
*/
/*@{*/
/*! @brief Read current value of the SPI_C2_SPISWAI field. */
#define SPI_RD_C2_SPISWAI(base) ((SPI_C2_REG(base) & SPI_C2_SPISWAI_MASK)
>> SPI_C2_SPISWAI_SHIFT)
#define SPI_BRD_C2_SPISWAI(base) (BME_UBXF8(&SPI_C2_REG(base),
SPI_C2_SPISWAI_SHIFT, SPI_C2_SPISWAI_WIDTH))

/*! @brief Set the SPISWAI field to a new value. */
#define SPI_WR_C2_SPISWAI(base, value) (SPI_RMW_C2(base,
SPI_C2_SPISWAI_MASK, SPI_C2_SPISWAI(value)))
#define SPI_BWR_C2_SPISWAI(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_SPISWAI_SHIFT), SPI_C2_SPISWAI_SHIFT,
SPI_C2_SPISWAI_WIDTH))

```

```

/*@}*/

/*
 * @name Register SPI_C2, field RXDMAE[2] (RW)
 *
 * This is the enable bit for a receive DMA request. When this bit is set
to 1,
 * a receive DMA request is asserted when both SPRF and SPE are set, and
the
 * interrupt from SPRF is disabled.
 *
 * Values:
 * - 0b0 - DMA request for receive is disabled and interrupt from SPRF is
allowed
 * - 0b1 - DMA request for receive is enabled and interrupt from SPRF is
disabled
 */
/*@*/
/*! @brief Read current value of the SPI_C2_RXDMAE field. */
#define SPI_RD_C2_RXDMAE(base) ((SPI_C2_REG(base) & SPI_C2_RXDMAE_MASK) >> SPI_C2_RXDMAE_SHIFT)
#define SPI_BRD_C2_RXDMAE(base) (BME_UBFX8(&SPI_C2_REG(base),
SPI_C2_RXDMAE_SHIFT, SPI_C2_RXDMAE_WIDTH))

/*! @brief Set the RXDMAE field to a new value. */
#define SPI_WR_C2_RXDMAE(base, value) (SPI_RMW_C2(base,
SPI_C2_RXDMAE_MASK, SPI_C2_RXDMAE(value)))
#define SPI_BWR_C2_RXDMAE(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_RXDMAE_SHIFT), SPI_C2_RXDMAE_SHIFT,
SPI_C2_RXDMAE_WIDTH))
/*@*/

/*
 * @name Register SPI_C2, field BIDIROE[3] (RW)
 *
 * When bidirectional mode is enabled, because SPI pin control 0 (SPC0)
is set
 * to 1, the BIDIROE bit determines whether the SPI data output driver is
enabled
 * to the single bidirectional SPI I/O pin. Depending on whether the SPI
is
 * configured as a master or a slave, it uses the MOSI (MOMI) or MISO
(SISO) pin,
 * respectively, as the single SPI data I/O pin. When SPC0 is 0, BIDIROE
has no
 * meaning or effect.
 *
 * Values:
 * - 0b0 - Output driver disabled so SPI data I/O pin acts as an input
 * - 0b1 - SPI I/O pin enabled as an output
 */
/*@*/
/*! @brief Read current value of the SPI_C2_BIDIROE field. */

```

```

#define SPI_RD_C2_BIDIROE(base) ((SPI_C2_REG(base) & SPI_C2_BIDIROE_MASK)
>> SPI_C2_BIDIROE_SHIFT)
#define SPI_BRD_C2_BIDIROE(base) (BME_UBFX8(&SPI_C2_REG(base),
SPI_C2_BIDIROE_SHIFT, SPI_C2_BIDIROE_WIDTH))

/*! @brief Set the BIDIROE field to a new value. */
#define SPI_WR_C2_BIDIROE(base, value) (SPI_RMW_C2(base,
SPI_C2_BIDIROE_MASK, SPI_C2_BIDIROE(value)))
#define SPI_BWR_C2_BIDIROE(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_BIDIROE_SHIFT), SPI_C2_BIDIROE_SHIFT,
SPI_C2_BIDIROE_WIDTH))
/*@}*/

/*!
 * @name Register SPI_C2, field MODFEN[4] (RW)
 *
 * When the SPI is configured for slave mode, this bit has no meaning or
effect.
 * (The SS pin is the slave select input.) In master mode, this bit
determines
 * how the SS pin is used. For details, refer to the description of the
SSOE bit
 * in the C1 register.
 *
 * Values:
 * - 0b0 - Mode fault function disabled, master SS pin reverts to
 *         general-purpose I/O not controlled by SPI
 * - 0b1 - Mode fault function enabled, master SS pin acts as the mode
fault
 *         input or the slave select output
 */
/*@{*/
/*! @brief Read current value of the SPI_C2_MODFEN field. */
#define SPI_RD_C2_MODFEN(base) ((SPI_C2_REG(base) & SPI_C2_MODFEN_MASK)
>> SPI_C2_MODFEN_SHIFT)
#define SPI_BRD_C2_MODFEN(base) (BME_UBFX8(&SPI_C2_REG(base),
SPI_C2_MODFEN_SHIFT, SPI_C2_MODFEN_WIDTH))

/*! @brief Set the MODFEN field to a new value. */
#define SPI_WR_C2_MODFEN(base, value) (SPI_RMW_C2(base,
SPI_C2_MODFEN_MASK, SPI_C2_MODFEN(value)))
#define SPI_BWR_C2_MODFEN(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_MODFEN_SHIFT), SPI_C2_MODFEN_SHIFT,
SPI_C2_MODFEN_WIDTH))
/*@}*/

/*!
 * @name Register SPI_C2, field TXDMAE[5] (RW)
 *
 * This bit enables a transmit DMA request. When this bit is set to 1, a
 * transmit DMA request is asserted when both SPTEF and SPE are set, and
the interrupt
 * from SPTEF is disabled.
 *

```

```

 * Values:
 * - 0b0 - DMA request for transmit is disabled and interrupt from SPTEF
is
 *     allowed
 * - 0b1 - DMA request for transmit is enabled and interrupt from SPTEF
is
 *     disabled
 */
/*@{*/
/*! @brief Read current value of the SPI_C2_TXDMAE field. */
#define SPI_RD_C2_TXDMAE(base) ((SPI_C2_REG(base) & SPI_C2_TXDMAE_MASK)
>> SPI_C2_TXDMAE_SHIFT)
#define SPI_BRD_C2_TXDMAE(base) (BME_UBFX8(&SPI_C2_REG(base),
SPI_C2_TXDMAE_SHIFT, SPI_C2_TXDMAE_WIDTH))

/*! @brief Set the TXDMAE field to a new value. */
#define SPI_WR_C2_TXDMAE(base, value) (SPI_RMW_C2(base,
SPI_C2_TXDMAE_MASK, SPI_C2_TXDMAE(value)))
#define SPI_BWR_C2_TXDMAE(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_TXDMAE_SHIFT), SPI_C2_TXDMAE_SHIFT,
SPI_C2_TXDMAE_WIDTH))
/*}@*/ */

/*!
 * @name Register SPI_C2, field SPMIE[7] (RW)
 *
 * This is the interrupt enable bit for the SPI receive data buffer
hardware
 * match (SPMF) function.
 *
 * Values:
 * - 0b0 - Interrupts from SPMF inhibited (use polling)
 * - 0b1 - When SPMF is 1, requests a hardware interrupt
 */
/*@{*/
/*! @brief Read current value of the SPI_C2_SPMIE field. */
#define SPI_RD_C2_SPMIE(base) ((SPI_C2_REG(base) & SPI_C2_SPMIE_MASK) >>
SPI_C2_SPMIE_SHIFT)
#define SPI_BRD_C2_SPMIE(base) (BME_UBFX8(&SPI_C2_REG(base),
SPI_C2_SPMIE_SHIFT, SPI_C2_SPMIE_WIDTH))

/*! @brief Set the SPMIE field to a new value. */
#define SPI_WR_C2_SPMIE(base, value) (SPI_RMW_C2(base, SPI_C2_SPMIE_MASK,
SPI_C2_SPMIE(value)))
#define SPI_BWR_C2_SPMIE(base, value) (BME_BFI8(&SPI_C2_REG(base),
((uint8_t)(value) << SPI_C2_SPMIE_SHIFT), SPI_C2_SPMIE_SHIFT,
SPI_C2_SPMIE_WIDTH))
/*}@*/ */

*****  

*****  

 * SPI_BR - SPI baud rate register

```

```
*****
 ****/
/*!
 * @brief SPI_BR - SPI baud rate register (RW)
 *
 * Reset value: 0x00U
 *
 * Use this register to set the prescaler and bit rate divisor for an SPI
 * master. This register may be read or written at any time.
 */
/*!
 * @name Constants and macros for entire SPI_BR register
 */
/*@{*/
#define SPI_RD_BR(base)          (SPI_BR_REG(base))
#define SPI_WR_BR(base, value)   (SPI_BR_REG(base) = (value))
#define SPI_RMW_BR(base, mask, value) (SPI_WR_BR(base, (SPI_RD_BR(base) &
~(mask)) | (value)))
#define SPI_SET_BR(base, value)   (BME_OR8(&SPI_BR_REG(base),
(uint8_t)(value)))
#define SPI_CLR_BR(base, value)   (BME_AND8(&SPI_BR_REG(base),
(uint8_t)(~(value))))
#define SPI_TOG_BR(base, value)   (BME_XOR8(&SPI_BR_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual SPI_BR bitfields
 */

/*!
 * @name Register SPI_BR, field SPR[3:0] (RW)
 *
 * This 4-bit field selects one of nine divisors for the SPI baud rate
divider.
 * The input to this divider comes from the SPI baud rate prescaler.
Refer to the
 * description of "SPI Baud Rate Generation" for details.
 *
 * Values:
 * - 0b0000 - Baud rate divisor is 2
 * - 0b0001 - Baud rate divisor is 4
 * - 0b0010 - Baud rate divisor is 8
 * - 0b0011 - Baud rate divisor is 16
 * - 0b0100 - Baud rate divisor is 32
 * - 0b0101 - Baud rate divisor is 64
 * - 0b0110 - Baud rate divisor is 128
 * - 0b0111 - Baud rate divisor is 256
 * - 0b1000 - Baud rate divisor is 512
 */
/*@*/
/*! @brief Read current value of the SPI_BR_SPR field. */

```

```

#define SPI_RD_BR_SPR(base) ((SPI_BR_REG(base) & SPI_BR_SPR_MASK) >>
SPI_BR_SPR_SHIFT)
#define SPI_BRD_BR_SPR(base) (BME_UBFX8(&SPI_BR_REG(base),
SPI_BR_SPR_SHIFT, SPI_BR_SPR_WIDTH))

/*! @brief Set the SPR field to a new value. */
#define SPI_WR_BR_SPR(base, value) (SPI_RMW_BR(base, SPI_BR_SPR_MASK,
SPI_BR_SPR(value)))
#define SPI_BWR_BR_SPR(base, value) (BME_BFI8(&SPI_BR_REG(base),
((uint8_t)(value) << SPI_BR_SPR_SHIFT), SPI_BR_SPR_SHIFT,
SPI_BR_SPR_WIDTH))
/*@}*/

/*!
 * @name Register SPI_BR, field SPPR[6:4] (RW)
 *
 * This 3-bit field selects one of eight divisors for the SPI baud rate
 * prescaler. The input to this prescaler is the bus rate clock (BUSCLK).
 * The output of
 * this prescaler drives the input of the SPI baud rate divider. Refer to
 * the
 * description of "SPI Baud Rate Generation" for details.
 *
 * Values:
 * - 0b000 - Baud rate prescaler divisor is 1
 * - 0b001 - Baud rate prescaler divisor is 2
 * - 0b010 - Baud rate prescaler divisor is 3
 * - 0b011 - Baud rate prescaler divisor is 4
 * - 0b100 - Baud rate prescaler divisor is 5
 * - 0b101 - Baud rate prescaler divisor is 6
 * - 0b110 - Baud rate prescaler divisor is 7
 * - 0b111 - Baud rate prescaler divisor is 8
 */
/*@*/
/*! @brief Read current value of the SPI_BR_SPPR field. */
#define SPI_RD_BR_SPPR(base) ((SPI_BR_REG(base) & SPI_BR_SPPR_MASK) >>
SPI_BR_SPPR_SHIFT)
#define SPI_BRD_BR_SPPR(base) (BME_UBFX8(&SPI_BR_REG(base),
SPI_BR_SPPR_SHIFT, SPI_BR_SPPR_WIDTH))

/*! @brief Set the SPPR field to a new value. */
#define SPI_WR_BR_SPPR(base, value) (SPI_RMW_BR(base, SPI_BR_SPPR_MASK,
SPI_BR_SPPR(value)))
#define SPI_BWR_BR_SPPR(base, value) (BME_BFI8(&SPI_BR_REG(base),
((uint8_t)(value) << SPI_BR_SPPR_SHIFT), SPI_BR_SPPR_SHIFT,
SPI_BR_SPPR_WIDTH))
/*@*/

*****  

* SPI_S - SPI status register  

*****  

*****/

```

```

/*!
 * @brief SPI_S - SPI status register (RW)
 *
 * Reset value: 0x20U
 *
 * This register contains read-only status bits. Writes have no meaning
 * or effect. Bits 3 through 0 are not implemented and always read 0.
 */
/*!
 * @name Constants and macros for entire SPI_S register
 */
/*@{*/
#define SPI_RD_S(base)          (SPI_S_REG(base))
#define SPI_WR_S(base, value)    (SPI_S_REG(base) = (value))
#define SPI_RMW_S(base, mask, value) (SPI_WR_S(base, (SPI_RD_S(base) &
~(mask)) | (value)))
#define SPI_SET_S(base, value)   (BME_OR8(&SPI_S_REG(base),
(uint8_t)(value)))
#define SPI_CLR_S(base, value)   (BME_AND8(&SPI_S_REG(base),
(uint8_t)(~(value))))
#define SPI_TOG_S(base, value)   (BME_XOR8(&SPI_S_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual SPI_S bitfields
 */
/*!
 * @name Register SPI_S, field MODF[4] (RO)
 *
 * MODF is set if the SPI is configured as a master and the slave select
 * input
 * goes low, indicating some other SPI device is also configured as a
 * master. The
 * SS pin acts as a mode fault error input only when MSTR is 1, MODFEN is
 * 1, and
 * SSOE is 0; otherwise, MODF will never be set. MODF is cleared by
 * reading MODF
 * while it is 1 and then writing to the SPI control register 1 (C1).
 *
 * Values:
 * - 0b0 - No mode fault error
 * - 0b1 - Mode fault error detected
 */
/*@*/
/*! @brief Read current value of the SPI_S_MODF field. */
#define SPI_RD_S_MODF(base) ((SPI_S_REG(base) & SPI_S_MODF_MASK) >>
SPI_S_MODF_SHIFT)
#define SPI_BRD_S_MODF(base) (BME_UBFX8(&SPI_S_REG(base),
SPI_S_MODF_SHIFT, SPI_S_MODF_WIDTH))
/*@}*/

```

```

/*!!
 * @name Register SPI_S, field SPTEF[5] (RO)
 *
 * This bit is set when the transmit data buffer is empty. When the
transmit DMA
 * request is disabled (TXDMAE is 0), SPTEF is cleared by reading the S
register
 * with SPTEF set and then writing a data value to the transmit buffer at
D. The
 * S register must be read with SPTEF set to 1 before writing data to the
D
 * register; otherwise, the D write is ignored. When the transmit DMA
request is
 * enabled (TXDMAE is 1), SPTEF is automatically cleared when the DMA
transfer for
 * the transmit DMA request is completed (TX DMA Done is asserted). SPTEF
is
 * automatically set when all data from the transmit buffer transfers
into the transmit
 * shift register. For an idle SPI, data written to D is transferred to
the
 * shifter almost immediately so that SPTEF is set within two bus cycles,
allowing a
 * second set of data to be queued into the transmit buffer. After
completion of
 * the transfer of the data in the shift register, the queued data from
the
 * transmit buffer automatically moves to the shifter, and SPTEF is set
to indicate
 * that room exists for new data in the transmit buffer. If no new data
is waiting
 * in the transmit buffer, SPTEF simply remains set and no data moves
from the
 * buffer to the shifter. If a transfer does not stop, the last data that
was
 * transmitted is sent out again.
*
* Values:
* - 0b0 - SPI transmit buffer not empty
* - 0b1 - SPI transmit buffer empty
*/
/*@{*/
/*! @brief Read current value of the SPI_S_SPTEF field. */
#define SPI_RD_S_SPTEF(base) ((SPI_S_REG(base) & SPI_S_SPTEF_MASK) >>
SPI_S_SPTEF_SHIFT)
#define SPI_BRD_S_SPTEF(base) (BME_UBFX8(&SPI_S_REG(base),
SPI_S_SPTEF_SHIFT, SPI_S_SPTEF_WIDTH))
/*}@*/
```

```

/*!!
 * @name Register SPI_S, field SPMF[6] (W1C)
*
```

```

 * SPMF is set after SPRF is 1 when the value in the receive data buffer
matches
 * the value in the M register. To clear the flag, read SPMF when it is
set and
 * then write a 1 to it.
 *
 * Values:
 * - 0b0 - Value in the receive data buffer does not match the value in
the M
 *      register
 * - 0b1 - Value in the receive data buffer matches the value in the M
register
 */
/*@{*/
/*! @brief Read current value of the SPI_S_SPMF field. */
#define SPI_RD_S_SPMF(base) ((SPI_S_REG(base) & SPI_S_SPMF_MASK) >>
SPI_S_SPMF_SHIFT)
#define SPI_BRD_S_SPMF(base) (BME_UBFX8(&SPI_S_REG(base),
SPI_S_SPMF_SHIFT, SPI_S_SPMF_WIDTH))

/*! @brief Set the SPMF field to a new value. */
#define SPI_WR_S_SPMF(base, value) (SPI_RMW_S(base, SPI_S_SPMF_MASK,
SPI_S_SPMF(value)))
#define SPI_BWR_S_SPMF(base, value) (BME_BFI8(&SPI_S_REG(base),
((uint8_t)(value) << SPI_S_SPMF_SHIFT), SPI_S_SPMF_SHIFT,
SPI_S_SPMF_WIDTH))
/*}@*/ */

/*!
 * @name Register SPI_S, field SPRF[7] (RO)
 *
 * SPRF is set at the completion of an SPI transfer to indicate that
received
 * data may be read from the SPI data (D) register. When the receive DMA
request is
 * disabled (RXDMAE is 0), SPRF is cleared by reading SPRF while it is
set and
 * then reading the SPI data register. When the receive DMA request is
enabled
 * (RXDMAE is 1), SPRF is automatically cleared when the DMA transfer for
the
 * receive DMA request is completed (RX DMA Done is asserted).
 *
 * Values:
 * - 0b0 - No data available in the receive data buffer
 * - 0b1 - Data available in the receive data buffer
 */
/*@{*/
/*! @brief Read current value of the SPI_S_SPRF field. */
#define SPI_RD_S_SPRF(base) ((SPI_S_REG(base) & SPI_S_SPRF_MASK) >>
SPI_S_SPRF_SHIFT)
#define SPI_BRD_S_SPRF(base) (BME_UBFX8(&SPI_S_REG(base),
SPI_S_SPRF_SHIFT, SPI_S_SPRF_WIDTH))
/*}@*/

```

```

/*****
 * SPI_D - SPI data register
 *****/
/*!
 * @brief SPI_D - SPI data register (RW)
 *
 * Reset value: 0x00U
 *
 * This register is both the input and output register for SPI data. A
 * write to
 *   the register writes to the transmit data buffer, allowing data to be
 *   queued
 *   and transmitted. When the SPI is configured as a master, data queued
 *   in the
 *   transmit data buffer is transmitted immediately after the previous
 *   transmission
 *   has completed. The SPTEF bit in the S register indicates when the
 *   transmit data
 *   buffer is ready to accept new data. When the transmit DMA request is
 *   disabled
 *   (TXDMAE is 0): The S register must be read when SPTEF is set before
 *   writing to
 *   the SPI data register; otherwise, the write is ignored. When the
 *   transmit DMA
 *   request is enabled (TXDMAE is 1) when SPTEF is set, the SPI data
 *   register can
 *   be written automatically by DMA without reading the S register first.
 * Data
 *   may be read from the SPI data register any time after SPRF is set and
 *   before
 *   another transfer is finished. Failure to read the data out of the
 *   receive data
 *   buffer before a new transfer ends causes a receive overrun condition,
 *   and the
 *   data from the new transfer is lost. The new data is lost because the
 *   receive
 *   buffer still held the previous character and was not ready to accept
 *   the new
 *   data. There is no indication for a receive overrun condition, so the
 *   application
 *   system designer must ensure that previous data has been read from the
 *   receive
 *   buffer before a new transfer is initiated.
 */
/*!
 * @name Constants and macros for entire SPI_D register
 */
/*@{*/
#define SPI_RD_D(base)           (SPI_D_REG(base))

```

```

#define SPI_WR_D(base, value)      (SPI_D_REG(base) = (value))
#define SPI_RMW_D(base, mask, value) (SPI_WR_D(base, (SPI_RD_D(base) &
~(mask)) | (value)))
#define SPI_SET_D(base, value)     (BME_OR8(&SPI_D_REG(base),
(uint8_t)(value)))
#define SPI_CLR_D(base, value)     (BME_AND8(&SPI_D_REG(base),
(uint8_t)(~(value))))
#define SPI_TOG_D(base, value)     (BME_XOR8(&SPI_D_REG(base),
(uint8_t)(value)))
/*@}*/

/*********************  

*****  

 * SPI_M - SPI match register  

*****  

****/  

  

/*!  

 * @brief SPI_M - SPI match register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * This register contains the hardware compare value. When the value  

received in  

 * the SPI receive data buffer equals this hardware compare value, the  

SPI match  

 * flag (SPMF) sets.  

 */  

/**!  

 * @name Constants and macros for entire SPI_M register  

 */  

/*@*/  

#define SPI_RD_M(base)           (SPI_M_REG(base))
#define SPI_WR_M(base, value)    (SPI_M_REG(base) = (value))
#define SPI_RMW_M(base, mask, value) (SPI_WR_M(base, (SPI_RD_M(base) &
~(mask)) | (value)))
#define SPI_SET_M(base, value)   (BME_OR8(&SPI_M_REG(base),
(uint8_t)(value)))
#define SPI_CLR_M(base, value)   (BME_AND8(&SPI_M_REG(base),
(uint8_t)(~(value))))
#define SPI_TOG_M(base, value)   (BME_XOR8(&SPI_M_REG(base),
(uint8_t)(value)))
/*@*/  

  

/*
 * MKL25Z4 TPM
 *
 * Timer/PWM Module
 *
 * Registers defined in this header file:
 * - TPM_SC - Status and Control
 * - TPM_CNT - Counter
 * - TPM_MOD - Modulo

```

```

* - TPM_CnSC - Channel (n) Status and Control
* - TPM_CnV - Channel (n) Value
* - TPM_STATUS - Capture and Compare Status
* - TPM_CONF - Configuration
*/
#define TPM_INSTANCE_COUNT (3U) /*!< Number of instances of the TPM
module. */
#define TPM0_IDX (0U) /*!< Instance number for TPM0. */
#define TPM1_IDX (1U) /*!< Instance number for TPM1. */
#define TPM2_IDX (2U) /*!< Instance number for TPM2. */

/***** ****
* TPM_SC - Status and Control
*****
*/
/*!
* @brief TPM_SC - Status and Control (RW)
*
* Reset value: 0x00000000U
*
* SC contains the overflow status flag and control bits used to
configure the
* interrupt enable, module configuration and prescaler factor. These
controls
* relate to all channels within this module.
*/
/*!
* @name Constants and macros for entire TPM_SC register
*/
/*@{ */
#define TPM_RD_SC(base)          (TPM_SC_REG(base))
#define TPM_WR_SC(base, value)    (TPM_SC_REG(base) = (value))
#define TPM_RMW_SC(base, mask, value) (TPM_WR_SC(base, (TPM_RD_SC(base) &
~(mask)) | (value)))
#define TPM_SET_SC(base, value)   (BME_OR32(&TPM_SC_REG(base),
(uint32_t)(value)))
#define TPM_CLR_SC(base, value)   (BME_AND32(&TPM_SC_REG(base),
(uint32_t)(~(value))))
#define TPM_TOG_SC(base, value)   (BME_XOR32(&TPM_SC_REG(base),
(uint32_t)(value)))
/*@} */

/*
* Constants & macros for individual TPM_SC bitfields
*/
/*!
* @name Register TPM_SC, field PS[2:0] (RW)
*

```

```

 * Selects one of 8 division factors for the clock mode selected by CMOD.
This
 * field is write protected. It can be written only when the counter is
disabled.
 *
 * Values:
 * - 0b000 - Divide by 1
 * - 0b001 - Divide by 2
 * - 0b010 - Divide by 4
 * - 0b011 - Divide by 8
 * - 0b100 - Divide by 16
 * - 0b101 - Divide by 32
 * - 0b110 - Divide by 64
 * - 0b111 - Divide by 128
 */
/*@{*/
/*! @brief Read current value of the TPM_SC_PS field. */
#define TPM_RD_SC_PS(base) ((TPM_SC_REG(base) & TPM_SC_PS_MASK) >>
TPM_SC_PS_SHIFT)
#define TPM_BRD_SC_PS(base) (BME_UBFX32(&TPM_SC_REG(base),
TPM_SC_PS_SHIFT, TPM_SC_PS_WIDTH))

/*! @brief Set the PS field to a new value. */
#define TPM_WR_SC_PS(base, value) (TPM_RMW_SC(base, (TPM_SC_PS_MASK |
TPM_SC_TOF_MASK), TPM_SC_PS(value)))
#define TPM_BWR_SC_PS(base, value) (BME_BFI32(&TPM_SC_REG(base),
((uint32_t)(value) << TPM_SC_PS_SHIFT), TPM_SC_PS_SHIFT,
TPM_SC_PS_WIDTH))
/*}@*/ */

/*!
 * @name Register TPM_SC, field CMOD[4:3] (RW)
 *
 * Selects the LPTPM counter clock modes. When disabling the counter,
this field
 * remain set until acknowledged in the LPTPM clock domain.
 *
 * Values:
 * - 0b00 - LPTPM counter is disabled
 * - 0b01 - LPTPM counter increments on every LPTPM counter clock
 * - 0b10 - LPTPM counter increments on rising edge of LPTPM_EXTCLK
synchronized
 *      to the LPTPM counter clock
 * - 0b11 - Reserved
 */
/*@{*/
/*! @brief Read current value of the TPM_SC_CMOD field. */
#define TPM_RD_SC_CMOD(base) ((TPM_SC_REG(base) & TPM_SC_CMOD_MASK) >>
TPM_SC_CMOD_SHIFT)
#define TPM_BRD_SC_CMOD(base) (BME_UBFX32(&TPM_SC_REG(base),
TPM_SC_CMOD_SHIFT, TPM_SC_CMOD_WIDTH))

/*! @brief Set the CMOD field to a new value. */

```

```

#define TPM_WR_SC_CMOD(base, value) (TPM_RMW_SC(base, (TPM_SC_CMOD_MASK |  
TPM_SC_TOF_MASK), TPM_SC_CMOD(value)))  
#define TPM_BWR_SC_CMOD(base, value) (BME_BFI32(&TPM_SC_REG(base),  
(uint32_t)(value) << TPM_SC_CMOD_SHIFT), TPM_SC_CMOD_SHIFT,  
TPM_SC_CMOD_WIDTH))  
/*@}*/

/*
 * @name Register TPM_SC, field CPWMS[5] (RW)
 *
 * Selects CPWM mode. This mode configures the LPTPM to operate in up-
down
 * counting mode. This field is write protected. It can be written only
when the
 * counter is disabled.
 *
 * Values:
 * - 0b0 - LPTPM counter operates in up counting mode.
 * - 0b1 - LPTPM counter operates in up-down counting mode.
 */
/*@*/

/*! @brief Read current value of the TPM_SC_CPWMS field. */  
#define TPM_RD_SC_CPWMS(base) ((TPM_SC_REG(base) & TPM_SC_CPWMS_MASK) >>  
TPM_SC_CPWMS_SHIFT)  
#define TPM_BRD_SC_CPWMS(base) (BME_UBFX32(&TPM_SC_REG(base),  
TPM_SC_CPWMS_SHIFT, TPM_SC_CPWMS_WIDTH))

/*! @brief Set the CPWMS field to a new value. */  
#define TPM_WR_SC_CPWMS(base, value) (TPM_RMW_SC(base, (TPM_SC_CPWMS_MASK  
| TPM_SC_TOF_MASK), TPM_SC_CPWMS(value)))  
#define TPM_BWR_SC_CPWMS(base, value) (BME_BFI32(&TPM_SC_REG(base),  
(uint32_t)(value) << TPM_SC_CPWMS_SHIFT), TPM_SC_CPWMS_SHIFT,  
TPM_SC_CPWMS_WIDTH))  
/*@*/
```

  

```

/*
 * @name Register TPM_SC, field TOIE[6] (RW)
 *
 * Enables LPTPM overflow interrupts.
 *
 * Values:
 * - 0b0 - Disable TOF interrupts. Use software polling or DMA request.
 * - 0b1 - Enable TOF interrupts. An interrupt is generated when TOF
equals one.
 */
/*@*/

/*! @brief Read current value of the TPM_SC_TOIE field. */  
#define TPM_RD_SC_TOIE(base) ((TPM_SC_REG(base) & TPM_SC_TOIE_MASK) >>  
TPM_SC_TOIE_SHIFT)  
#define TPM_BRD_SC_TOIE(base) (BME_UBFX32(&TPM_SC_REG(base),  
TPM_SC_TOIE_SHIFT, TPM_SC_TOIE_WIDTH))

/*! @brief Set the TOIE field to a new value. */
```

```

#define TPM_WR_SC_TOIE(base, value) (TPM_RMW_SC(base, (TPM_SC_TOIE_MASK |  
TPM_SC_TOF_MASK), TPM_SC_TOIE(value)))  
#define TPM_BWR_SC_TOIE(base, value) (BME_BFI32(&TPM_SC_REG(base),  
(uint32_t)(value) << TPM_SC_TOIE_SHIFT), TPM_SC_TOIE_SHIFT,  
TPM_SC_TOIE_WIDTH))  
/*@}*/

/*!  
 * @name Register TPM_SC, field TOF[7] (W1C)  
 *  
 * Set by hardware when the LPTPM counter equals the value in the MOD  
register  
 * and increments. The TOF bit is cleared by writing a 1 to TOF bit.  
Writing a 0  
 * to TOF has no effect. If another LPTPM overflow occurs between the  
flag setting  
 * and the flag clearing, the write operation has no effect; therefore,  
TOF  
 * remains set indicating another overflow has occurred. In this case a  
TOF interrupt  
 * request is not lost due to a delay in clearing the previous TOF.  
*  
* Values:  
* - 0b0 - LPTPM counter has not overflowed.  
* - 0b1 - LPTPM counter has overflowed.  
*/  
/*@{*/  
/*! @brief Read current value of the TPM_SC_TOF field. */  
#define TPM_RD_SC_TOF(base) ((TPM_SC_REG(base) & TPM_SC_TOF_MASK) >>  
TPM_SC_TOF_SHIFT)  
#define TPM_BRD_SC_TOF(base) (BME_UBFX32(&TPM_SC_REG(base),  
TPM_SC_TOF_SHIFT, TPM_SC_TOF_WIDTH))

/*! @brief Set the TOF field to a new value. */  
#define TPM_WR_SC_TOF(base, value) (TPM_RMW_SC(base, TPM_SC_TOF_MASK,  
TPM_SC_TOF(value)))  
#define TPM_BWR_SC_TOF(base, value) (BME_BFI32(&TPM_SC_REG(base),  
(uint32_t)(value) << TPM_SC_TOF_SHIFT), TPM_SC_TOF_SHIFT,  
TPM_SC_TOF_WIDTH))  
/*@}*/

/*!  
 * @name Register TPM_SC, field DMA[8] (RW)  
 *  
 * Enables DMA transfers for the overflow flag.  
*  
* Values:  
* - 0b0 - Disables DMA transfers.  
* - 0b1 - Enables DMA transfers.  
*/  
/*@{*/  
/*! @brief Read current value of the TPM_SC_DMA field. */  
#define TPM_RD_SC_DMA(base) ((TPM_SC_REG(base) & TPM_SC_DMA_MASK) >>  
TPM_SC_DMA_SHIFT)

```

```

#define TPM_BRD_SC_DMA(base)  (BME_UBFX32(&TPM_SC_REG(base),  

TPM_SC_DMA_SHIFT, TPM_SC_DMA_WIDTH))

/*! @brief Set the DMA field to a new value. */  

#define TPM_WR_SC_DMA(base, value) (TPM_RMW_SC(base, (TPM_SC_DMA_MASK |  

TPM_SC_TOF_MASK), TPM_SC_DMA(value)))  

#define TPM_BWR_SC_DMA(base, value) (BME_BFI32(&TPM_SC_REG(base),  

(uint32_t)(value) << TPM_SC_DMA_SHIFT), TPM_SC_DMA_SHIFT,  

TPM_SC_DMA_WIDTH)  

/*@}*/

/*********************  

*****  

 * TPM_CNT - Counter  

*****  

/*********************  

****/  

  

/*!  

 * @brief TPM_CNT - Counter (RW)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The CNT register contains the LPTPM counter value. Reset clears the  

CNT  

 * register. Writing any value to COUNT also clears the counter. When  

debug is active,  

 * the LPTPM counter does not increment unless configured otherwise.  

Reading the  

 * CNT register adds two wait states to the register access due to  

 * synchronization delays.  

*/  

/*!  

 * @name Constants and macros for entire TPM_CNT register  

*/  

/*@*/  

#define TPM_RD_CNT(base)          (TPM_CNT_REG(base))  

#define TPM_WR_CNT(base, value)   (TPM_CNT_REG(base) = (value))  

#define TPM_RMW_CNT(base, mask, value) (TPM_WR_CNT(base,  

(TPM_RD_CNT(base) & ~mask) | (value)))  

#define TPM_SET_CNT(base, value)  (BME_OR32(&TPM_CNT_REG(base),  

(uint32_t)(value)))  

#define TPM_CLR_CNT(base, value)  (BME_AND32(&TPM_CNT_REG(base),  

(uint32_t)(~(value))))  

#define TPM_TOG_CNT(base, value)  (BME_XOR32(&TPM_CNT_REG(base),  

(uint32_t)(value)))  

/*@*/  

  

/*  

 * Constants & macros for individual TPM_CNT bitfields  

*/  

  

/*!  

 * @name Register TPM_CNT, field COUNT[15:0] (RW)

```

```

        */
/*@{*/
/*! @brief Read current value of the TPM_CNT_COUNT field. */
#define TPM_RD_CNT_COUNT(base) ((TPM_CNT_REG(base) & TPM_CNT_COUNT_MASK)
>> TPM_CNT_COUNT_SHIFT)
#define TPM_BRD_CNT_COUNT(base) (BME_UBFX32(&TPM_CNT_REG(base),
TPM_CNT_COUNT_SHIFT, TPM_CNT_COUNT_WIDTH))

/*! @brief Set the COUNT field to a new value. */
#define TPM_WR_CNT_COUNT(base, value) (TPM_RMW_CNT(base,
TPM_CNT_COUNT_MASK, TPM_CNT_COUNT(value)))
#define TPM_BWR_CNT_COUNT(base, value) (BME_BFI32(&TPM_CNT_REG(base),
((uint32_t)(value) << TPM_CNT_COUNT_SHIFT), TPM_CNT_COUNT_SHIFT,
TPM_CNT_COUNT_WIDTH))
/*}@*/



/********************* TPM_MOD - Modulo *********************



* TPM_MOD - Modulo

*****/



/*!
 * @brief TPM_MOD - Modulo (RW)
 *
 * Reset value: 0x0000FFFFU
 *
 * The Modulo register contains the modulo value for the LPTPM counter.
When the
 * LPTPM counter reaches the modulo value and increments, the overflow
flag
 * (TOF) is set and the next value of LPTPM counter depends on the
selected counting
 * method (see Counter ). Writing to the MOD register latches the value
into a
 * buffer. The MOD register is updated with the value of its write buffer
according
 * to MOD Register Update . It is recommended to initialize the LPTPM
counter
 * (write to CNT) before writing to the MOD register to avoid confusion
about when
 * the first counter overflow will occur.
 */
/*!
 * @name Constants and macros for entire TPM_MOD register
 */
/*@{*/
#define TPM_RD_MOD(base)          (TPM_MOD_REG(base))
#define TPM_WR_MOD(base, value)   (TPM_MOD_REG(base) = (value))
#define TPM_RMW_MOD(base, mask, value) (TPM_WR_MOD(base,
(TPM_RD_MOD(base) & ~mask) | (value)))
#define TPM_SET_MOD(base, value)  (BME_OR32(&TPM_MOD_REG(base),
(uint32_t)(value)))

```

```

#define TPM_CLR_MOD(base, value) (BME_AND32(&TPM_MOD_REG(base),  

(uint32_t)(~(value))))  

#define TPM_TOG_MOD(base, value) (BME_XOR32(&TPM_MOD_REG(base),  

(uint32_t)(value)))  

/*@*/
```

/\*
 \* Constants & macros for individual TPM\_MOD bitfields
 \*/

```

/*!  

 * @name Register TPM_MOD, field MOD[15:0] (RW)  

 * When writing this field, all bytes must be written at the same time.  

 */  

/*@{*/  

/*! @brief Read current value of the TPM_MOD_MOD field. */  

#define TPM_RD_MOD_MOD(base) ((TPM_MOD_REG(base) & TPM_MOD_MOD_MASK) >>  

TPM_MOD_MOD_SHIFT)  

#define TPM_BRD_MOD_MOD(base) (BME_UBFX32(&TPM_MOD_REG(base),  

TPM_MOD_MOD_SHIFT, TPM_MOD_MOD_WIDTH))
```

```

/*! @brief Set the MOD field to a new value. */  

#define TPM_WR_MOD_MOD(base, value) (TPM_RMW_MOD(base, TPM_MOD_MOD_MASK,  

TPM_MOD_MOD(value)))  

#define TPM_BWR_MOD_MOD(base, value) (BME_BFI32(&TPM_MOD_REG(base),  

((uint32_t)(value) << TPM_MOD_MOD_SHIFT), TPM_MOD_MOD_SHIFT,  

TPM_MOD_MOD_WIDTH))  

/*@*/
```

\*\*\*\*\*  
\*\*\*\*\*  
\* TPM\_CnSC - Channel (n) Status and Control  
\*\*\*\*\*  
\*\*\*\*\* /

```

/*!  

 * @brief TPM_CnSC - Channel (n) Status and Control (RW)  

 * Reset value: 0x00000000U  

 * CnSC contains the channel-interrupt-status flag and control bits used  

to  

* configure the interrupt enable, channel configuration, and pin  

function. When  

* switching from one channel mode to a different channel mode, the  

channel must  

* first be disabled and this must be acknowledged in the LPTPM counter  

clock domain.  

* Mode, Edge, and Level Selection CPWMS MSnB:MSnA ELSnB:ELSnA Mode  

* Configuration X 00 00 None Channel disabled X 01/10/11 00 Software  

compare Pin not used
```

```

 * for LPTPM 0 00 01 Input capture Capture on Rising Edge Only 10 Capture
on
 * Falling Edge Only 11 Capture on Rising or Falling Edge 01 01 Output
compare Toggle
 * Output on match 10 Clear Output on match 11 Set Output on match 10 10
 * Edge-aligned PWM High-true pulses (clear Output on match, set Output
on reload) X1
 * Low-true pulses (set Output on match, clear Output on reload) 11 10
Output compare
 * Pulse Output low on match X1 Pulse Output high on match 1 10 10
 * Center-aligned PWM High-true pulses (clear Output on match-up, set
Output on match-down) X1
 * Low-true pulses (set Output on match-up, clear Output on match-down)
*/
/*!
 * @name Constants and macros for entire TPM_CnSC register
 */
/*@{*/
#define TPM_RD_CnSC(base, index) (TPM_CnSC_REG(base, index))
#define TPM_WR_CnSC(base, index, value) (TPM_CnSC_REG(base, index) = (value))
#define TPM_RMW_CnSC(base, index, mask, value) (TPM_WR_CnSC(base, index,
(TPM_RD_CnSC(base, index) & ~mask)) | (value)))
#define TPM_SET_CnSC(base, index, value) (BME_OR32(&TPM_CnSC_REG(base,
index), (uint32_t)(value)))
#define TPM_CLR_CnSC(base, index, value) (BME_AND32(&TPM_CnSC_REG(base,
index), (uint32_t)(~value)))
#define TPM_TOG_CnSC(base, index, value) (BME_XOR32(&TPM_CnSC_REG(base,
index), (uint32_t)(value)))
/*@}*/
/*
 * Constants & macros for individual TPM_CnSC bitfields
 */
/*!
 * @name Register TPM_CnSC, field DMA[0] (RW)
 *
 * Enables DMA transfers for the channel.
 *
 * Values:
 * - 0b0 - Disable DMA transfers.
 * - 0b1 - Enable DMA transfers.
 */
/*@*/
/*! @brief Read current value of the TPM_CnSC_DMA field. */
#define TPM_RD_CnSC_DMA(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_DMA_MASK) >> TPM_CnSC_DMA_SHIFT)
#define TPM_BRD_CnSC_DMA(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_DMA_SHIFT, TPM_CnSC_DMA_WIDTH))

/*! @brief Set the DMA field to a new value. */
#define TPM_WR_CnSC_DMA(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_DMA_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_DMA(value)))

```

```

#define TPM_BWR_CnSC_DMA(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_DMA_SHIFT), TPM_CnSC_DMA_SHIFT, TPM_CnSC_DMA_WIDTH))
/*@}*/

/*!
 * @name Register TPM_CnSC, field ELSA[2] (RW)
 *
 * The functionality of ELSB and ELSA depends on the channel mode. When a
 * channel is disabled, this bit will not change state until acknowledged
in the LPTPM
 * counter clock domain.
 */
/*@{*/
/*! @brief Read current value of the TPM_CnSC_ELSA field. */
#define TPM_RD_CnSC_ELSA(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_ELSA_MASK) >> TPM_CnSC_ELSA_SHIFT)
#define TPM_BRD_CnSC_ELSA(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_ELSA_SHIFT, TPM_CnSC_ELSA_WIDTH))

/*! @brief Set the ELSA field to a new value. */
#define TPM_WR_CnSC_ELSA(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_ELSA_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_ELSA(value)))
#define TPM_BWR_CnSC_ELSA(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_ELSA_SHIFT), TPM_CnSC_ELSA_SHIFT, TPM_CnSC_ELSA_WIDTH))
/*@}*/

/*!
 * @name Register TPM_CnSC, field ELSB[3] (RW)
 *
 * The functionality of ELSB and ELSA depends on the channel mode. When a
 * channel is disabled, this bit will not change state until acknowledged
in the LPTPM
 * counter clock domain.
 */
/*@{*/
/*! @brief Read current value of the TPM_CnSC_ELSB field. */
#define TPM_RD_CnSC_ELSB(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_ELSB_MASK) >> TPM_CnSC_ELSB_SHIFT)
#define TPM_BRD_CnSC_ELSB(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_ELSB_SHIFT, TPM_CnSC_ELSB_WIDTH))

/*! @brief Set the ELSB field to a new value. */
#define TPM_WR_CnSC_ELSB(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_ELSB_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_ELSB(value)))
#define TPM_BWR_CnSC_ELSB(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_ELSB_SHIFT), TPM_CnSC_ELSB_SHIFT, TPM_CnSC_ELSB_WIDTH))
/*@}*/

/*!
 * @name Register TPM_CnSC, field MSA[4] (RW)
 *

```

```

    * Used for further selections in the channel logic. Its functionality is
    * dependent on the channel mode. When a channel is disabled, this bit
will not change
    * state until acknowledged in the LPTPM counter clock domain.
    */
/*@{*/
/*! @brief Read current value of the TPM_CnSC_MSA field. */
#define TPM_RD_CnSC_MSA(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_MSA_MASK) >> TPM_CnSC_MSA_SHIFT)
#define TPM_BRD_CnSC_MSA(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_MSA_SHIFT, TPM_CnSC_MSA_WIDTH))

/*! @brief Set the MSA field to a new value. */
#define TPM_WR_CnSC_MSA(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_MSA_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_MSA(value)))
#define TPM_BWR_CnSC_MSA(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_MSA_SHIFT), TPM_CnSC_MSA_SHIFT, TPM_CnSC_MSA_WIDTH))
/*@}*/

/*!
* @name Register TPM_CnSC, field MSB[5] (RW)
*
* Used for further selections in the channel logic. Its functionality is
* dependent on the channel mode. When a channel is disabled, this bit
will not change
* state until acknowledged in the LPTPM counter clock domain.
*/
/*@{*/
/*! @brief Read current value of the TPM_CnSC_MSB field. */
#define TPM_RD_CnSC_MSB(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_MSB_MASK) >> TPM_CnSC_MSB_SHIFT)
#define TPM_BRD_CnSC_MSB(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_MSB_SHIFT, TPM_CnSC_MSB_WIDTH))

/*! @brief Set the MSB field to a new value. */
#define TPM_WR_CnSC_MSB(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_MSB_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_MSB(value)))
#define TPM_BWR_CnSC_MSB(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_MSB_SHIFT), TPM_CnSC_MSB_SHIFT, TPM_CnSC_MSB_WIDTH))
/*@}*/

/*!
* @name Register TPM_CnSC, field CHIE[6] (RW)
*
* Enables channel interrupts.
*
* Values:
* - 0b0 - Disable channel interrupts.
* - 0b1 - Enable channel interrupts.
*/
/*@{*/
/*! @brief Read current value of the TPM_CnSC_CHIE field. */

```

```

#define TPM_RD_CnSC_CHIE(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_CHIE_MASK) >> TPM_CnSC_CHIE_SHIFT)
#define TPM_BRD_CnSC_CHIE(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_CHIE_SHIFT, TPM_CnSC_CHIE_WIDTH))

/*! @brief Set the CHIE field to a new value. */
#define TPM_WR_CnSC_CHIE(base, index, value) (TPM_RMW_CnSC(base, index,
(TPM_CnSC_CHIE_MASK | TPM_CnSC_CHF_MASK), TPM_CnSC_CHIE(value)))
#define TPM_BWR_CnSC_CHIE(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_CHIE_SHIFT), TPM_CnSC_CHIE_SHIFT, TPM_CnSC_CHIE_WIDTH))
/*@}*/

/*
 * @name Register TPM_CnSC, field CHF[7] (W1C)
 *
 * Set by hardware when an event occurs on the channel. CHF is cleared by
 * writing a 1 to the CHF bit. Writing a 0 to CHF has no effect. If
another event
 * occurs between the CHF sets and the write operation, the write
operation has no
 * effect; therefore, CHF remains set indicating another event has
occurred. In this
 * case a CHF interrupt request is not lost due to the delay in clearing
the
 * previous CHF.
 *
 * Values:
 * - 0b0 - No channel event has occurred.
 * - 0b1 - A channel event has occurred.
 */
/*@{*/
/*! @brief Read current value of the TPM_CnSC_CHF field. */
#define TPM_RD_CnSC_CHF(base, index) ((TPM_CnSC_REG(base, index) &
TPM_CnSC_CHF_MASK) >> TPM_CnSC_CHF_SHIFT)
#define TPM_BRD_CnSC_CHF(base, index) (BME_UBFX32(&TPM_CnSC_REG(base,
index), TPM_CnSC_CHF_SHIFT, TPM_CnSC_CHF_WIDTH))

/*! @brief Set the CHF field to a new value. */
#define TPM_WR_CnSC_CHF(base, index, value) (TPM_RMW_CnSC(base, index,
TPM_CnSC_CHF_MASK, TPM_CnSC_CHF(value)))
#define TPM_BWR_CnSC_CHF(base, index, value)
(BME_BFI32(&TPM_CnSC_REG(base, index), ((uint32_t)(value) <<
TPM_CnSC_CHF_SHIFT), TPM_CnSC_CHF_SHIFT, TPM_CnSC_CHF_WIDTH))
/*@}*/

*****
***** TPM_CnV - Channel (n) Value
***** /
/*!

```

```

* @brief TPM_CnV - Channel (n) Value (RW)
*
* Reset value: 0x00000000U
*
* These registers contain the captured LPTPM counter value for the input
modes
* or the match value for the output modes. In input capture mode, any
write to a
* CnV register is ignored. In compare modes, writing to a CnV register
latches
* the value into a buffer. A CnV register is updated with the value of
its write
* buffer according to CnV Register Update .
*/
/*!
* @name Constants and macros for entire TPM_CnV register
*/
/*@{*/
#define TPM_RD_CnV(base, index)  (TPM_CnV_REG(base, index))
#define TPM_WR_CnV(base, index, value) (TPM_CnV_REG(base, index) =
(value))
#define TPM_RMW_CnV(base, index, mask, value) (TPM_WR_CnV(base, index,
(TPM_RD_CnV(base, index) & ~mask) | (value)))
#define TPM_SET_CnV(base, index, value) (BME_OR32(&TPM_CnV_REG(base,
index), (uint32_t)(value)))
#define TPM_CLR_CnV(base, index, value) (BME_AND32(&TPM_CnV_REG(base,
index), (uint32_t)(~(value))))
#define TPM_TOG_CnV(base, index, value) (BME_XOR32(&TPM_CnV_REG(base,
index), (uint32_t)(value)))
/*@}*/
/*
* Constants & macros for individual TPM_CnV bitfields
*/
/*!
* @name Register TPM_CnV, field VAL[15:0] (RW)
*
* Captured LPTPM counter value of the input modes or the match value for
the
* output modes. When writing this field, all bytes must be written at
the same
* time.
*/
/*@{*/
/*! @brief Read current value of the TPM_CnV_VAL field. */
#define TPM_RD_CnV_VAL(base, index) ((TPM_CnV_REG(base, index) &
TPM_CnV_VAL_MASK) >> TPM_CnV_VAL_SHIFT)
#define TPM_BRD_CnV_VAL(base, index) (BME_UBFX32(&TPM_CnV_REG(base,
index), TPM_CnV_VAL_SHIFT, TPM_CnV_VAL_WIDTH))

/*! @brief Set the VAL field to a new value. */
#define TPM_WR_CnV_VAL(base, index, value) (TPM_RMW_CnV(base, index,
TPM_CnV_VAL_MASK, TPM_CnV_VAL(value)))

```

```

#define TPM_BWR_CnV_VAL(base, index, value) (BME_BFI32(&TPM_CnV_REG(base),  

index), ((uint32_t)(value) << TPM_CnV_VAL_SHIFT), TPM_CnV_VAL_SHIFT,  

TPM_CnV_VAL_WIDTH))  

/*@}*/

/*********************  

*****  

 * TPM_STATUS - Capture and Compare Status  

*****/  

  

/*!  

 * @brief TPM_STATUS - Capture and Compare Status (RW)  

 *  

 * Reset value: 0x00000000U  

 *  

 * The STATUS register contains a copy of the status flag CHnF bit (in  

CnSC) for  

 * each LPTPM channel, as well as the TOF bit (in SC), for software  

convenience.  

 * Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits  

can be  

 * checked using only one read of STATUS. All CHnF bits can be cleared by  

 * writing all ones to STATUS. Hardware sets the individual channel flags  

when an event  

 * occurs on the channel. CHF is cleared by writing a 1 to the CHF bit.  

Writing  

 * a 0 to CHF has no effect. If another event occurs between the flag  

setting and  

 * the write operation, the write operation has no effect; therefore, CHF  

 * remains set indicating another event has occurred. In this case a CHF  

interrupt  

 * request is not lost due to the clearing sequence for a previous CHF.  

 */  

/*!  

 * @name Constants and macros for entire TPM_STATUS register  

*/  

/*@{*/  

#define TPM_RD_STATUS(base) (TPM_STATUS_REG(base))  

#define TPM_WR_STATUS(base, value) (TPM_STATUS_REG(base) = (value))  

#define TPM_RMW_STATUS(base, mask, value) (TPM_WR_STATUS(base,  

(TPM_RD_STATUS(base) & ~mask) | (value)))  

#define TPM_SET_STATUS(base, value) (BME_OR32(&TPM_STATUS_REG(base),  

(uint32_t)(value)))  

#define TPM_CLR_STATUS(base, value) (BME_AND32(&TPM_STATUS_REG(base),  

(uint32_t)(~(value))))  

#define TPM_TOG_STATUS(base, value) (BME_XOR32(&TPM_STATUS_REG(base),  

(uint32_t)(value)))  

/*@}/
```

/\*  
 \* Constants & macros for individual TPM\_STATUS bitfields  
\*/

```

/*!
 * @name Register TPM_STATUS, field CH0F[0] (W1C)
 *
 * See the register description.
 *
 * Values:
 * - 0b0 - No channel event has occurred.
 * - 0b1 - A channel event has occurred.
 */
/*@{*/
/*! @brief Read current value of the TPM_STATUS_CH0F field. */
#define TPM_RD_STATUS_CH0F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH0F_MASK) >> TPM_STATUS_CH0F_SHIFT)
#define TPM_BRD_STATUS_CH0F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH0F_SHIFT, TPM_STATUS_CH0F_WIDTH))

/*! @brief Set the CH0F field to a new value. */
#define TPM_WR_STATUS_CH0F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK | TPM_STATUS_CH2F_MASK |
TPM_STATUS_CH3F_MASK | TPM_STATUS_CH4F_MASK | TPM_STATUS_CH5F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH0F(value)))
#define TPM_BWR_STATUS_CH0F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH0F_SHIFT), TPM_STATUS_CH0F_SHIFT, TPM_STATUS_CH0F_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register TPM_STATUS, field CH1F[1] (W1C)
 *
 * See the register description.
 *
 * Values:
 * - 0b0 - No channel event has occurred.
 * - 0b1 - A channel event has occurred.
 */
/*@{*/
/*! @brief Read current value of the TPM_STATUS_CH1F field. */
#define TPM_RD_STATUS_CH1F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH1F_MASK) >> TPM_STATUS_CH1F_SHIFT)
#define TPM_BRD_STATUS_CH1F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH1F_SHIFT, TPM_STATUS_CH1F_WIDTH))

/*! @brief Set the CH1F field to a new value. */
#define TPM_WR_STATUS_CH1F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH1F_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH2F_MASK |
TPM_STATUS_CH3F_MASK | TPM_STATUS_CH4F_MASK | TPM_STATUS_CH5F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH1F(value)))
#define TPM_BWR_STATUS_CH1F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH1F_SHIFT), TPM_STATUS_CH1F_SHIFT, TPM_STATUS_CH1F_WIDTH))
/*}@*/
```

  

```

/*!
```

```

* @name Register TPM_STATUS, field CH2F[2] (W1C)
*
* See the register description.
*
* Values:
* - 0b0 - No channel event has occurred.
* - 0b1 - A channel event has occurred.
*/
/*@{/ */
/*! @brief Read current value of the TPM_STATUS_CH2F field. */
#define TPM_RD_STATUS_CH2F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH2F_MASK) >> TPM_STATUS_CH2F_SHIFT)
#define TPM_BRD_STATUS_CH2F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH2F_SHIFT, TPM_STATUS_CH2F_WIDTH))

/*! @brief Set the CH2F field to a new value. */
#define TPM_WR_STATUS_CH2F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH2F_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK |
TPM_STATUS_CH3F_MASK | TPM_STATUS_CH4F_MASK | TPM_STATUS_CH5F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH2F(value)))
#define TPM_BWR_STATUS_CH2F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH2F_SHIFT), TPM_STATUS_CH2F_SHIFT, TPM_STATUS_CH2F_WIDTH))
/*@{/ */

/*!
* @name Register TPM_STATUS, field CH3F[3] (W1C)
*
* See the register description.
*
* Values:
* - 0b0 - No channel event has occurred.
* - 0b1 - A channel event has occurred.
*/
/*@{/ */
/*! @brief Read current value of the TPM_STATUS_CH3F field. */
#define TPM_RD_STATUS_CH3F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH3F_MASK) >> TPM_STATUS_CH3F_SHIFT)
#define TPM_BRD_STATUS_CH3F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH3F_SHIFT, TPM_STATUS_CH3F_WIDTH))

/*! @brief Set the CH3F field to a new value. */
#define TPM_WR_STATUS_CH3F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH3F_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK |
TPM_STATUS_CH2F_MASK | TPM_STATUS_CH4F_MASK | TPM_STATUS_CH5F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH3F(value)))
#define TPM_BWR_STATUS_CH3F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH3F_SHIFT), TPM_STATUS_CH3F_SHIFT, TPM_STATUS_CH3F_WIDTH))
/*@{/ */

/*!
* @name Register TPM_STATUS, field CH4F[4] (W1C)
*

```

```

* See the register description.
*
* Values:
* - 0b0 - No channel event has occurred.
* - 0b1 - A channel event has occurred.
*/
/*@{ */
/*! @brief Read current value of the TPM_STATUS_CH4F field. */
#define TPM_RD_STATUS_CH4F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH4F_MASK) >> TPM_STATUS_CH4F_SHIFT)
#define TPM_BRD_STATUS_CH4F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH4F_SHIFT, TPM_STATUS_CH4F_WIDTH))

/*! @brief Set the CH4F field to a new value. */
#define TPM_WR_STATUS_CH4F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH4F_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK |
TPM_STATUS_CH2F_MASK | TPM_STATUS_CH3F_MASK | TPM_STATUS_CH5F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH4F(value)))
#define TPM_BWR_STATUS_CH4F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH4F_SHIFT), TPM_STATUS_CH4F_SHIFT, TPM_STATUS_CH4F_WIDTH))
/*}@*/ */

/*!
* @name Register TPM_STATUS, field CH5F[5] (W1C)
*
* See the register description.
*
* Values:
* - 0b0 - No channel event has occurred.
* - 0b1 - A channel event has occurred.
*/
/*@{ */
/*! @brief Read current value of the TPM_STATUS_CH5F field. */
#define TPM_RD_STATUS_CH5F(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_CH5F_MASK) >> TPM_STATUS_CH5F_SHIFT)
#define TPM_BRD_STATUS_CH5F(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_CH5F_SHIFT, TPM_STATUS_CH5F_WIDTH))

/*! @brief Set the CH5F field to a new value. */
#define TPM_WR_STATUS_CH5F(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_CH5F_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK |
TPM_STATUS_CH2F_MASK | TPM_STATUS_CH3F_MASK | TPM_STATUS_CH4F_MASK |
TPM_STATUS_TOF_MASK), TPM_STATUS_CH5F(value)))
#define TPM_BWR_STATUS_CH5F(base, value)
(BME_BFI32(&TPM_STATUS_REG(base), ((uint32_t)(value) <<
TPM_STATUS_CH5F_SHIFT), TPM_STATUS_CH5F_SHIFT, TPM_STATUS_CH5F_WIDTH))
/*}@*/ */

/*!
* @name Register TPM_STATUS, field TOF[8] (W1C)
*
* See register description
*/

```

```

* Values:
* - 0b0 - LPTPM counter has not overflowed.
* - 0b1 - LPTPM counter has overflowed.
*/
/*@{ */
/*! @brief Read current value of the TPM_STATUS_TOF field. */
#define TPM_RD_STATUS_TOF(base) ((TPM_STATUS_REG(base) &
TPM_STATUS_TOF_MASK) >> TPM_STATUS_TOF_SHIFT)
#define TPM_BRD_STATUS_TOF(base) (BME_UBFX32(&TPM_STATUS_REG(base),
TPM_STATUS_TOF_SHIFT, TPM_STATUS_TOF_WIDTH))

/*! @brief Set the TOF field to a new value. */
#define TPM_WR_STATUS_TOF(base, value) (TPM_RMW_STATUS(base,
(TPM_STATUS_TOF_MASK | TPM_STATUS_CH0F_MASK | TPM_STATUS_CH1F_MASK |
TPM_STATUS_CH2F_MASK | TPM_STATUS_CH3F_MASK | TPM_STATUS_CH4F_MASK |
TPM_STATUS_CH5F_MASK), TPM_STATUS_TOF(value)))
#define TPM_BWR_STATUS_TOF(base, value) (BME_BFI32(&TPM_STATUS_REG(base),
((uint32_t)(value) << TPM_STATUS_TOF_SHIFT), TPM_STATUS_TOF_SHIFT,
TPM_STATUS_TOF_WIDTH))
/*@} */

/********************* TPM_CONF - Configuration ********************
*****
* TPM_CONF - Configuration
*****
/*
* @brief TPM_CONF - Configuration (RW)
*
* Reset value: 0x00000000U
*
* This register selects the behavior in debug and wait modes and the use
of an
* external global time base.
*/
/*
* @name Constants and macros for entire TPM_CONF register
*/
/*@{ */
#define TPM_RD_CONF(base) (TPM_CONF_REG(base))
#define TPM_WR_CONF(base, value) (TPM_CONF_REG(base) = (value))
#define TPM_RMW_CONF(base, mask, value) (TPM_WR_CONF(base,
(TPM_RD_CONF(base) & ~mask) | value)))
#define TPM_SET_CONF(base, value) (BME_OR32(&TPM_CONF_REG(base),
(uint32_t)(value)))
#define TPM_CLR_CONF(base, value) (BME_AND32(&TPM_CONF_REG(base),
(uint32_t)(~value))))
#define TPM_TOG_CONF(base, value) (BME_XOR32(&TPM_CONF_REG(base),
(uint32_t)(value)))
/*@} */

/*

```

```

 * Constants & macros for individual TPM_CONF bitfields
 */

/*!
 * @name Register TPM_CONF, field DOZEEN[5] (RW)
 *
 * Configures the LPTPM behavior in wait mode.
 *
 * Values:
 * - 0b0 - Internal LPTPM counter continues in Doze mode.
 * - 0b1 - Internal LPTPM counter is paused and does not increment during
Doze
 *      mode. Trigger inputs and input capture events are also ignored.
 */
/*@{*/
/*! @brief Read current value of the TPM_CONF_DOZEEN field. */
#define TPM_RD_CONF_DOZEEN(base) ((TPM_CONF_REG(base) &
TPM_CONF_DOZEEN_MASK) >> TPM_CONF_DOZEEN_SHIFT)
#define TPM_BRD_CONF_DOZEEN(base) (BME_UBFX32(&TPM_CONF_REG(base),
TPM_CONF_DOZEEN_SHIFT, TPM_CONF_DOZEEN_WIDTH))

/*! @brief Set the DOZEEN field to a new value. */
#define TPM_WR_CONF_DOZEEN(base, value) (TPM_RMW_CONF(base,
TPM_CONF_DOZEEN_MASK, TPM_CONF_DOZEEN(value)))
#define TPM_BWR_CONF_DOZEEN(base, value) (BME_BFI32(&TPM_CONF_REG(base),
((uint32_t)(value) << TPM_CONF_DOZEEN_SHIFT), TPM_CONF_DOZEEN_SHIFT,
TPM_CONF_DOZEEN_WIDTH))
/*@}*/

/*!
 * @name Register TPM_CONF, field DBGMODE[7:6] (RW)
 *
 * Configures the LPTPM behavior in debug mode. All other configurations
are
 * reserved.
 *
 * Values:
 * - 0b00 - LPTPM counter is paused and does not increment during debug
mode.
 *      Trigger inputs and input capture events are also ignored.
 * - 0b11 - LPTPM counter continues in debug mode.
 */
/*@{*/
/*! @brief Read current value of the TPM_CONF_DBGMODE field. */
#define TPM_RD_CONF_DBGMODE(base) ((TPM_CONF_REG(base) &
TPM_CONF_DBGMODE_MASK) >> TPM_CONF_DBGMODE_SHIFT)
#define TPM_BRD_CONF_DBGMODE(base) (BME_UBFX32(&TPM_CONF_REG(base),
TPM_CONF_DBGMODE_SHIFT, TPM_CONF_DBGMODE_WIDTH))

/*! @brief Set the DBGMODE field to a new value. */
#define TPM_WR_CONF_DBGMODE(base, value) (TPM_RMW_CONF(base,
TPM_CONF_DBGMODE_MASK, TPM_CONF_DBGMODE(value)))

```

```

#define TPM_BWR_CONF_DBGMODE(base, value) (BME_BFI32(&TPM_CONF_REG(base), \
((uint32_t)(value) << TPM_CONF_DBGMODE_SHIFT), TPM_CONF_DBGMODE_SHIFT, \
TPM_CONF_DBGMODE_WIDTH))
/*@}*/

/*!
 * @name Register TPM_CONF, field GTBEEN[9] (RW)
 *
 * Configures the LPTPM to use an externally generated global time base
counter.
 * When an externally generated timebase is used, the internal LPTPM
counter is
 * not used by the channels but can be used to generate a periodic
interrupt or
 * DMA request using the Modulo register and timer overflow flag.
 *
 * Values:
 * - 0b0 - All channels use the internally generated LPTPM counter as
their
 *   timebase
 * - 0b1 - All channels use an externally generated global timebase as
their
 *   timebase
 */
/*@*/
/*! @brief Read current value of the TPM_CONF_GTBEEN field. */
#define TPM_RD_CONF_GTBEEN(base) ((TPM_CONF_REG(base) &
TPM_CONF_GTBEEN_MASK) >> TPM_CONF_GTBEEN_SHIFT)
#define TPM_BRD_CONF_GTBEEN(base) (BME_UBFX32(&TPM_CONF_REG(base),
TPM_CONF_GTBEEN_SHIFT, TPM_CONF_GTBEEN_WIDTH))

/*! @brief Set the GTBEEN field to a new value. */
#define TPM_WR_CONF_GTBEEN(base, value) (TPM_RMW_CONF(base,
TPM_CONF_GTBEEN_MASK, TPM_CONF_GTBEEN(value)))
#define TPM_BWR_CONF_GTBEEN(base, value) (BME_BFI32(&TPM_CONF_REG(base),
((uint32_t)(value) << TPM_CONF_GTBEEN_SHIFT), TPM_CONF_GTBEEN_SHIFT,
TPM_CONF_GTBEEN_WIDTH))
/*@*/

/*!
 * @name Register TPM_CONF, field CSOT[16] (RW)
 *
 * When set, the LPTPM counter will not start incrementing after it is
enabled
 * until a rising edge on the selected trigger input is detected. If the
LPTPM
 * counter is stopped due to an overflow, a rising edge on the selected
trigger
 * input will also cause the LPTPM counter to start incrementing again.
The trigger
 * input is ignored if the LPTPM counter is paused during debug mode or
doze mode.
 * This field should only be changed when the LPTPM counter is disabled.
 *

```

```

 * Values:
 * - 0b0 - LPTPM counter starts to increment immediately, once it is
enabled.
 * - 0b1 - LPTPM counter only starts to increment when it a rising edge
on the
 *      selected input trigger is detected, after it has been enabled or
after it
 *      has stopped due to overflow.
 */
/*@{*/
/*! @brief Read current value of the TPM_CONF_CSOT field. */
#define TPM_RD_CONF_CSOT(base) ((TPM_CONF_REG(base) & TPM_CONF_CSOT_MASK)
>> TPM_CONF_CSOT_SHIFT)
#define TPM_BRD_CONF_CSOT(base) (BME_UBFX32(&TPM_CONF_REG(base),
TPM_CONF_CSOT_SHIFT, TPM_CONF_CSOT_WIDTH))

/*! @brief Set the CSOT field to a new value. */
#define TPM_WR_CONF_CSOT(base, value) (TPM_RMW_CONF(base,
TPM_CONF_CSOT_MASK, TPM_CONF_CSOT(value)))
#define TPM_BWR_CONF_CSOT(base, value) (BME_BFI32(&TPM_CONF_REG(base),
((uint32_t)(value) << TPM_CONF_CSOT_SHIFT), TPM_CONF_CSOT_SHIFT,
TPM_CONF_CSOT_WIDTH))
/*}@*/ */

/*!
 * @name Register TPM_CONF, field CSOO[17] (RW)
 *
 * When set, the LPTPM counter will stop incrementing once the counter
equals
 * the MOD value and incremented (this also sets the TOF). Reloading the
counter
 * with zero due to writing to the counter register or due to a trigger
input does
 * not cause the counter to stop incrementing. Once the counter has
stopped
 * incrementing, the counter will not start incrementing unless it is
disabled and
 * then enabled again, or a rising edge on the selected trigger input is
detected
 * when CSOT set. This field should only be changed when the LPTPM
counter is
 * disabled.
 *
 * Values:
 * - 0b0 - LPTPM counter continues incrementing or decrementing after
overflow
 * - 0b1 - LPTPM counter stops incrementing or decrementing after
overflow.
 */
/*@{*/
/*! @brief Read current value of the TPM_CONF_CSOO field. */
#define TPM_RD_CONF_CSOO(base) ((TPM_CONF_REG(base) & TPM_CONF_CSOO_MASK)
>> TPM_CONF_CSOO_SHIFT)

```

```

#define TPM_BRD_CONF_CS0O(base) (BME_UBFX32(&TPM_CONF_REG(base),  

TPM_CONF_CS0O_SHIFT, TPM_CONF_CS0O_WIDTH))

/*! @brief Set the CSOO field to a new value. */  

#define TPM_WR_CONF_CS0O(base, value) (TPM_RMW_CONF(base,  

TPM_CONF_CS0O_MASK, TPM_CONF_CS0O(value)))  

#define TPM_BWR_CONF_CS0O(base, value) (BME_BFI32(&TPM_CONF_REG(base),  

((uint32_t)(value) << TPM_CONF_CS0O_SHIFT), TPM_CONF_CS0O_SHIFT,  

TPM_CONF_CS0O_WIDTH))  

/*@}*/

/*!  

 * @name Register TPM_CONF, field CROT[18] (RW)  

 *  

 * When set, the LPTPM counter will reload with zero (and initialize PWM  

outputs  

 * to their default value) when a rising edge is detected on the selected  

 * trigger input. The trigger input is ignored if the LPTPM counter is  

paused during  

 * debug mode or doze mode. This field should only be changed when the  

LPTPM  

 * counter is disabled.  

 *  

 * Values:  

 * - 0b0 - Counter is not reloaded due to a rising edge on the selected  

input  

 * trigger  

 * - 0b1 - Counter is reloaded when a rising edge is detected on the  

selected  

 * input trigger  

 */  

/*@*/  

/*! @brief Read current value of the TPM_CONF_CROT field. */  

#define TPM_RD_CONF_CROT(base) ((TPM_CONF_REG(base) & TPM_CONF_CROT_MASK)  

>> TPM_CONF_CROT_SHIFT)  

#define TPM_BRD_CONF_CROT(base) (BME_UBFX32(&TPM_CONF_REG(base),  

TPM_CONF_CROT_SHIFT, TPM_CONF_CROT_WIDTH))

/*! @brief Set the CROT field to a new value. */  

#define TPM_WR_CONF_CROT(base, value) (TPM_RMW_CONF(base,  

TPM_CONF_CROT_MASK, TPM_CONF_CROT(value)))  

#define TPM_BWR_CONF_CROT(base, value) (BME_BFI32(&TPM_CONF_REG(base),  

((uint32_t)(value) << TPM_CONF_CROT_SHIFT), TPM_CONF_CROT_SHIFT,  

TPM_CONF_CROT_WIDTH))  

/*@*/

/*!  

 * @name Register TPM_CONF, field TRGSEL[27:24] (RW)  

 *  

 * Selects the input trigger to use for starting the counter and/or  

reloading  

 * the counter. This field should only be changed when the LPTPM counter  

is  

 * disabled. See Chip configuration section for available options.

```

```

        */
/*@{*/
/*! @brief Read current value of the TPM_CONF_TRGSEL field. */
#define TPM_RD_CONF_TRGSEL(base) ((TPM_CONF_REG(base) &
TPM_CONF_TRGSEL_MASK) >> TPM_CONF_TRGSEL_SHIFT)
#define TPM_BRD_CONF_TRGSEL(base) (BME_UBFX32(&TPM_CONF_REG(base),
TPM_CONF_TRGSEL_SHIFT, TPM_CONF_TRGSEL_WIDTH))

/*! @brief Set the TRGSEL field to a new value. */
#define TPM_WR_CONF_TRGSEL(base, value) (TPM_RMW_CONF(base,
TPM_CONF_TRGSEL_MASK, TPM_CONF_TRGSEL(value)))
#define TPM_BWR_CONF_TRGSEL(base, value) (BME_BFI32(&TPM_CONF_REG(base),
((uint32_t)(value) << TPM_CONF_TRGSEL_SHIFT), TPM_CONF_TRGSEL_SHIFT,
TPM_CONF_TRGSEL_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 TSI
 *
 * Touch sense input
 *
 * Registers defined in this header file:
 * - TSI_GENCS - TSI General Control and Status Register
 * - TSI_DATA - TSI DATA Register
 * - TSI_TSHD - TSI Threshold Register
 */

#define TSI_INSTANCE_COUNT (1U) /*!< Number of instances of the TSI
module. */
#define TSI0_IDX (0U) /*!< Instance number for TSI0. */

*****  

*****  

    * TSI_GENCS - TSI General Control and Status Register  

*****  

*****  

    *!  

    * @brief TSI_GENCS - TSI General Control and Status Register (RW)  

    *  

    * Reset value: 0x00000000U  

    *  

    * This control register provides various control and configuration
information  

    * for the TSI module. When TSI is working, the configuration bits
(GENCS[TSIEN],  

    * GENCS[TSIIEN], and GENCS[STM]) must not be changed. The EOSF flag is
kept  

    * until the software acknowledge it.  

    */  

/*!  

    * @name Constants and macros for entire TSI_GENCS register  

    */

```

```

/*@{*/
#define TSI_RD_GENCS(base)          (TSI_GENCS_REG(base))
#define TSI_WR_GENCS(base, value)   (TSI_GENCS_REG(base) = (value))
#define TSI_RMW_GENCS(base, mask, value) (TSI_WR_GENCS(base,
(TSI_RD_GENCS(base) & ~mask)) | (value)))
#define TSI_SET_GENCS(base, value)  (BME_OR32(&TSI_GENCS_REG(base),
(uint32_t)(value)))
#define TSI_CLR_GENCS(base, value)  (BME_AND32(&TSI_GENCS_REG(base),
(uint32_t)(~(value))))
#define TSI_TOG_GENCS(base, value)  (BME_XOR32(&TSI_GENCS_REG(base),
(uint32_t)(value)))
/*@}*/

/*
 * Constants & macros for individual TSI_GENCS bitfields
 */

/*!
 * @name Register TSI_GENCS, field CURSW[1] (RW)
 *
 * This bit specifies if the current sources of electrode oscillator and
 * reference oscillator are swapped.
 *
 * Values:
 * - 0b0 - The current source pair are not swapped.
 * - 0b1 - The current source pair are swapped.
 */
/*@*/
/*! @brief Read current value of the TSI_GENCS_CURSW field. */
#define TSI_RD_GENCS_CURSW(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_CURSW_MASK) >> TSI_GENCS_CURSW_SHIFT)
#define TSI_BRD_GENCS_CURSW(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_CURSW_SHIFT, TSI_GENCS_CURSW_WIDTH))

/*! @brief Set the CURSW field to a new value. */
#define TSI_WR_GENCS_CURSW(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_CURSW_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_CURSW(value)))
#define TSI_BWR_GENCS_CURSW(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_CURSW_SHIFT), TSI_GENCS_CURSW_SHIFT,
TSI_GENCS_CURSW_WIDTH))
/*@*/

/*!
 * @name Register TSI_GENCS, field EOSF[2] (W1C)
 *
 * This flag is set when all active electrodes are finished scanning
 * after a
 * scan trigger. Write "1" , when this flag is set, to clear it.
 *
 * Values:
 * - 0b0 - Scan not complete.
 * - 0b1 - Scan complete.
 */

```

```

/*@{*/
/*! @brief Read current value of the TSI_GENCS_EOSF field. */
#define TSI_RD_GENCS_EOSF(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_EOSF_MASK) >> TSI_GENCS_EOSF_SHIFT)
#define TSI_BRD_GENCS_EOSF(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_EOSF_SHIFT, TSI_GENCS_EOSF_WIDTH))

/*! @brief Set the EOSF field to a new value. */
#define TSI_WR_GENCS_EOSF(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK), TSI_GENCS_EOSF(value)))
#define TSI_BWR_GENCS_EOSF(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_EOSF_SHIFT), TSI_GENCS_EOSF_SHIFT,
TSI_GENCS_EOSF_WIDTH))
/*@}*/

/*
 * @name Register TSI_GENCS, field SCNIP[3] (RO)
 *
 * This read-only bit indicates if scan is in progress. This bit will get
 * asserted after the analog bias circuit is stable after a trigger and
it changes
 * automatically by the TSI.
 *
 * Values:
 * - 0b0 - No scan in progress.
 * - 0b1 - Scan in progress.
 */
/*@*/
/*! @brief Read current value of the TSI_GENCS_SCNIP field. */
#define TSI_RD_GENCS_SCNIP(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_SCNIP_MASK) >> TSI_GENCS_SCNIP_SHIFT)
#define TSI_BRD_GENCS_SCNIP(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_SCNIP_SHIFT, TSI_GENCS_SCNIP_WIDTH))
/*@*/

/*
 * @name Register TSI_GENCS, field STM[4] (RW)
 *
 * This bit specifies the trigger mode. User is allowed to change this
bit when
 * TSI is not working in progress.
 *
 * Values:
 * - 0b0 - Software trigger scan.
 * - 0b1 - Hardware trigger scan.
 */
/*@*/
/*! @brief Read current value of the TSI_GENCS_STM field. */
#define TSI_RD_GENCS_STM(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_STM_MASK) >> TSI_GENCS_STM_SHIFT)
#define TSI_BRD_GENCS_STM(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_STM_SHIFT, TSI_GENCS_STM_WIDTH))

/*! @brief Set the STM field to a new value. */

```

```

#define TSI_WR_GENCS_STM(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_STM_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_STM(value)))
#define TSI_BWR_GENCS_STM(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_STM_SHIFT), TSI_GENCS_STM_SHIFT,
TSI_GENCS_STM_WIDTH))
/*@}*/

/*
 * @name Register TSI_GENCS, field STPE[5] (RW)
 *
 * This bit enables TSI module function in low power modes (stop, VLPS,
LLS and
 * VLLS{3,2,1}).
 *
 * Values:
 * - 0b0 - TSI is disabled when MCU goes into low power mode.
 * - 0b1 - Allows TSI to continue running in all low power modes.
 */
/*@{*/
/*! @brief Read current value of the TSI_GENCS_STPE field. */
#define TSI_RD_GENCS_STPE(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_STPE_MASK) >> TSI_GENCS_STPE_SHIFT)
#define TSI_BRD_GENCS_STPE(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_STPE_SHIFT, TSI_GENCS_STPE_WIDTH))

/*! @brief Set the STPE field to a new value. */
#define TSI_WR_GENCS_STPE(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_STPE_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_STPE(value)))
#define TSI_BWR_GENCS_STPE(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_STPE_SHIFT), TSI_GENCS_STPE_SHIFT,
TSI_GENCS_STPE_WIDTH))
/*@}*/

/*
 * @name Register TSI_GENCS, field TSIIEN[6] (RW)
 *
 * This bit enables TSI module interrupt request to CPU when the scan
completes.
 * The interrupt will wake MCU from low power mode if this interrupt is
enabled.
 *
 * Values:
 * - 0b0 - TSI interrupt is disabled.
 * - 0b1 - TSI interrupt is enabled.
 */
/*@{*/
/*! @brief Read current value of the TSI_GENCS_TSIIEN field. */
#define TSI_RD_GENCS_TSIIEN(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_TSIIEN_MASK) >> TSI_GENCS_TSIIEN_SHIFT)
#define TSI_BRD_GENCS_TSIIEN(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_TSIIEN_SHIFT, TSI_GENCS_TSIIEN_WIDTH))

```

```

/*! @brief Set the TSIIEN field to a new value. */
#define TSI_WR_GENCS_TSIIEN(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_TSIIEN_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_TSIIEN(value)))
#define TSI_BWR_GENCS_TSIIEN(base, value)
(BME_BFI32(&TSI_GENCS_REG(base), ((uint32_t)(value) <<
TSI_GENCS_TSIIEN_SHIFT), TSI_GENCS_TSIIEN_SHIFT, TSI_GENCS_TSIIEN_WIDTH))
/*@}*/

/*!!
 * @name Register TSI_GENCS, field TSIEN[7] (RW)
 *
 * This bit enables TSI module.
 *
 * Values:
 * - 0b0 - TSI module disabled.
 * - 0b1 - TSI module enabled.
 */
/*@{*/
/*! @brief Read current value of the TSI_GENCS_TSIEN field. */
#define TSI_RD_GENCS_TSIEN(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_TSIEN_MASK) >> TSI_GENCS_TSIEN_SHIFT)
#define TSI_BRD_GENCS_TSIEN(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_TSIEN_SHIFT, TSI_GENCS_TSIEN_WIDTH))

/*! @brief Set the TSIEN field to a new value. */
#define TSI_WR_GENCS_TSIEN(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_TSIEN_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_TSIEN(value)))
#define TSI_BWR_GENCS_TSIEN(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_TSIEN_SHIFT), TSI_GENCS_TSIEN_SHIFT,
TSI_GENCS_TSIEN_WIDTH))
/*@}*/

/*!!
 * @name Register TSI_GENCS, field NSCN[12:8] (RW)
 *
 * These bits indicate the scan number for each electrode. The scan
number is
 * equal to NSCN + 1, which allows the scan time ranges from 1 to 32. By
default,
 * NSCN is configured as 0, which asserts the TSI scans once on the
selected
 * electrode channel.
 *
 * Values:
 * - 0b00000 - Once per electrode
 * - 0b00001 - Twice per electrode
 * - 0b00010 - 3 times per electrode
 * - 0b00011 - 4 times per electrode
 * - 0b00100 - 5 times per electrode
 * - 0b00101 - 6 times per electrode
 * - 0b00110 - 7 times per electrode
 * - 0b00111 - 8 times per electrode

```

```

* - 0b01000 - 9 times per electrode
* - 0b01001 - 10 times per electrode
* - 0b01010 - 11 times per electrode
* - 0b01011 - 12 times per electrode
* - 0b01100 - 13 times per electrode
* - 0b01101 - 14 times per electrode
* - 0b01110 - 15 times per electrode
* - 0b01111 - 16 times per electrode
* - 0b10000 - 17 times per electrode
* - 0b10001 - 18 times per electrode
* - 0b10010 - 19 times per electrode
* - 0b10011 - 20 times per electrode
* - 0b10100 - 21 times per electrode
* - 0b10101 - 22 times per electrode
* - 0b10110 - 23 times per electrode
* - 0b10111 - 24 times per electrode
* - 0b11000 - 25 times per electrode
* - 0b11001 - 26 times per electrode
* - 0b11010 - 27 times per electrode
* - 0b11011 - 28 times per electrode
* - 0b11100 - 29 times per electrode
* - 0b11101 - 30 times per electrode
* - 0b11110 - 31 times per electrode
* - 0b11111 - 32 times per electrode
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_NSCN field. */
#define TSI_RD_GENCS_NSCN(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_NSCN_MASK) >> TSI_GENCS_NSCN_SHIFT)
#define TSI_BRD_GENCS_NSCN(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_NSCN_SHIFT, TSI_GENCS_NSCN_WIDTH))

/*! @brief Set the NSCN field to a new value. */
#define TSI_WR_GENCS_NSCN(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_NSCN_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_NSCN(value)))
#define TSI_BWR_GENCS_NSCN(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_NSCN_SHIFT), TSI_GENCS_NSCN_SHIFT,
TSI_GENCS_NSCN_WIDTH))
/*}@*/ */

/*! 
 * @name Register TSI_GENCS, field PS[15:13] (RW)
 *
 * These bits indicate the prescaler of the output of electrode
oscillator.
*
* Values:
* - 0b000 - Electrode Oscillator Frequency divided by 1
* - 0b001 - Electrode Oscillator Frequency divided by 2
* - 0b010 - Electrode Oscillator Frequency divided by 4
* - 0b011 - Electrode Oscillator Frequency divided by 8
* - 0b100 - Electrode Oscillator Frequency divided by 16
* - 0b101 - Electrode Oscillator Frequency divided by 32

```

```

* - 0b110 - Electrode Oscillator Frequency divided by 64
* - 0b111 - Electrode Oscillator Frequency divided by 128
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_PS field. */
#define TSI_RD_GENCS_PS(base) ((TSI_GENCS_REG(base) & TSI_GENCS_PS_MASK)
>> TSI_GENCS_PS_SHIFT)
#define TSI_BRD_GENCS_PS(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_PS_SHIFT, TSI_GENCS_PS_WIDTH))

/*! @brief Set the PS field to a new value. */
#define TSI_WR_GENCS_PS(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_PS_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_PS(value)))
#define TSI_BWR_GENCS_PS(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_PS_SHIFT), TSI_GENCS_PS_SHIFT,
TSI_GENCS_PS_WIDTH))
/*@}*/

/*!
* @name Register TSI_GENCS, field EXTCHRG[18:16] (RW)
*
* These bits indicate the electrode oscillator charge and discharge
current
* value.
*
* Values:
* - 0b000 - 500 nA.
* - 0b001 - 1 uA.
* - 0b010 - 2 uA.
* - 0b011 - 4 uA.
* - 0b100 - 8 uA.
* - 0b101 - 16 uA.
* - 0b110 - 32 uA.
* - 0b111 - 64 uA.
*/
/*@*/
/*! @brief Read current value of the TSI_GENCS_EXTCHRG field. */
#define TSI_RD_GENCS_EXTCHRG(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_EXTCHRG_MASK) >> TSI_GENCS_EXTCHRG_SHIFT)
#define TSI_BRD_GENCS_EXTCHRG(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_EXTCHRG_SHIFT, TSI_GENCS_EXTCHRG_WIDTH))

/*! @brief Set the EXTCHRG field to a new value. */
#define TSI_WR_GENCS_EXTCHRG(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_EXTCHRG_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_EXTCHRG(value)))
#define TSI_BWR_GENCS_EXTCHRG(base, value)
(BME_BFI32(&TSI_GENCS_REG(base), ((uint32_t)(value) <<
TSI_GENCS_EXTCHRG_SHIFT), TSI_GENCS_EXTCHRG_SHIFT,
TSI_GENCS_EXTCHRG_WIDTH))
/*@}*/

/*

```

```

* @name Register TSI_GENCS, field DVOLT[20:19] (RW)
*
* These bits indicate the oscillator's voltage rails as below.
*
* Values:
* - 0b00 - DV = 1.03 V; VP = 1.33 V; Vm = 0.30 V.
* - 0b01 - DV = 0.73 V; VP = 1.18 V; Vm = 0.45 V.
* - 0b10 - DV = 0.43 V; VP = 1.03 V; Vm = 0.60 V.
* - 0b11 - DV = 0.29 V; VP = 0.95 V; Vm = 0.67 V.
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_DVOLT field. */
#define TSI_RD_GENCS_DVOLT(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_DVOLT_MASK) >> TSI_GENCS_DVOLT_SHIFT)
#define TSI_BRD_GENCS_DVOLT(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_DVOLT_SHIFT, TSI_GENCS_DVOLT_WIDTH))

/*! @brief Set the DVOLT field to a new value. */
#define TSI_WR_GENCS_DVOLT(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_DVOLT_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_DVOLT(value)))
#define TSI_BWR_GENCS_DVOLT(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_DVOLT_SHIFT), TSI_GENCS_DVOLT_SHIFT,
TSI_GENCS_DVOLT_WIDTH))
/*@}*/

/*
* @name Register TSI_GENCS, field REFCHRG[23:21] (RW)
*
* These bits indicate the reference oscillator charge and discharge
current
* value.
*
* Values:
* - 0b000 - 500 nA.
* - 0b001 - 1 uA.
* - 0b010 - 2 uA.
* - 0b011 - 4 uA.
* - 0b100 - 8 uA.
* - 0b101 - 16 uA.
* - 0b110 - 32 uA.
* - 0b111 - 64 uA.
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_REFCHRG field. */
#define TSI_RD_GENCS_REFCHRG(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_REFCHRG_MASK) >> TSI_GENCS_REFCHRG_SHIFT)
#define TSI_BRD_GENCS_REFCHRG(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_REFCHRG_SHIFT, TSI_GENCS_REFCHRG_WIDTH))

/*! @brief Set the REFCHRG field to a new value. */
#define TSI_WR_GENCS_REFCHRG(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_REFCHRG_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_REFCHRG(value)))

```

```

#define TSI_BWR_GENCS_REFCHRG(base, value)
(BME_BFI32(&TSI_GENCS_REG(base), ((uint32_t)(value) <<
TSI_GENCS_REFCHRG_SHIFT), TSI_GENCS_REFCHRG_SHIFT,
TSI_GENCS_REFCHRG_WIDTH))
/*@}*/

/*
 * @name Register TSI_GENCS, field MODE[27:24] (RW)
 *
 * Set up TSI analog modes, especially, setting MODE[3:2] to not 2'b00
will
 * configure TSI to noise detection modes. MODE[1:0] take no effect on
TSI operation
 * mode and should always write to 2'b00 for setting up. When reading
this field
 * will return the analog status. Refer to chapter "Noise detection mode"
for
 * details.
 *
 * Values:
 * - 0b0000 - Set TSI in capacitive sensing(non-noise detection) mode.
 * - 0b0100 - Set TSI analog to work in single threshold noise detection
mode
 *      and the frequency limitation circuit is disabled.
 * - 0b1000 - Set TSI analog to work in single threshold noise detection
mode
 *      and the frequency limitation circuit is enabled to work in higher
 *      frequencies operations.
 * - 0b1100 - Set TSI analog to work in automatic noise detection mode.
 */
/*@{*/
/*! @brief Read current value of the TSI_GENCS_MODE field. */
#define TSI_RD_GENCS_MODE(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_MODE_MASK) >> TSI_GENCS_MODE_SHIFT)
#define TSI_BRD_GENCS_MODE(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_MODE_SHIFT, TSI_GENCS_MODE_WIDTH))

/*! @brief Set the MODE field to a new value. */
#define TSI_WR_GENCS_MODE(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_MODE_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_MODE(value)))
#define TSI_BWR_GENCS_MODE(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_MODE_SHIFT), TSI_GENCS_MODE_SHIFT,
TSI_GENCS_MODE_WIDTH))
/*@}*/

/*
 * @name Register TSI_GENCS, field ESOR[28] (RW)
 *
 * This bit is used to select out-of-range or end-of-scan event to
generate an
 * interrupt.
 *
 * Values:

```

```

* - 0b0 - Out-of-range interrupt is allowed.
* - 0b1 - End-of-scan interrupt is allowed.
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_ESOR field. */
#define TSI_RD_GENCS_ESOR(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_ESOR_MASK) >> TSI_GENCS_ESOR_SHIFT)
#define TSI_BRD_GENCS_ESOR(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_ESOR_SHIFT, TSI_GENCS_ESOR_WIDTH))

/*! @brief Set the ESOR field to a new value. */
#define TSI_WR_GENCS_ESOR(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_ESOR_MASK | TSI_GENCS_EOSF_MASK | TSI_GENCS_OUTRGF_MASK),
TSI_GENCS_ESOR(value)))
#define TSI_BWR_GENCS_ESOR(base, value) (BME_BFI32(&TSI_GENCS_REG(base),
((uint32_t)(value) << TSI_GENCS_ESOR_SHIFT), TSI_GENCS_ESOR_SHIFT,
TSI_GENCS_ESOR_WIDTH))
/*}@*/ */

/*!
* @name Register TSI_GENCS, field OUTRGF[31] (W1C)
*
* This flag is set if the result register of the enabled electrode is
out of
* the range defined by the TSI_THRESHOLD register. This flag is set only
when TSI
* is configured in non-noise detection mode. It can be read once the CPU
wakes.
* Write "1" , when this flag is set, to clear it.
*/
/*@{*/
/*! @brief Read current value of the TSI_GENCS_OUTRGF field. */
#define TSI_RD_GENCS_OUTRGF(base) ((TSI_GENCS_REG(base) &
TSI_GENCS_OUTRGF_MASK) >> TSI_GENCS_OUTRGF_SHIFT)
#define TSI_BRD_GENCS_OUTRGF(base) (BME_UBFX32(&TSI_GENCS_REG(base),
TSI_GENCS_OUTRGF_SHIFT, TSI_GENCS_OUTRGF_WIDTH))

/*! @brief Set the OUTRGF field to a new value. */
#define TSI_WR_GENCS_OUTRGF(base, value) (TSI_RMW_GENCS(base,
(TSI_GENCS_OUTRGF_MASK | TSI_GENCS_EOSF_MASK), TSI_GENCS_OUTRGF(value)))
#define TSI_BWR_GENCS_OUTRGF(base, value)
(BME_BFI32(&TSI_GENCS_REG(base), ((uint32_t)(value) <<
TSI_GENCS_OUTRGF_SHIFT), TSI_GENCS_OUTRGF_SHIFT, TSI_GENCS_OUTRGF_WIDTH))
/*}@*/ */

/********************* ****
* TSI_DATA - TSI DATA Register
*****/
/*!
* @brief TSI_DATA - TSI DATA Register (RW)

```

```

/*
 * Reset value: 0x0000000U
 */
/*!
 * @name Constants and macros for entire TSI_DATA register
 */
/*@{*/
#define TSI_RD_DATA(base)          (TSI_DATA_REG(base))
#define TSI_WR_DATA(base, value)   (TSI_DATA_REG(base) = (value))
#define TSI_RMW_DATA(base, mask, value) (TSI_WR_DATA(base,
(TSI_RD_DATA(base) & ~mask) | (value)))
#define TSI_SET_DATA(base, value)  (BME_OR32(&TSI_DATA_REG(base),
(uint32_t)(value)))
#define TSI_CLR_DATA(base, value)  (BME_AND32(&TSI_DATA_REG(base),
(uint32_t)(~(value))))
#define TSI_TOG_DATA(base, value)  (BME_XOR32(&TSI_DATA_REG(base),
(uint32_t)(value)))
/*}@*/ */

/*
 * Constants & macros for individual TSI_DATA bitfields
 */
/*!
 * @name Register TSI_DATA, field TSICNT[15:0] (RO)
 *
 * These read-only bits record the accumulated scan counter value ticked
by the
 * reference oscillator.
 */
/*@{*/
/*! @brief Read current value of the TSI_DATA_TSICNT field. */
#define TSI_RD_DATA_TSICNT(base) ((TSI_DATA_REG(base) &
TSI_DATA_TSICNT_MASK) >> TSI_DATA_TSICNT_SHIFT)
#define TSI_BRD_DATA_TSICNT(base) (BME_UBFX32(&TSI_DATA_REG(base),
TSI_DATA_TSICNT_SHIFT, TSI_DATA_TSICNT_WIDTH))
/*}@*/ */

/*!
 * @name Register TSI_DATA, field SWTS[22] (WORZ)
 *
 * This write-only bit is a software start trigger. When STM bit is
clear, write
 * "1" to this bit will start a scan. The electrode channel to be scanned
is
 * determinated by TSI_DATA[TSICH] bits.
 *
 * Values:
 * - 0b0 - No effect.
 * - 0b1 - Start a scan to determine which channel is specified by
 *         TSI_DATA[TSICH].
 */
/*@{*/
/*! @brief Set the SWTS field to a new value. */

```

```

#define TSI_WR_DATA_SWTS(base, value) (TSI_RMW_DATA(base,
TSI_DATA_SWTS_MASK, TSI_DATA_SWTS(value)))
#define TSI_BWR_DATA_SWTS(base, value) (BME_BFI32(&TSI_DATA_REG(base),
((uint32_t)(value) << TSI_DATA_SWTS_SHIFT), TSI_DATA_SWTS_SHIFT,
TSI_DATA_SWTS_WIDTH))
/*@}*/

/*
 * @name Register TSI_DATA, field DMAEN[23] (RW)
 *
 * This bit is used together with the TSI interrupt enable bits(TSIIIE,
ESOR) to
 * generate a DMA transfer request instead of an interrupt.
 *
 * Values:
 * - 0b0 - Interrupt is selected when the interrupt enable bit is set and
the
 *         corresponding TSI events assert.
 * - 0b1 - DMA transfer request is selected when the interrupt enable bit
is set
 *         and the corresponding TSI events assert.
 */
/*@{*/
/*! @brief Read current value of the TSI_DATA_DMAEN field. */
#define TSI_RD_DATA_DMAEN(base) ((TSI_DATA_REG(base) &
TSI_DATA_DMAEN_MASK) >> TSI_DATA_DMAEN_SHIFT)
#define TSI_BRD_DATA_DMAEN(base) (BME_UBFX32(&TSI_DATA_REG(base),
TSI_DATA_DMAEN_SHIFT, TSI_DATA_DMAEN_WIDTH))

/*! @brief Set the DMAEN field to a new value. */
#define TSI_WR_DATA_DMAEN(base, value) (TSI_RMW_DATA(base,
TSI_DATA_DMAEN_MASK, TSI_DATA_DMAEN(value)))
#define TSI_BWR_DATA_DMAEN(base, value) (BME_BFI32(&TSI_DATA_REG(base),
((uint32_t)(value) << TSI_DATA_DMAEN_SHIFT), TSI_DATA_DMAEN_SHIFT,
TSI_DATA_DMAEN_WIDTH))
/*@}*/

/*
 * @name Register TSI_DATA, field TSICH[31:28] (RW)
 *
 * These bits specify current channel to be measured. In hardware trigger
mode
 * (TSI_GENCS[STM] = 1), the scan will not start until the hardware
trigger
 * occurs. In software trigger mode (TSI_GENCS[STM] = 0), the scan starts
immediately
 * when TSI_DATA[SWTS] bit is written by 1.
 *
 * Values:
 * - 0b0000 - Channel 0.
 * - 0b0001 - Channel 1.
 * - 0b0010 - Channel 2.
 * - 0b0011 - Channel 3.
 * - 0b0100 - Channel 4.

```

```

* - 0b0101 - Channel 5.
* - 0b0110 - Channel 6.
* - 0b0111 - Channel 7.
* - 0b1000 - Channel 8.
* - 0b1001 - Channel 9.
* - 0b1010 - Channel 10.
* - 0b1011 - Channel 11.
* - 0b1100 - Channel 12.
* - 0b1101 - Channel 13.
* - 0b1110 - Channel 14.
* - 0b1111 - Channel 15.
*/
/*@{*/
/*! @brief Read current value of the TSI_DATA_TSICH field. */
#define TSI_RD_DATA_TSICH(base) ((TSI_DATA_REG(base) &
TSI_DATA_TSICH_MASK) >> TSI_DATA_TSICH_SHIFT)
#define TSI_BRD_DATA_TSICH(base) (BME_UBFX32(&TSI_DATA_REG(base),
TSI_DATA_TSICH_SHIFT, TSI_DATA_TSICH_WIDTH))

/*! @brief Set the TSICH field to a new value. */
#define TSI_WR_DATA_TSICH(base, value) (TSI_RMW_DATA(base,
TSI_DATA_TSICH_MASK, TSI_DATA_TSICH(value)))
#define TSI_BWR_DATA_TSICH(base, value) (BME_BFI32(&TSI_DATA_REG(base),
(uint32_t)(value) << TSI_DATA_TSICH_SHIFT), TSI_DATA_TSICH_SHIFT,
TSI_DATA_TSICH_WIDTH)
/*}@*/}

/********************* TSI Threshold Register ********************
*****
 * TSI_TSHD - TSI Threshold Register
*****
/********************* TSI Threshold Register ********************
****/


/*!
 * @brief TSI_TSHD - TSI Threshold Register (RW)
 *
 * Reset value: 0x00000000U
 */
/*!
 * @name Constants and macros for entire TSI_TSHD register
 */
/*@{*/
#define TSI_RD_TSHD(base) (TSI_TSHD_REG(base))
#define TSI_WR_TSHD(base, value) (TSI_TSHD_REG(base) = (value))
#define TSI_RMW_TSHD(base, mask, value) (TSI_WR_TSHD(base,
(TSI_RD_TSHD(base) & ~mask) | (value)))
#define TSI_SET_TSHD(base, value) (BME_OR32(&TSI_TSHD_REG(base),
(uint32_t)(value)))
#define TSI_CLR_TSHD(base, value) (BME_AND32(&TSI_TSHD_REG(base),
(uint32_t)(~(value))))
#define TSI_TOG_TSHD(base, value) (BME_XOR32(&TSI_TSHD_REG(base),
(uint32_t)(value)))
/*}@*/

```

```

/*
 * Constants & macros for individual TSI_TSHD bitfields
 */

/*!
 * @name Register TSI_TSHD, field THRESL[15:0] (RW)
 *
 * This half-word specifies the low threshold of the wakeup channel.
 */
/*@{*/
/*! @brief Read current value of the TSI_TSHD_THRESL field. */
#define TSI_RD_TSHD_THRESL(base) ((TSI_TSHD_REG(base) &
TSI_TSHD_THRESL_MASK) >> TSI_TSHD_THRESL_SHIFT)
#define TSI_BRD_TSHD_THRESL(base) (BME_UBFX32(&TSI_TSHD_REG(base),
TSI_TSHD_THRESL_SHIFT, TSI_TSHD_THRESL_WIDTH))

/*! @brief Set the THRESL field to a new value. */
#define TSI_WR_TSHD_THRESL(base, value) (TSI_RMW_TSHD(base,
TSI_TSHD_THRESL_MASK, TSI_TSHD_THRESL(value)))
#define TSI_BWR_TSHD_THRESL(base, value) (BME_BFI32(&TSI_TSHD_REG(base),
((uint32_t)(value) << TSI_TSHD_THRESL_SHIFT), TSI_TSHD_THRESL_SHIFT,
TSI_TSHD_THRESL_WIDTH))
/*}@*/ */

/*!
 * @name Register TSI_TSHD, field THRESH[31:16] (RW)
 *
 * This half-word specifies the high threshold of the wakeup channel.
 */
/*@{*/
/*! @brief Read current value of the TSI_TSHD_THRESH field. */
#define TSI_RD_TSHD_THRESH(base) ((TSI_TSHD_REG(base) &
TSI_TSHD_THRESH_MASK) >> TSI_TSHD_THRESH_SHIFT)
#define TSI_BRD_TSHD_THRESH(base) (BME_UBFX32(&TSI_TSHD_REG(base),
TSI_TSHD_THRESH_SHIFT, TSI_TSHD_THRESH_WIDTH))

/*! @brief Set the THRESH field to a new value. */
#define TSI_WR_TSHD_THRESH(base, value) (TSI_RMW_TSHD(base,
TSI_TSHD_THRESH_MASK, TSI_TSHD_THRESH(value)))
#define TSI_BWR_TSHD_THRESH(base, value) (BME_BFI32(&TSI_TSHD_REG(base),
((uint32_t)(value) << TSI_TSHD_THRESH_SHIFT), TSI_TSHD_THRESH_SHIFT,
TSI_TSHD_THRESH_WIDTH))
/*}@*/ */

/*
 * MKL25Z4 UART
 *
 * Universal Asynchronous Receiver/Transmitter (UART)
 *
 * Registers defined in this header file:
 * - UART_BDH - UART Baud Rate Register: High
 * - UART_BDL - UART Baud Rate Register: Low
 * - UART_C1 - UART Control Register 1

```

```

* - UART_C2 - UART Control Register 2
* - UART_S1 - UART Status Register 1
* - UART_S2 - UART Status Register 2
* - UART_C3 - UART Control Register 3
* - UART_D - UART Data Register
* - UART_C4 - UART Control Register 4
*/
#define UART_INSTANCE_COUNT (3U) /*!< Number of instances of the UART
module. */
#define UART1_IDX (1U) /*!< Instance number for UART1. */
#define UART2_IDX (2U) /*!< Instance number for UART2. */

/***** ****
* UART_BDH - UART Baud Rate Register: High
*****
*/
/*!
* @brief UART_BDH - UART Baud Rate Register: High (RW)
*
* Reset value: 0x00U
*
* This register, along with UART_BDL, controls the prescale divisor for
UART
* baud rate generation. To update the 13-bit baud rate setting
[SBR12:SBR0], first
* write to UART_BDH to buffer the high half of the new value and then
write to
* UART_BDL. The working value in UART_BDH does not change until UART_BDL
is
* written.
*/
/*!
* @name Constants and macros for entire UART_BDH register
*/
/*@{ */
#define UART_RD_BDH(base) (UART_BDH_REG(base))
#define UART_WR_BDH(base, value) (UART_BDH_REG(base) = (value))
#define UART_RMW_BDH(base, mask, value) (UART_WR_BDH(base,
(UART_RD_BDH(base) & ~mask) | (value)))
#define UART_SET_BDH(base, value) (BME_OR8(&UART_BDH_REG(base),
(uint8_t)(value)))
#define UART_CLR_BDH(base, value) (BME_AND8(&UART_BDH_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_BDH(base, value) (BME_XOR8(&UART_BDH_REG(base),
(uint8_t)(value)))
/*@} */

/*
* Constants & macros for individual UART_BDH bitfields
*/

```

```

/*!
 * @name Register UART_BDH, field SBR[4:0] (RW)
 *
 * The 13 bits in SBR[12:0] are referred to collectively as BR, and they
set the
 * modulo divide rate for the UART baud rate generator. When BR is
cleared, the
 * UART baud rate generator is disabled to reduce supply current. When BR
is 1 -
 * 8191, the UART baud rate equals BUSCLK/(16*BR) .
*/
/*@{*/
/*! @brief Read current value of the UART_BDH_SBR field. */
#define UART_RD_BDH_SBR(base) ((UART_BDH_REG(base) & UART_BDH_SBR_MASK)
>> UART_BDH_SBR_SHIFT)
#define UART_BRD_BDH_SBR(base) (BME_UBFX8(&UART_BDH_REG(base),
UART_BDH_SBR_SHIFT, UART_BDH_SBR_WIDTH))

/*! @brief Set the SBR field to a new value. */
#define UART_WR_BDH_SBR(base, value) (UART_RMW_BDH(base,
UART_BDH_SBR_MASK, UART_BDH_SBR(value)))
#define UART_BWR_BDH_SBR(base, value) (BME_BFI8(&UART_BDH_REG(base),
((uint8_t)(value) << UART_BDH_SBR_SHIFT), UART_BDH_SBR_SHIFT,
UART_BDH_SBR_WIDTH))
/*@}*/

/*!
 * @name Register UART_BDH, field SBNS[5] (RW)
 *
 * SBNS determines whether data characters are one or two stop bits.
 *
 * Values:
 * - 0b0 - One stop bit.
 * - 0b1 - Two stop bit.
*/
/*@{*/
/*! @brief Read current value of the UART_BDH_SBNS field. */
#define UART_RD_BDH_SBNS(base) ((UART_BDH_REG(base) & UART_BDH_SBNS_MASK)
>> UART_BDH_SBNS_SHIFT)
#define UART_BRD_BDH_SBNS(base) (BME_UBFX8(&UART_BDH_REG(base),
UART_BDH_SBNS_SHIFT, UART_BDH_SBNS_WIDTH))

/*! @brief Set the SBNS field to a new value. */
#define UART_WR_BDH_SBNS(base, value) (UART_RMW_BDH(base,
UART_BDH_SBNS_MASK, UART_BDH_SBNS(value)))
#define UART_BWR_BDH_SBNS(base, value) (BME_BFI8(&UART_BDH_REG(base),
((uint8_t)(value) << UART_BDH_SBNS_SHIFT), UART_BDH_SBNS_SHIFT,
UART_BDH_SBNS_WIDTH))
/*@}*/

/*!
 * @name Register UART_BDH, field RXEDGIE[6] (RW)
*

```

```

* Values:
* - 0b0 - Hardware interrupts from UART_S2[RXEDGIF] disabled (use
polling).
* - 0b1 - Hardware interrupt requested when UART_S2[RXEDGIF] flag is 1.
*/
/*@{*/
/*! @brief Read current value of the UART_BDH_RXEDGIE field. */
#define UART_RD_BDH_RXEDGIE(base) ((UART_BDH_REG(base) &
UART_BDH_RXEDGIE_MASK) >> UART_BDH_RXEDGIE_SHIFT)
#define UART_BRD_BDH_RXEDGIE(base) (BME_UBFX8(&UART_BDH_REG(base),
UART_BDH_RXEDGIE_SHIFT, UART_BDH_RXEDGIE_WIDTH))

/*! @brief Set the RXEDGIE field to a new value. */
#define UART_WR_BDH_RXEDGIE(base, value) (UART_RMW_BDH(base,
UART_BDH_RXEDGIE_MASK, UART_BDH_RXEDGIE(value)))
#define UART_BWR_BDH_RXEDGIE(base, value) (BME_BFI8(&UART_BDH_REG(base),
((uint8_t)(value) << UART_BDH_RXEDGIE_SHIFT), UART_BDH_RXEDGIE_SHIFT,
UART_BDH_RXEDGIE_WIDTH))
/*@}*/

/*!
* @name Register UART_BDH, field LBKDIE[7] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from UART_S2[LBKDIIF] disabled (use
polling).
* - 0b1 - Hardware interrupt requested when UART_S2[LBKDIIF] flag is 1.
*/
/*@{*/
/*! @brief Read current value of the UART_BDH_LBKDIIE field. */
#define UART_RD_BDH_LBKDIIE(base) ((UART_BDH_REG(base) &
UART_BDH_LBKDIIE_MASK) >> UART_BDH_LBKDIIE_SHIFT)
#define UART_BRD_BDH_LBKDIIE(base) (BME_UBFX8(&UART_BDH_REG(base),
UART_BDH_LBKDIIE_SHIFT, UART_BDH_LBKDIIE_WIDTH))

/*! @brief Set the LBKDIE field to a new value. */
#define UART_WR_BDH_LBKDIIE(base, value) (UART_RMW_BDH(base,
UART_BDH_LBKDIIE_MASK, UART_BDH_LBKDIIE(value)))
#define UART_BWR_BDH_LBKDIIE(base, value) (BME_BFI8(&UART_BDH_REG(base),
((uint8_t)(value) << UART_BDH_LBKDIIE_SHIFT), UART_BDH_LBKDIIE_SHIFT,
UART_BDH_LBKDIIE_WIDTH))
/*@}*/

/*****
*****
* UART_BDL - UART Baud Rate Register: Low
*****
****/

/*!
* @brief UART_BDL - UART Baud Rate Register: Low (RW)
*
* Reset value: 0x04U

```

```

/*
 * This register, along with UART_BDH, control the prescale divisor for
UART
 * baud rate generation. To update the 13-bit baud rate setting
[SBR12:SBR0], first
 * write to UART_BDH to buffer the high half of the new value and then
write to
 * UART_BDL. The working value in UART_BDH does not change until UART_BDL
is
 * written. UART_BDL is reset to a non-zero value, so after reset the
baud rate
 * generator remains disabled until the first time the receiver or
transmitter is
 * enabled; that is, UART_C2[RE] or UART_C2[TE] bits are written to 1.
 */
/*!
 * @name Constants and macros for entire UART_BDL register
 */
/*@{ */
#define UART_RD_BDL(base)          (UART_BDL_REG(base))
#define UART_WR_BDL(base, value)   (UART_BDL_REG(base) = (value))
#define UART_RMW_BDL(base, mask, value) (UART_WR_BDL(base,
(UART_RD_BDL(base) & ~mask)) | (value)))
#define UART_SET_BDL(base, value)  (BME_OR8(&UART_BDL_REG(base),
(uint8_t)(value)))
#define UART_CLR_BDL(base, value)  (BME_AND8(&UART_BDL_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_BDL(base, value)  (BME_XOR8(&UART_BDL_REG(base),
(uint8_t)(value)))
/*@} */

/***** UART_C1 - UART Control Register 1 *****/
/*@{ */

/* @brief UART_C1 - UART Control Register 1 (RW)
 *
 * Reset value: 0x00U
 *
 * This read/write register controls various optional features of the
UART
 * system.
 */
/*!
 * @name Constants and macros for entire UART_C1 register
 */
/*@{ */
#define UART_RD_C1(base)          (UART_C1_REG(base))
#define UART_WR_C1(base, value)   (UART_C1_REG(base) = (value))

```

```

#define UART_RMWC1(base, mask, value) (UART_WR_C1(base,
(UART_RD_C1(base) & ~(mask)) | (value)))
#define UART_SET_C1(base, value) (BME_OR8(&UART_C1_REG(base),
(uint8_t)(value)))
#define UART_CLR_C1(base, value) (BME_AND8(&UART_C1_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_C1(base, value) (BME_XOR8(&UART_C1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART_C1 bitfields
 */

/*!!
 * @name Register UART_C1, field PT[0] (RW)
 *
 * Provided parity is enabled (PE = 1), this bit selects even or odd
 * parity. Odd
 * parity means the total number of 1s in the data character, including
 * the
 * parity bit, is odd. Even parity means the total number of 1s in the
 * data
 * character, including the parity bit, is even.
 *
 * Values:
 * - 0b0 - Even parity.
 * - 0b1 - Odd parity.
 */
/*@{*/
/*! @brief Read current value of the UART_C1_PT field. */
#define UART_RD_C1_PT(base) ((UART_C1_REG(base) & UART_C1_PT_MASK) >>
UART_C1_PT_SHIFT)
#define UART_BRD_C1_PT(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_PT_SHIFT, UART_C1_PT_WIDTH))

/*!! @brief Set the PT field to a new value. */
#define UART_WR_C1_PT(base, value) (UART_RMWC1(base, UART_C1_PT_MASK,
UART_C1_PT(value)))
#define UART_BWR_C1_PT(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_PT_SHIFT), UART_C1_PT_SHIFT,
UART_C1_PT_WIDTH))
/*@}*/

/*
 * @name Register UART_C1, field PE[1] (RW)
 *
 * Enables hardware parity generation and checking. When parity is
 * enabled, the
 * most significant bit (msb) of the data character, eighth or ninth data
 * bit, is
 * treated as the parity bit.
 *
 * Values:

```

```

* - 0b0 - No hardware parity generation or checking.
* - 0b1 - Parity enabled.
*/
/*@{*/
/*! @brief Read current value of the UART_C1_PE field. */
#define UART_RD_C1_PE(base) ((UART_C1_REG(base) & UART_C1_PE_MASK) >>
UART_C1_PE_SHIFT)
#define UART_BRD_C1_PE(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_PE_SHIFT, UART_C1_PE_WIDTH))

/*! @brief Set the PE field to a new value. */
#define UART_WR_C1_PE(base, value) (UART_RMW_C1(base, UART_C1_PE_MASK,
UART_C1_PE(value)))
#define UART_BWR_C1_PE(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_PE_SHIFT), UART_C1_PE_SHIFT,
UART_C1_PE_WIDTH))
/*}@*/
```

```

/*!!
* @name Register UART_C1, field ILT[2] (RW)
*
* Setting this bit to 1 ensures that the stop bits and logic 1 bits at
the end
* of a character do not count toward the 10 or 11 bit times of logic
high level
* needed by the idle line detection logic.
*
* Values:
* - 0b0 - Idle character bit count starts after start bit.
* - 0b1 - Idle character bit count starts after stop bit.
*/
/*@{*/
/*! @brief Read current value of the UART_C1_ILT field. */
#define UART_RD_C1_ILT(base) ((UART_C1_REG(base) & UART_C1_ILT_MASK) >>
UART_C1_ILT_SHIFT)
#define UART_BRD_C1_ILT(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_ILT_SHIFT, UART_C1_ILT_WIDTH))

/*! @brief Set the ILT field to a new value. */
#define UART_WR_C1_ILT(base, value) (UART_RMW_C1(base, UART_C1_ILT_MASK,
UART_C1_ILT(value)))
#define UART_BWR_C1_ILT(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_ILT_SHIFT), UART_C1_ILT_SHIFT,
UART_C1_ILT_WIDTH))
/*}@*/
```

```

/*!!
* @name Register UART_C1, field WAKE[3] (RW)
*
* Values:
* - 0b0 - Idle-line wakeup.
* - 0b1 - Address-mark wakeup.
*/
/*@{*/

```

```

/*! @brief Read current value of the UART_C1_WAKE field. */
#define UART_RD_C1_WAKE(base) ((UART_C1_REG(base) & UART_C1_WAKE_MASK) >>
UART_C1_WAKE_SHIFT)
#define UART_BRD_C1_WAKE(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_WAKE_SHIFT, UART_C1_WAKE_WIDTH))

/*! @brief Set the WAKE field to a new value. */
#define UART_WR_C1_WAKE(base, value) (UART_RMW_C1(base,
UART_C1_WAKE_MASK, UART_C1_WAKE(value)))
#define UART_BWR_C1_WAKE(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_WAKE_SHIFT), UART_C1_WAKE_SHIFT,
UART_C1_WAKE_WIDTH))
/*@}*/

/*!
* @name Register UART_C1, field M[4] (RW)
*
* Values:
* - 0b0 - Normal - start + 8 data bits (lsb first) + stop.
* - 0b1 - Receiver and transmitter use 9-bit data characters start + 8
data
*       bits (lsb first) + 9th data bit + stop.
*/
/*@{*/
/*! @brief Read current value of the UART_C1_M field. */
#define UART_RD_C1_M(base) ((UART_C1_REG(base) & UART_C1_M_MASK) >>
UART_C1_M_SHIFT)
#define UART_BRD_C1_M(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_M_SHIFT, UART_C1_M_WIDTH))

/*! @brief Set the M field to a new value. */
#define UART_WR_C1_M(base, value) (UART_RMW_C1(base, UART_C1_M_MASK,
UART_C1_M(value)))
#define UART_BWR_C1_M(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_M_SHIFT), UART_C1_M_SHIFT, UART_C1_M_WIDTH))
/*@}*/

/*!
* @name Register UART_C1, field RSRC[5] (RW)
*
* This bit has no meaning or effect unless the LOOPS bit is set to 1.
When
* LOOPS is set, the receiver input is internally connected to the TxD
pin and RSRC
* determines whether this connection is also connected to the
transmitter output.
*
* Values:
* - 0b0 - Provided LOOPS is set, RSRC is cleared, selects internal loop
back
*       mode and the UART does not use the RxD pins.
* - 0b1 - Single-wire UART mode where the TxD pin is connected to the
*       transmitter output and receiver input.
*/

```

```

/*@{*/
/*! @brief Read current value of the UART_C1_RSRC field. */
#define UART_RD_C1_RSRC(base) ((UART_C1_REG(base) & UART_C1_RSRC_MASK) >>
UART_C1_RSRC_SHIFT)
#define UART_BRD_C1_RSRC(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_RSRC_SHIFT, UART_C1_RSRC_WIDTH))

/*! @brief Set the RSRC field to a new value. */
#define UART_WR_C1_RSRC(base, value) (UART_RMW_C1(base,
UART_C1_RSRC_MASK, UART_C1_RSRC(value)))
#define UART_BWR_C1_RSRC(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_RSRC_SHIFT), UART_C1_RSRC_SHIFT,
UART_C1_RSRC_WIDTH))
/*@}*/

/*
 * @name Register UART_C1, field UARTSWAI[6] (RW)
 *
 * Values:
 * - 0b0 - UART clocks continue to run in wait mode so the UART can be
the
 *         source of an interrupt that wakes up the CPU.
 * - 0b1 - UART clocks freeze while CPU is in wait mode.
 */
/*@*/
/*! @brief Read current value of the UART_C1_UARTSWAI field. */
#define UART_RD_C1_UARTSWAI(base) ((UART_C1_REG(base) &
UART_C1_UARTSWAI_MASK) >> UART_C1_UARTSWAI_SHIFT)
#define UART_BRD_C1_UARTSWAI(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_UARTSWAI_SHIFT, UART_C1_UARTSWAI_WIDTH))

/*! @brief Set the UARTSWAI field to a new value. */
#define UART_WR_C1_UARTSWAI(base, value) (UART_RMW_C1(base,
UART_C1_UARTSWAI_MASK, UART_C1_UARTSWAI(value)))
#define UART_BWR_C1_UARTSWAI(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_UARTSWAI_SHIFT), UART_C1_UARTSWAI_SHIFT,
UART_C1_UARTSWAI_WIDTH))
/*@*/

/*
 * @name Register UART_C1, field LOOPS[7] (RW)
 *
 * Selects between loop back modes and normal 2-pin full-duplex modes.
When
 * LOOPS is set, the transmitter output is internally connected to the
receiver input.
 *
 * Values:
 * - 0b0 - Normal operation - RxD and TxD use separate pins.
 * - 0b1 - Loop mode or single-wire mode where transmitter outputs are
         internally connected to receiver input. (See RSRC bit.) RxD pin is
not used by UART.
 */
/*@*/

```

```

/*! @brief Read current value of the UART_C1_LOOPS field. */
#define UART_RD_C1_LOOPS(base) ((UART_C1_REG(base) & UART_C1_LOOPS_MASK)
>> UART_C1_LOOPS_SHIFT)
#define UART_BRD_C1_LOOPS(base) (BME_UBFX8(&UART_C1_REG(base),
UART_C1_LOOPS_SHIFT, UART_C1_LOOPS_WIDTH))

/*! @brief Set the LOOPS field to a new value. */
#define UART_WR_C1_LOOPS(base, value) (UART_RMW_C1(base,
UART_C1_LOOPS_MASK, UART_C1_LOOPS(value)))
#define UART_BWR_C1_LOOPS(base, value) (BME_BFI8(&UART_C1_REG(base),
((uint8_t)(value) << UART_C1_LOOPS_SHIFT), UART_C1_LOOPS_SHIFT,
UART_C1_LOOPS_WIDTH))
/*@}*/

/********************* ****
 * UART_C2 - UART Control Register 2
***** */

/*!
* @brief UART_C2 - UART Control Register 2 (RW)
*
* Reset value: 0x00U
*
* This register can be read or written at any time.
*/
/*!
* @name Constants and macros for entire UART_C2 register
*/
/*@{*/
#define UART_RD_C2(base) (UART_C2_REG(base))
#define UART_WR_C2(base, value) (UART_C2_REG(base) = (value))
#define UART_RMW_C2(base, mask, value) (UART_WR_C2(base,
(UART_RD_C2(base) & ~mask) | value))
#define UART_SET_C2(base, value) (BME_OR8(&UART_C2_REG(base),
(uint8_t)(value)))
#define UART_CLR_C2(base, value) (BME_AND8(&UART_C2_REG(base),
(uint8_t)(~value)))
#define UART_TOG_C2(base, value) (BME_XOR8(&UART_C2_REG(base),
(uint8_t)(value)))
/*@}*/

/*
* Constants & macros for individual UART_C2 bitfields
*/
/*!
* @name Register UART_C2, field SBK[0] (RW)
*
* Writing a 1 and then a 0 to SBK queues a break character in the
transmit data

```

```

 * stream. Additional break characters of 10 or 11 or 12, 13 or 14 or 15
if
 * BRK13 = 1, bit times of logic 0 are queued as long as SBK is set.
Depending on the
 * timing of the set and clear of SBK relative to the information
currently
 * being transmitted, a second break character may be queued before
software clears
 * SBK.
 *
 * Values:
 * - 0b0 - Normal transmitter operation.
 * - 0b1 - Queue break character(s) to be sent.
 */
/*@{*/
/*! @brief Read current value of the UART_C2_SBK field. */
#define UART_RD_C2_SBK(base) ((UART_C2_REG(base) & UART_C2_SBK_MASK) >>
UART_C2_SBK_SHIFT)
#define UART_BRD_C2_SBK(base) (BME_UBFX8(&UART_C2_REG(base),
UART_C2_SBK_SHIFT, UART_C2_SBK_WIDTH))

/*! @brief Set the SBK field to a new value. */
#define UART_WR_C2_SBK(base, value) (UART_RMW_C2(base, UART_C2_SBK_MASK,
UART_C2_SBK(value)))
#define UART_BWR_C2_SBK(base, value) (BME_BFI8(&UART_C2_REG(base),
((uint8_t)(value) << UART_C2_SBK_SHIFT), UART_C2_SBK_SHIFT,
UART_C2_SBK_WIDTH))
/*@}*/

/*!
 * @name Register UART_C2, field RWU[1] (RW)
 *
 * This bit can be written to 1 to place the UART receiver in a standby
state
 * where it waits for automatic hardware detection of a selected wakeup
condition.
 * The wakeup condition is an idle line between messages, WAKE = 0, idle-
line
 * wakeup, or a logic 1 in the most significant data bit in a character,
WAKE = 1,
 * address-mark wakeup. Application software sets RWU and, normally, a
selected
 * hardware condition automatically clears RWU.
 *
 * Values:
 * - 0b0 - Normal UART receiver operation.
 * - 0b1 - UART receiver in standby waiting for wakeup condition.
 */
/*@*/
/*! @brief Read current value of the UART_C2_RWU field. */
#define UART_RD_C2_RWU(base) ((UART_C2_REG(base) & UART_C2_RWU_MASK) >>
UART_C2_RWU_SHIFT)
#define UART_BRD_C2_RWU(base) (BME_UBFX8(&UART_C2_REG(base),
UART_C2_RWU_SHIFT, UART_C2_RWU_WIDTH))

```

```

/*! @brief Set the RWU field to a new value. */
#define UART_WR_C2_RWU(base, value) (UART_RMW_C2(base, UART_C2_RWU_MASK,
UART_C2_RWU(value)))
#define UART_BWR_C2_RWU(base, value) (BME_BFI8(&UART_C2_REG(base),
((uint8_t)(value) << UART_C2_RWU_SHIFT), UART_C2_RWU_SHIFT,
UART_C2_RWU_WIDTH))
/*@}*/

/*!!
 * @name Register UART_C2, field RE[2] (RW)
 *
 * When the UART receiver is off, the RxD pin reverts to being a general-purpose
 * port I/O pin. If LOOPS is set the RxD pin reverts to being a general-purpose
 * I/O pin even if RE is set.
 *
 * Values:
 * - 0b0 - Receiver off.
 * - 0b1 - Receiver on.
 */
/*@{*/
/*! @brief Read current value of the UART_C2_RE field. */
#define UART_RD_C2_RE(base) ((UART_C2_REG(base) & UART_C2_RE_MASK) >>
UART_C2_RE_SHIFT)
#define UART_BRD_C2_RE(base) (BME_UBFX8(&UART_C2_REG(base),
UART_C2_RE_SHIFT, UART_C2_RE_WIDTH))

/*! @brief Set the RE field to a new value. */
#define UART_WR_C2_RE(base, value) (UART_RMW_C2(base, UART_C2_RE_MASK,
UART_C2_RE(value)))
#define UART_BWR_C2_RE(base, value) (BME_BFI8(&UART_C2_REG(base),
((uint8_t)(value) << UART_C2_RE_SHIFT), UART_C2_RE_SHIFT,
UART_C2_RE_WIDTH))
/*@}*/

/*!!
 * @name Register UART_C2, field TE[3] (RW)
 *
 * TE must be 1 to use the UART transmitter. When TE is set, the UART
 * forces the
 * TxD pin to act as an output for the UART system. When the UART is
 * configured
 * for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the
 * direction of
 * traffic on the single UART communication line (TxD pin). TE can also
 * queue an
 * idle character by clearing TE then setting TE while a transmission is
 * in
 * progress. When TE is written to 0, the transmitter keeps control of
 * the port TxD
 * pin until any data, queued idle, or queued break character finishes
 * transmitting

```

```

* before allowing the pin to revert to a general-purpose I/O pin.
*
* Values:
* - 0b0 - Transmitter off.
* - 0b1 - Transmitter on.
*/
/*@{ */
/*! @brief Read current value of the UART_C2_TE field. */
#define UART_RD_C2_TE(base) ((UART_C2_REG(base) & UART_C2_TE_MASK) >>
UART_C2_TE_SHIFT)
#define UART_BRD_C2_TE(base) (BME_UBFX8(&UART_C2_REG(base),
UART_C2_TE_SHIFT, UART_C2_TE_WIDTH))

/*! @brief Set the TE field to a new value. */
#define UART_WR_C2_TE(base, value) (UART_RMW_C2(base, UART_C2_TE_MASK,
UART_C2_TE(value)))
#define UART_BWR_C2_TE(base, value) (BME_BFI8(&UART_C2_REG(base),
((uint8_t)(value) << UART_C2_TE_SHIFT), UART_C2_TE_SHIFT,
UART_C2_TE_WIDTH))
/*}@ */ */

/*!
* @name Register UART_C2, field ILIE[4] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from IDLE disabled; use polling.
* - 0b1 - Hardware interrupt requested when IDLE flag is 1.
*/
/*@{ */
/*! @brief Read current value of the UART_C2_ILIE field. */
#define UART_RD_C2_ILIE(base) ((UART_C2_REG(base) & UART_C2_ILIE_MASK) >>
UART_C2_ILIE_SHIFT)
#define UART_BRD_C2_ILIE(base) (BME_UBFX8(&UART_C2_REG(base),
UART_C2_ILIE_SHIFT, UART_C2_ILIE_WIDTH))

/*! @brief Set the ILIE field to a new value. */
#define UART_WR_C2_ILIE(base, value) (UART_RMW_C2(base,
UART_C2_ILIE_MASK, UART_C2_ILIE(value)))
#define UART_BWR_C2_ILIE(base, value) (BME_BFI8(&UART_C2_REG(base),
((uint8_t)(value) << UART_C2_ILIE_SHIFT), UART_C2_ILIE_SHIFT,
UART_C2_ILIE_WIDTH))
/*}@ */ */

/*!
* @name Register UART_C2, field RIE[5] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from RDRF disabled; use polling.
* - 0b1 - Hardware interrupt requested when RDRF flag is 1.
*/
/*@{ */
/*! @brief Read current value of the UART_C2_RIE field. */
#define UART_RD_C2_RIE(base) ((UART_C2_REG(base) & UART_C2_RIE_MASK) >>
UART_C2_RIE_SHIFT)

```

```

#define UART_BRD_C2_RIE(base)  (BME_UBFX8(&UART_C2_REG(base),  

UART_C2_RIE_SHIFT, UART_C2_RIE_WIDTH))

/*! @brief Set the RIE field to a new value. */  

#define UART_WR_C2_RIE(base, value)  (UART_RMW_C2(base, UART_C2_RIE_MASK,  

UART_C2_RIE(value)))  

#define UART_BWR_C2_RIE(base, value)  (BME_BFI8(&UART_C2_REG(base),  

(uint8_t)(value) << UART_C2_RIE_SHIFT), UART_C2_RIE_SHIFT,  

UART_C2_RIE_WIDTH))  

/*@}*/

/*!  

 * @name Register UART_C2, field TCIE[6] (RW)  

 *  

 * Values:  

 * - 0b0 - Hardware interrupts from TC disabled; use polling.  

 * - 0b1 - Hardware interrupt requested when TC flag is 1.  

 */  

/*@{*/  

/*! @brief Read current value of the UART_C2_TCIE field. */  

#define UART_RD_C2_TCIE(base)  ((UART_C2_REG(base) & UART_C2_TCIE_MASK) >>  

UART_C2_TCIE_SHIFT)  

#define UART_BRD_C2_TCIE(base)  (BME_UBFX8(&UART_C2_REG(base),  

UART_C2_TCIE_SHIFT, UART_C2_TCIE_WIDTH))

/*! @brief Set the TCIE field to a new value. */  

#define UART_WR_C2_TCIE(base, value)  (UART_RMW_C2(base,  

UART_C2_TCIE_MASK, UART_C2_TCIE(value)))  

#define UART_BWR_C2_TCIE(base, value)  (BME_BFI8(&UART_C2_REG(base),  

(uint8_t)(value) << UART_C2_TCIE_SHIFT), UART_C2_TCIE_SHIFT,  

UART_C2_TCIE_WIDTH))  

/*@}*/

/*!  

 * @name Register UART_C2, field TIE[7] (RW)  

 *  

 * Values:  

 * - 0b0 - Hardware interrupts from TDRE disabled; use polling.  

 * - 0b1 - Hardware interrupt requested when TDRE flag is 1.  

 */  

/*@{*/  

/*! @brief Read current value of the UART_C2_TIE field. */  

#define UART_RD_C2_TIE(base)  ((UART_C2_REG(base) & UART_C2_TIE_MASK) >>  

UART_C2_TIE_SHIFT)  

#define UART_BRD_C2_TIE(base)  (BME_UBFX8(&UART_C2_REG(base),  

UART_C2_TIE_SHIFT, UART_C2_TIE_WIDTH))

/*! @brief Set the TIE field to a new value. */  

#define UART_WR_C2_TIE(base, value)  (UART_RMW_C2(base, UART_C2_TIE_MASK,  

UART_C2_TIE(value)))  

#define UART_BWR_C2_TIE(base, value)  (BME_BFI8(&UART_C2_REG(base),  

(uint8_t)(value) << UART_C2_TIE_SHIFT), UART_C2_TIE_SHIFT,  

UART_C2_TIE_WIDTH))  

/*@}*/

```

```

/*****
 * UART_S1 - UART Status Register 1
 *****/
/*!
 * @brief UART_S1 - UART Status Register 1 (RO)
 *
 * Reset value: 0xC0U
 *
 * This register has eight read-only status flags. Writes have no effect.
 * Special software sequences, which do not involve writing to this
register, clear
 * these status flags.
 */
/*!
 * @name Constants and macros for entire UART_S1 register
 */
/*@*/
#define UART_RD_S1(base)           (UART_S1_REG(base))
/*@*/



/*
 * Constants & macros for individual UART_S1 bitfields
 */

/*!
 * @name Register UART_S1, field PF[0] (RO)
 *
 * PF is set at the same time as RDRE when parity is enabled (PE = 1) and
the
 * parity bit in the received character does not agree with the expected
parity
 * value. To clear PF, read UART_S1 and then read the UART data register
(UART_D).
 *
 * Values:
 * - 0b0 - No parity error.
 * - 0b1 - Parity error.
 */
/*@*/
/*! @brief Read current value of the UART_S1_PF field. */
#define UART_RD_S1_PF(base) ((UART_S1_REG(base) & UART_S1_PF_MASK) >>
UART_S1_PF_SHIFT)
#define UART_BRD_S1_PF(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_PF_SHIFT, UART_S1_PF_WIDTH))
/*@*/



/*!
 * @name Register UART_S1, field FE[1] (RO)
 *

```

```

    * FE is set at the same time as RDRF when the receiver detects a logic 0
    where
        * the stop bits was expected. This suggests the receiver was not
        properly
        * aligned to a character frame. To clear FE, read UART_S1 with FE set
        and then read
            * the UART data register (UART_D).
        *
        * Values:
        * - 0b0 - No framing error detected. This does not guarantee the framing
        is
            * correct.
        * - 0b1 - Framing error.
        */
/*@{*/
/*! @brief Read current value of the UART_S1_FE field. */
#define UART_RD_S1_FE(base) ((UART_S1_REG(base) & UART_S1_FE_MASK) >>
UART_S1_FE_SHIFT)
#define UART_BRD_S1_FE(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_FE_SHIFT, UART_S1_FE_WIDTH))
/*@}*/
/*!
 * @name Register UART_S1, field NF[2] (RO)
 *
 * The advanced sampling technique used in the receiver takes seven
samples
 * during the start bit and three samples in each data bit and the stop
bits. If any
 * of these samples disagrees with the rest of the samples within any bit
time in
 * the frame, the flag NF is set at the same time as RDRF is set for the
 * character. To clear NF, read UART_S1 and then read the UART data
register (UART_D).
 *
 * Values:
 * - 0b0 - No noise detected.
 * - 0b1 - Noise detected in the received character in UART_D.
 */
/*@{*/
/*! @brief Read current value of the UART_S1_NF field. */
#define UART_RD_S1_NF(base) ((UART_S1_REG(base) & UART_S1_NF_MASK) >>
UART_S1_NF_SHIFT)
#define UART_BRD_S1_NF(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_NF_SHIFT, UART_S1_NF_WIDTH))
/*@}*/
/*!
 * @name Register UART_S1, field OR[3] (RO)
 *
 * OR is set when a new serial character is ready to be transferred to
the
 * receive data register (buffer), but the previously received character
has not been

```

```

    * read from UART_D yet. In this case, the new character, and all
associated
    * error information, is lost because there is no room to move it into
UART_D. To
    * clear OR, read UART_S1 with OR set and then read the UART data
register (UART_D).
    *
    * Values:
    * - 0b0 - No overrun.
    * - 0b1 - Receive overrun (new UART data lost).
    */
/*@{*/
/*! @brief Read current value of the UART_S1_OR field. */
#define UART_RD_S1_OR(base) ((UART_S1_REG(base) & UART_S1_OR_MASK) >>
UART_S1_OR_SHIFT)
#define UART_BRD_S1_OR(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_OR_SHIFT, UART_S1_OR_WIDTH))
/*}@*/ */

/*!!
* @name Register UART_S1, field IDLE[4] (RO)
*
* IDLE is set when the UART receive line becomes idle for a full
character time
* after a period of activity. When ILT is cleared, the receiver starts
counting
* idle bit times after the start bit. If the receive character is all
1s, these
* bit times and the stop bits time count toward the full character time
of
* logic high, 10 or 11 bit times depending on the M control bit, needed
for the
* receiver to detect an idle line. When ILT is set, the receiver doesn't
start
* counting idle bit times until after the stop bits. The stop bits and
any logic high
* bit times at the end of the previous character do not count toward the
full
* character time of logic high needed for the receiver to detect an idle
line. To
* clear IDLE, read UART_S1 with IDLE set and then read the UART data
register
* (UART_D). After IDLE has been cleared, it cannot become set again
until after a
* new character has been received and RDRF has been set. IDLE is set
only once
* even if the receive line remains idle for an extended period.
*
* Values:
* - 0b0 - No idle line detected.
* - 0b1 - Idle line was detected.
*/
/*@{*/
/*! @brief Read current value of the UART_S1_IDLE field. */

```

```

#define UART_RD_S1_IDLE(base) ((UART_S1_REG(base) & UART_S1_IDLE_MASK) >>
UART_S1_IDLE_SHIFT)
#define UART_BRD_S1_IDLE(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_IDLE_SHIFT, UART_S1_IDLE_WIDTH))
/*@}*/

/*!!
 * @name Register UART_S1, field RDRF[5] (RO)
 *
 * RDRF becomes set when a character transfers from the receive shifter
into the
 * receive data register (UART_D). To clear RDRF, read UART_S1 with RDRF
set and
 * then read the UART data register (UART_D).
 *
 * Values:
 * - 0b0 - Receive data register empty.
 * - 0b1 - Receive data register full.
 */
/*@{*/
/*! @brief Read current value of the UART_S1_RDRF field. */
#define UART_RD_S1_RDRF(base) ((UART_S1_REG(base) & UART_S1_RDRF_MASK) >>
UART_S1_RDRF_SHIFT)
#define UART_BRD_S1_RDRF(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_RDRF_SHIFT, UART_S1_RDRF_WIDTH))
/*@}*/

/*!!
 * @name Register UART_S1, field TC[6] (RO)
 *
 * TC is set out of reset and when TDRE is set and no data, preamble, or
break
 * character is being transmitted. TC is cleared automatically by reading
UART_S1
 * with TC set and then doing one of the following: Write to the UART
data
 * register (UART_D) to transmit new data Queue a preamble by changing TE
from 0 to 1
 * Queue a break character by writing 1 to UART_C2[SBK]
 *
 * Values:
 * - 0b0 - Transmitter active (sending data, a preamble, or a break).
 * - 0b1 - Transmitter idle (transmission activity complete).
 */
/*@{*/
/*! @brief Read current value of the UART_S1_TC field. */
#define UART_RD_S1_TC(base) ((UART_S1_REG(base) & UART_S1_TC_MASK) >>
UART_S1_TC_SHIFT)
#define UART_BRD_S1_TC(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_TC_SHIFT, UART_S1_TC_WIDTH))
/*@}*/

/*!!
 * @name Register UART_S1, field TDRE[7] (RO)

```

```

/*
 * TDRE is set out of reset and when a transmit data value transfers from
the
 * transmit data buffer to the transmit shifter, leaving room for a new
character
 * in the buffer. To clear TDRE, read UART_S1 with TDRE set and then
write to the
 * UART data register (UART_D).
*
* Values:
* - 0b0 - Transmit data register (buffer) full.
* - 0b1 - Transmit data register (buffer) empty.
*/
/*@{*/
/*! @brief Read current value of the UART_S1_TDRE field. */
#define UART_RD_S1_TDRE(base) ((UART_S1_REG(base) & UART_S1_TDRE_MASK) >>
UART_S1_TDRE_SHIFT)
#define UART_BRD_S1_TDRE(base) (BME_UBFX8(&UART_S1_REG(base),
UART_S1_TDRE_SHIFT, UART_S1_TDRE_WIDTH))
/*}@*/



/***** ****
* UART_S2 - UART Status Register 2
***** */

/*!
* @brief UART_S2 - UART Status Register 2 (RW)
*
* Reset value: 0x00U
*
* This register contains one read-only status flag. When using an
internal
* oscillator in a LIN system, it is necessary to raise the break
detection threshold
* one bit time. Under the worst case timing conditions allowed in LIN,
it is
* possible that a 0x00 data character can appear to be 10.26 bit times
long at a
* slave running 14% faster than the master. This would trigger normal
break
* detection circuitry designed to detect a 10-bit break symbol. When the
LBKDE bit is
* set, framing errors are inhibited and the break detection threshold
changes
* from 10 bits to 11 bits, preventing false detection of a 0x00 data
character as
* a LIN break symbol.
*/
/*!
* @name Constants and macros for entire UART_S2 register
*/

```

```

/*@{*/
#define UART_RD_S2(base)          (UART_S2_REG(base))
#define UART_WR_S2(base, value)   (UART_S2_REG(base) = (value))
#define UART_RMW_S2(base, mask, value) (UART_WR_S2(base,
(UART_RD_S2(base) & ~mask) | (value)))
#define UART_SET_S2(base, value)  (BME_OR8(&UART_S2_REG(base),
(uint8_t)(value)))
#define UART_CLR_S2(base, value)  (BME_AND8(&UART_S2_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_S2(base, value)  (BME_XOR8(&UART_S2_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART_S2 bitfields
 */

/*!
 * @name Register UART_S2, field RAF[0] (RO)
 *
 * RAF is set when the UART receiver detects the beginning of a valid
start bit,
 * and RAF is cleared automatically when the receiver detects an idle
line. This
 * status flag can be used to check whether an UART character is being
received
 * before instructing the MCU to go to stop mode.
 *
 * Values:
 * - 0b0 - UART receiver idle waiting for a start bit.
 * - 0b1 - UART receiver active (RxD input not idle).
 */
/*@*/
/*! @brief Read current value of the UART_S2_RAF field. */
#define UART_RD_S2_RAF(base) ((UART_S2_REG(base) & UART_S2_RAF_MASK) >>
UART_S2_RAF_SHIFT)
#define UART_BRD_S2_RAF(base) (BME_UBFX8(&UART_S2_REG(base),
UART_S2_RAF_SHIFT, UART_S2_RAF_WIDTH))
/*@*/



/*!
 * @name Register UART_S2, field LBKDE[1] (RW)
 *
 * LBKDE selects a longer break character detection length. While LBKDE
is set,
 * framing error (FE) and receive data register full (RDRF) flags are
prevented
 * from setting.
 *
 * Values:
 * - 0b0 - Break character is detected at length 10 bit times (if M = 0,
SBNS =
 *      0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1,
SBNS =

```

```

        1).
* - 0b1 - Break character is detected at length of 11 bit times (if M =
0, SBNS
*      = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 13 (if M =
1, SBNS
*      = 1).
*/
/*@{*/
/*! @brief Read current value of the UART_S2_LBKDE field. */
#define UART_RD_S2_LBKDE(base) ((UART_S2_REG(base) & UART_S2_LBKDE_MASK)
>> UART_S2_LBKDE_SHIFT)
#define UART_BRD_S2_LBKDE(base) (BME_UBXF8(&UART_S2_REG(base),
UART_S2_LBKDE_SHIFT, UART_S2_LBKDE_WIDTH))

/*! @brief Set the LBKDE field to a new value. */
#define UART_WR_S2_LBKDE(base, value) (UART_RMW_S2(base,
UART_S2_LBKDE_MASK, UART_S2_LBKDE(value)))
#define UART_BWR_S2_LBKDE(base, value) (BME_BFI8(&UART_S2_REG(base),
((uint8_t)(value) << UART_S2_LBKDE_SHIFT), UART_S2_LBKDE_SHIFT,
UART_S2_LBKDE_WIDTH))
/*}@*/ */

/*!
* @name Register UART_S2, field BRK13[2] (RW)
*
* BRK13 selects a longer transmitted break character length. Detection
of a
* framing error is not affected by the state of this bit.
*
* Values:
* - 0b0 - Break character is transmitted with length of 10 bit times (if
M = 0,
*      SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if
M = 1,
*      SBNS = 1).
* - 0b1 - Break character is transmitted with length of 13 bit times (if
M = 0,
*      SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if
M = 1,
*      SBNS = 1).
*/
/*@{*/
/*! @brief Read current value of the UART_S2_BRK13 field. */
#define UART_RD_S2_BRK13(base) ((UART_S2_REG(base) & UART_S2_BRK13_MASK)
>> UART_S2_BRK13_SHIFT)
#define UART_BRD_S2_BRK13(base) (BME_UBXF8(&UART_S2_REG(base),
UART_S2_BRK13_SHIFT, UART_S2_BRK13_WIDTH))

/*! @brief Set the BRK13 field to a new value. */
#define UART_WR_S2_BRK13(base, value) (UART_RMW_S2(base,
UART_S2_BRK13_MASK, UART_S2_BRK13(value)))
#define UART_BWR_S2_BRK13(base, value) (BME_BFI8(&UART_S2_REG(base),
((uint8_t)(value) << UART_S2_BRK13_SHIFT), UART_S2_BRK13_SHIFT,
UART_S2_BRK13_WIDTH))

```

```

/*@}*/

/*
 * @name Register UART_S2, field RWUID[3] (RW)
 *
 * RWUID controls whether the idle character that wakes up the receiver
sets the
 * IDLE bit.
 *
 * Values:
 * - 0b0 - During receive standby state (RWU = 1), the IDLE bit does not
get set
 *      upon detection of an idle character.
 * - 0b1 - During receive standby state (RWU = 1), the IDLE bit gets set
upon
 *      detection of an idle character.
 */
/*@{*/
/*! @brief Read current value of the UART_S2_RXINV field. */
#define UART_RD_S2_RXINV(base) ((UART_S2_REG(base) & UART_S2_RXINV_MASK)
>> UART_S2_RXINV_SHIFT)
#define UART_BRD_S2_RXINV(base) (BME_UBXF8(&UART_S2_REG(base),
UART_S2_RXINV_SHIFT, UART_S2_RXINV_WIDTH))

/*! @brief Set the RXINV field to a new value. */
#define UART_WR_S2_RXINV(base, value) (UART_RMW_S2(base,
UART_S2_RXINV_MASK, UART_S2_RXINV(value)))
#define UART_BWR_S2_RXINV(base, value) (BME_BFI8(&UART_S2_REG(base),
((uint8_t)(value) << UART_S2_RXINV_SHIFT), UART_S2_RXINV_SHIFT,
UART_S2_RXINV_WIDTH))
/*@}*/

/*
 * @name Register UART_S2, field RXINV[4] (RW)
 *
 * Setting this bit reverses the polarity of the received data input.
Setting
 * RXINV inverts the RxD input for all cases: data bits, start and stop
bits,
 * break, and idle.
 *
 * Values:
 * - 0b0 - Receive data not inverted.
 * - 0b1 - Receive data inverted.
 */
/*@{*/
/*! @brief Read current value of the UART_S2_RXINV field. */
#define UART_RD_S2_RXINV(base) ((UART_S2_REG(base) & UART_S2_RXINV_MASK)
>> UART_S2_RXINV_SHIFT)
#define UART_BRD_S2_RXINV(base) (BME_UBXF8(&UART_S2_REG(base),
UART_S2_RXINV_SHIFT, UART_S2_RXINV_WIDTH))

/*! @brief Set the RXINV field to a new value. */

```

```

#define UART_WR_S2_RXINV(base, value) (UART_RMW_S2(base,
UART_S2_RXINV_MASK, UART_S2_RXINV(value)))
#define UART_BWR_S2_RXINV(base, value) (BME_BFI8(&UART_S2_REG(base),
((uint8_t)(value) << UART_S2_RXINV_SHIFT), UART_S2_RXINV_SHIFT,
UART_S2_RXINV_WIDTH))
/*@}*/

/*
 * @name Register UART_S2, field RXEDGIF[6] (RW)
 *
 * RXEDGIF is set when an active edge, falling if RXINV = 0, rising if
RXINV=1,
 * on the RxD pin occurs. RXEDGIF is cleared by writing a 1 to it.
 *
 * Values:
 * - 0b0 - No active edge on the receive pin has occurred.
 * - 0b1 - An active edge on the receive pin has occurred.
 */
/*@{*/
/*! @brief Read current value of the UART_S2_RXEDGIF field. */
#define UART_RD_S2_RXEDGIF(base) ((UART_S2_REG(base) &
UART_S2_RXEDGIF_MASK) >> UART_S2_RXEDGIF_SHIFT)
#define UART_BRD_S2_RXEDGIF(base) (BME_UBFX8(&UART_S2_REG(base),
UART_S2_RXEDGIF_SHIFT, UART_S2_RXEDGIF_WIDTH))

/*! @brief Set the RXEDGIF field to a new value. */
#define UART_WR_S2_RXEDGIF(base, value) (UART_RMW_S2(base,
UART_S2_RXEDGIF_MASK, UART_S2_RXEDGIF(value)))
#define UART_BWR_S2_RXEDGIF(base, value) (BME_BFI8(&UART_S2_REG(base),
((uint8_t)(value) << UART_S2_RXEDGIF_SHIFT), UART_S2_RXEDGIF_SHIFT,
UART_S2_RXEDGIF_WIDTH))
/*@}*/

/*
 * @name Register UART_S2, field LBKDIF[7] (RW)
 *
 * LBKDIF is set when the LIN break detect circuitry is enabled and a LIN
break
 * character is detected. LBKDIF is cleared by writing a 1 to it.
 *
 * Values:
 * - 0b0 - No LIN break character has been detected.
 * - 0b1 - LIN break character has been detected.
 */
/*@{*/
/*! @brief Read current value of the UART_S2_LBKDIF field. */
#define UART_RD_S2_LBKDIF(base) ((UART_S2_REG(base) &
UART_S2_LBKDIF_MASK) >> UART_S2_LBKDIF_SHIFT)
#define UART_BRD_S2_LBKDIF(base) (BME_UBFX8(&UART_S2_REG(base),
UART_S2_LBKDIF_SHIFT, UART_S2_LBKDIF_WIDTH))

/*! @brief Set the LBKDIF field to a new value. */
#define UART_WR_S2_LBKDIF(base, value) (UART_RMW_S2(base,
UART_S2_LBKDIF_MASK, UART_S2_LBKDIF(value)))

```

```

#define UART_BWR_S2_LBKDIF(base, value) (BME_BFI8(&UART_S2_REG(base), \
((uint8_t)(value) << UART_S2_LBKDIF_SHIFT), UART_S2_LBKDIF_SHIFT, \
UART_S2_LBKDIF_WIDTH))
/*@}*/

/********************* UART_C3 - UART Control Register 3 ********************
*****
 * UART_C3 - UART Control Register 3
*****
 *****/
/*!
 * @brief UART_C3 - UART Control Register 3 (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire UART_C3 register
 */
/*@*/
#define UART_RD_C3(base) (UART_C3_REG(base))
#define UART_WR_C3(base, value) (UART_C3_REG(base) = (value))
#define UART_RMW_C3(base, mask, value) (UART_WR_C3(base, \
(UART_RD_C3(base) & ~mask) | (value)))
#define UART_SET_C3(base, value) (BME_OR8(&UART_C3_REG(base), \
(uint8_t)(value)))
#define UART_CLR_C3(base, value) (BME_AND8(&UART_C3_REG(base), \
(uint8_t)(~(value))))
#define UART_TOG_C3(base, value) (BME_XOR8(&UART_C3_REG(base), \
(uint8_t)(value)))
/*@*/ */

/*
 * Constants & macros for individual UART_C3 bitfields
 */
/*!
 * @name Register UART_C3, field PEIE[0] (RW)
 *
 * This bit enables the parity error flag (PF) to generate hardware
interrupt
 * requests.
 *
 * Values:
 * - 0b0 - PF interrupts disabled; use polling).
 * - 0b1 - Hardware interrupt requested when PF is set.
 */
/*@*/
/*! @brief Read current value of the UART_C3_PEIE field. */
#define UART_RD_C3_PEIE(base) ((UART_C3_REG(base) & UART_C3_PEIE_MASK) >> \
UART_C3_PEIE_SHIFT)
#define UART_BRD_C3_PEIE(base) (BME_UBFX8(&UART_C3_REG(base), \
UART_C3_PEIE_SHIFT, UART_C3_PEIE_WIDTH))

```

```

/*! @brief Set the PEIE field to a new value. */
#define UART_WR_C3_PEIE(base, value) (UART_RMW_C3(base,
UART_C3_PEIE_MASK, UART_C3_PEIE(value)))
#define UART_BWR_C3_PEIE(base, value) (BME_BFI8(&UART_C3_REG(base),
((uint8_t)(value) << UART_C3_PEIE_SHIFT), UART_C3_PEIE_SHIFT,
UART_C3_PEIE_WIDTH))
/*@}*/

/*!!
 * @name Register UART_C3, field FEIE[1] (RW)
 *
 * This bit enables the framing error flag (FE) to generate hardware
interrupt
 * requests.
 *
 * Values:
 * - 0b0 - FE interrupts disabled; use polling).
 * - 0b1 - Hardware interrupt requested when FE is set.
 */
/*@{*/
/*! @brief Read current value of the UART_C3_FEIE field. */
#define UART_RD_C3_FEIE(base) ((UART_C3_REG(base) & UART_C3_FEIE_MASK) >>
UART_C3_FEIE_SHIFT)
#define UART_BRD_C3_FEIE(base) (BME_UBFX8(&UART_C3_REG(base),
UART_C3_FEIE_SHIFT, UART_C3_FEIE_WIDTH))

/*! @brief Set the FEIE field to a new value. */
#define UART_WR_C3_FEIE(base, value) (UART_RMW_C3(base,
UART_C3_FEIE_MASK, UART_C3_FEIE(value)))
#define UART_BWR_C3_FEIE(base, value) (BME_BFI8(&UART_C3_REG(base),
((uint8_t)(value) << UART_C3_FEIE_SHIFT), UART_C3_FEIE_SHIFT,
UART_C3_FEIE_WIDTH))
/*@}*/

/*!!
 * @name Register UART_C3, field NEIE[2] (RW)
 *
 * This bit enables the noise flag (NF) to generate hardware interrupt
requests.
 *
 * Values:
 * - 0b0 - NF interrupts disabled; use polling).
 * - 0b1 - Hardware interrupt requested when NF is set.
 */
/*@{*/
/*! @brief Read current value of the UART_C3_NEIE field. */
#define UART_RD_C3_NEIE(base) ((UART_C3_REG(base) & UART_C3_NEIE_MASK) >>
UART_C3_NEIE_SHIFT)
#define UART_BRD_C3_NEIE(base) (BME_UBFX8(&UART_C3_REG(base),
UART_C3_NEIE_SHIFT, UART_C3_NEIE_WIDTH))

/*! @brief Set the NEIE field to a new value. */

```

```

#define UART_WR_C3_NEIE(base, value) (UART_RMW_C3(base,
UART_C3_NEIE_MASK, UART_C3_NEIE(value)))
#define UART_BWR_C3_NEIE(base, value) (BME_BFI8(&UART_C3_REG(base),
((uint8_t)(value) << UART_C3_NEIE_SHIFT), UART_C3_NEIE_SHIFT,
UART_C3_NEIE_WIDTH))
/*@}*/

/*
 * @name Register UART_C3, field ORIE[3] (RW)
 *
 * This bit enables the overrun flag (OR) to generate hardware interrupt
 * requests.
 *
 * Values:
 * - 0b0 - OR interrupts disabled; use polling.
 * - 0b1 - Hardware interrupt requested when OR is set.
 */
/*@{*/
/*! @brief Read current value of the UART_C3_ORIE field. */
#define UART_RD_C3_ORIE(base) ((UART_C3_REG(base) & UART_C3_ORIE_MASK) >>
UART_C3_ORIE_SHIFT)
#define UART_BRD_C3_ORIE(base) (BME_UBFX8(&UART_C3_REG(base),
UART_C3_ORIE_SHIFT, UART_C3_ORIE_WIDTH))

/*! @brief Set the ORIE field to a new value. */
#define UART_WR_C3_ORIE(base, value) (UART_RMW_C3(base,
UART_C3_ORIE_MASK, UART_C3_ORIE(value)))
#define UART_BWR_C3_ORIE(base, value) (BME_BFI8(&UART_C3_REG(base),
((uint8_t)(value) << UART_C3_ORIE_SHIFT), UART_C3_ORIE_SHIFT,
UART_C3_ORIE_WIDTH))
/*@}*/

/*
 * @name Register UART_C3, field TXINV[4] (RW)
 *
 * Setting this bit reverses the polarity of the transmitted data output.
 * Setting TXINV inverts the TxD output for all cases: data bits, start
and stop bits,
 * break, and idle.
 *
 * Values:
 * - 0b0 - Transmit data not inverted.
 * - 0b1 - Transmit data inverted.
 */
/*@{*/
/*! @brief Read current value of the UART_C3_TXINV field. */
#define UART_RD_C3_TXINV(base) ((UART_C3_REG(base) & UART_C3_TXINV_MASK)
>> UART_C3_TXINV_SHIFT)
#define UART_BRD_C3_TXINV(base) (BME_UBFX8(&UART_C3_REG(base),
UART_C3_TXINV_SHIFT, UART_C3_TXINV_WIDTH))

/*! @brief Set the TXINV field to a new value. */
#define UART_WR_C3_TXINV(base, value) (UART_RMW_C3(base,
UART_C3_TXINV_MASK, UART_C3_TXINV(value)))

```

```

#define UART_BWR_C3_TXINV(base, value) (BME_BFI8(&UART_C3_REG(base),  

((uint8_t)(value) << UART_C3_TXINV_SHIFT), UART_C3_TXINV_SHIFT,  

UART_C3_TXINV_WIDTH))  

/*@}*/

/*!  

 * @name Register UART_C3, field TXDIR[5] (RW)  

 *  

 * When the UART is configured for single-wire half-duplex operation  

(LOOPS =  

 * RSRC = 1), this bit determines the direction of data at the TxD pin.  

 *  

 * Values:  

 * - 0b0 - TxD pin is an input in single-wire mode.  

 * - 0b1 - TxD pin is an output in single-wire mode.  

 */  

/*@{*/  

/*! @brief Read current value of the UART_C3_TXDIR field. */  

#define UART_RD_C3_TXDIR(base) ((UART_C3_REG(base) & UART_C3_TXDIR_MASK)  

>> UART_C3_TXDIR_SHIFT)  

#define UART_BRD_C3_TXDIR(base) (BME_UBFX8(&UART_C3_REG(base),  

UART_C3_TXDIR_SHIFT, UART_C3_TXDIR_WIDTH))

/*! @brief Set the TXDIR field to a new value. */  

#define UART_WR_C3_TXDIR(base, value) (UART_RMW_C3(base,  

UART_C3_TXDIR_MASK, UART_C3_TXDIR(value)))  

#define UART_BWR_C3_TXDIR(base, value) (BME_BFI8(&UART_C3_REG(base),  

((uint8_t)(value) << UART_C3_TXDIR_SHIFT), UART_C3_TXDIR_SHIFT,  

UART_C3_TXDIR_WIDTH))  

/*@}*/

/*!  

 * @name Register UART_C3, field T8[6] (RW)  

 *  

 * When the UART is configured for 9-bit data (M = 1), T8 may be thought  

of as a  

 * ninth transmit data bit to the left of the msb of the data in the  

UART_D  

 * register. When writing 9-bit data, the entire 9-bit value is  

transferred to the  

 * UART shift register after UART_D is written so T8 should be written,  

if it needs  

 * to change from its previous value, before UART_D is written. If T8  

does not  

 * need to change in the new value, such as when it is used to generate  

mark or  

 * space parity, it need not be written each time UART_D is written.  

 */  

/*@{*/  

/*! @brief Read current value of the UART_C3_T8 field. */  

#define UART_RD_C3_T8(base) ((UART_C3_REG(base) & UART_C3_T8_MASK) >>  

UART_C3_T8_SHIFT)  

#define UART_BRD_C3_T8(base) (BME_UBFX8(&UART_C3_REG(base),  

UART_C3_T8_SHIFT, UART_C3_T8_WIDTH))

```

```

/*! @brief Set the T8 field to a new value. */
#define UART_WR_C3_T8(base, value) (UART_RMW_C3(base, UART_C3_T8_MASK,
UART_C3_T8(value)))
#define UART_BWR_C3_T8(base, value) (BME_BFI8(&UART_C3_REG(base),
((uint8_t)(value) << UART_C3_T8_SHIFT), UART_C3_T8_SHIFT,
UART_C3_T8_WIDTH))
/*@}*/

/*!!
 * @name Register UART_C3, field R8[7] (RO)
 *
 * When the UART is configured for 9-bit data (M = 1), R8 can be thought
 * of as a
 * ninth receive data bit to the left of the msb of the buffered data in
 * the
 * UART_D register. When reading 9-bit data, read R8 before reading
 * UART_D because
 * reading UART_D completes automatic flag clearing sequences that could
 * allow R8
 * and UART_D to be overwritten with new data.
 */
/*@{*/
/*! @brief Read current value of the UART_C3_R8 field. */
#define UART_RD_C3_R8(base) ((UART_C3_REG(base) & UART_C3_R8_MASK) >>
UART_C3_R8_SHIFT)
#define UART_BRD_C3_R8(base) (BME_UBFX8(&UART_C3_REG(base),
UART_C3_R8_SHIFT, UART_C3_R8_WIDTH))
/*@}*/

/******************
*****
 * UART_D - UART Data Register
*****
******************/

/*!!
 * @brief UART_D - UART Data Register (RW)
 *
 * Reset value: 0x00U
 *
 * This register is actually two separate registers. Reads return the
 * contents
 * of the read-only receive data buffer and writes go to the write-only
 * transmit
 * data buffer. Reads and writes of this register are also involved in
 * the
 * automatic flag clearing mechanisms for the UART status flags.
 */
/*!
 * @name Constants and macros for entire UART_D register
 */
/*@{*/

```

```

#define UART_RD_D(base)           (UART_D_REG(base))
#define UART_WR_D(base, value)    (UART_D_REG(base) = (value))
#define UART_RMW_D(base, mask, value) (UART_WR_D(base, (UART_RD_D(base) &
~(mask)) | (value)))
#define UART_SET_D(base, value)   (BME_OR8(&UART_D_REG(base),
(uint8_t)(value)))
#define UART_CLR_D(base, value)   (BME_AND8(&UART_D_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_D(base, value)   (BME_XOR8(&UART_D_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART_D bitfields
 */

/*!!
 * @name Register UART_D, field R0T0[0] (RW)
 *
 * Read receive data buffer 0 or write transmit data buffer 0.
 */
/*@{*/
/**! @brief Read current value of the UART_D_R0T0 field. */
#define UART_RD_D_R0T0(base) ((UART_D_REG(base) & UART_D_R0T0_MASK) >>
UART_D_R0T0_SHIFT)
#define UART_BRD_D_R0T0(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R0T0_SHIFT, UART_D_R0T0_WIDTH))

/**! @brief Set the R0T0 field to a new value. */
#define UART_WR_D_R0T0(base, value) (UART_RMW_D(base, UART_D_R0T0_MASK,
UART_D_R0T0(value)))
#define UART_BWR_D_R0T0(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R0T0_SHIFT), UART_D_R0T0_SHIFT,
UART_D_R0T0_WIDTH))
/*@}*/

/*!!
 * @name Register UART_D, field R1T1[1] (RW)
 *
 * Read receive data buffer 1 or write transmit data buffer 1.
 */
/*@{*/
/**! @brief Read current value of the UART_D_R1T1 field. */
#define UART_RD_D_R1T1(base) ((UART_D_REG(base) & UART_D_R1T1_MASK) >>
UART_D_R1T1_SHIFT)
#define UART_BRD_D_R1T1(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R1T1_SHIFT, UART_D_R1T1_WIDTH))

/**! @brief Set the R1T1 field to a new value. */
#define UART_WR_D_R1T1(base, value) (UART_RMW_D(base, UART_D_R1T1_MASK,
UART_D_R1T1(value)))
#define UART_BWR_D_R1T1(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R1T1_SHIFT), UART_D_R1T1_SHIFT,
UART_D_R1T1_WIDTH))

```

```

/*@}*/

/*
 * @name Register UART_D, field R2T2[2] (RW)
 *
 * Read receive data buffer 2 or write transmit data buffer 2.
 */
/*@{*/
/*! @brief Read current value of the UART_D_R2T2 field. */
#define UART_RD_D_R2T2(base) ((UART_D_REG(base) & UART_D_R2T2_MASK) >>
UART_D_R2T2_SHIFT)
#define UART_BRD_D_R2T2(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R2T2_SHIFT, UART_D_R2T2_WIDTH))

/*! @brief Set the R2T2 field to a new value. */
#define UART_WR_D_R2T2(base, value) (UART_RMW_D(base, UART_D_R2T2_MASK,
UART_D_R2T2(value)))
#define UART_BWR_D_R2T2(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R2T2_SHIFT), UART_D_R2T2_SHIFT,
UART_D_R2T2_WIDTH))
/*@}*/

/*
 * @name Register UART_D, field R3T3[3] (RW)
 *
 * Read receive data buffer 3 or write transmit data buffer 3.
 */
/*@{*/
/*! @brief Read current value of the UART_D_R3T3 field. */
#define UART_RD_D_R3T3(base) ((UART_D_REG(base) & UART_D_R3T3_MASK) >>
UART_D_R3T3_SHIFT)
#define UART_BRD_D_R3T3(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R3T3_SHIFT, UART_D_R3T3_WIDTH))

/*! @brief Set the R3T3 field to a new value. */
#define UART_WR_D_R3T3(base, value) (UART_RMW_D(base, UART_D_R3T3_MASK,
UART_D_R3T3(value)))
#define UART_BWR_D_R3T3(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R3T3_SHIFT), UART_D_R3T3_SHIFT,
UART_D_R3T3_WIDTH))
/*@}*/

/*
 * @name Register UART_D, field R4T4[4] (RW)
 *
 * Read receive data buffer 4 or write transmit data buffer 4.
 */
/*@{*/
/*! @brief Read current value of the UART_D_R4T4 field. */
#define UART_RD_D_R4T4(base) ((UART_D_REG(base) & UART_D_R4T4_MASK) >>
UART_D_R4T4_SHIFT)
#define UART_BRD_D_R4T4(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R4T4_SHIFT, UART_D_R4T4_WIDTH))

```

```

/*! @brief Set the R4T4 field to a new value. */
#define UART_WR_D_R4T4(base, value) (UART_RMW_D(base, UART_D_R4T4_MASK,
UART_D_R4T4(value)))
#define UART_BWR_D_R4T4(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R4T4_SHIFT), UART_D_R4T4_SHIFT,
UART_D_R4T4_WIDTH))
/*@}*/

/*!!
 * @name Register UART_D, field R5T5[5] (RW)
 *
 * Read receive data buffer 5 or write transmit data buffer 5.
 */
/*@{*/
/*! @brief Read current value of the UART_D_R5T5 field. */
#define UART_RD_D_R5T5(base) ((UART_D_REG(base) & UART_D_R5T5_MASK) >>
UART_D_R5T5_SHIFT)
#define UART_BRD_D_R5T5(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R5T5_SHIFT, UART_D_R5T5_WIDTH))

/*! @brief Set the R5T5 field to a new value. */
#define UART_WR_D_R5T5(base, value) (UART_RMW_D(base, UART_D_R5T5_MASK,
UART_D_R5T5(value)))
#define UART_BWR_D_R5T5(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R5T5_SHIFT), UART_D_R5T5_SHIFT,
UART_D_R5T5_WIDTH))
/*@}*/

/*!!
 * @name Register UART_D, field R6T6[6] (RW)
 *
 * Read receive data buffer 6 or write transmit data buffer 6.
 */
/*@{*/
/*! @brief Read current value of the UART_D_R6T6 field. */
#define UART_RD_D_R6T6(base) ((UART_D_REG(base) & UART_D_R6T6_MASK) >>
UART_D_R6T6_SHIFT)
#define UART_BRD_D_R6T6(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R6T6_SHIFT, UART_D_R6T6_WIDTH))

/*! @brief Set the R6T6 field to a new value. */
#define UART_WR_D_R6T6(base, value) (UART_RMW_D(base, UART_D_R6T6_MASK,
UART_D_R6T6(value)))
#define UART_BWR_D_R6T6(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R6T6_SHIFT), UART_D_R6T6_SHIFT,
UART_D_R6T6_WIDTH))
/*@}*/

/*!!
 * @name Register UART_D, field R7T7[7] (RW)
 *
 * Read receive data buffer 7 or write transmit data buffer 7.
 */
/*@{*/

```

```

/*! @brief Read current value of the UART_D_R7T7 field. */
#define UART_RD_D_R7T7(base) ((UART_D_REG(base) & UART_D_R7T7_MASK) >>
UART_D_R7T7_SHIFT)
#define UART_BRD_D_R7T7(base) (BME_UBFX8(&UART_D_REG(base),
UART_D_R7T7_SHIFT, UART_D_R7T7_WIDTH))

/*! @brief Set the R7T7 field to a new value. */
#define UART_WR_D_R7T7(base, value) (UART_RMW_D(base, UART_D_R7T7_MASK,
UART_D_R7T7(value)))
#define UART_BWR_D_R7T7(base, value) (BME_BFI8(&UART_D_REG(base),
((uint8_t)(value) << UART_D_R7T7_SHIFT), UART_D_R7T7_SHIFT,
UART_D_R7T7_WIDTH))
/*@}*/

/********************* ****
 * UART_C4 - UART Control Register 4
***** */

/*!!
 * @brief UART_C4 - UART Control Register 4 (RW)
 *
 * Reset value: 0x00U
 */
/*!!
 * @name Constants and macros for entire UART_C4 register
 */
/*@{*/
#define UART_RD_C4(base) (UART_C4_REG(base))
#define UART_WR_C4(base, value) (UART_C4_REG(base) = (value))
#define UART_RMW_C4(base, mask, value) (UART_WR_C4(base,
(UART_RD_C4(base) & ~mask) | (value)))
#define UART_SET_C4(base, value) (BME_OR8(&UART_C4_REG(base),
(uint8_t)(value)))
#define UART_CLR_C4(base, value) (BME_AND8(&UART_C4_REG(base),
(uint8_t)(~(value))))
#define UART_TOG_C4(base, value) (BME_XOR8(&UART_C4_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART_C4 bitfields
 */

/*!!
 * @name Register UART_C4, field RDMAS[5] (RW)
 *
 * RDMAS configures the receiver data register full flag, RDRF, to
 * generate
 * interrupt or DMA requests if RIE is set. If RIE is cleared, the RDRF
 * DMA and RDRF
 * interrupt request signals are not asserted when the RDRF flag is set,

```

```

* regardless of the state of RDMAS.
*
* Values:
* - 0b0 - If RIE is set and the RDRF flag is set, the RDRF interrupt
request
*     signal is asserted to request interrupt service.
* - 0b1 - If RIE is set and the RDRF flag is set, the RDRF DMA request
signal
*     is asserted to request a DMA transfer.
*/
/*@{*/
/*! @brief Read current value of the UART_C4_RDMAS field. */
#define UART_RD_C4_RDMAS(base) ((UART_C4_REG(base) & UART_C4_RDMAS_MASK)
>> UART_C4_RDMAS_SHIFT)
#define UART_BRD_C4_RDMAS(base) (BME_UBFX8(&UART_C4_REG(base),
UART_C4_RDMAS_SHIFT, UART_C4_RDMAS_WIDTH))

/*! @brief Set the RDMAS field to a new value. */
#define UART_WR_C4_RDMAS(base, value) (UART_RMW_C4(base,
UART_C4_RDMAS_MASK, UART_C4_RDMAS(value)))
#define UART_BWR_C4_RDMAS(base, value) (BME_BFI8(&UART_C4_REG(base),
((uint8_t)(value) << UART_C4_RDMAS_SHIFT), UART_C4_RDMAS_SHIFT,
UART_C4_RDMAS_WIDTH))
/*}@*/
```

/\*!

\* @name Register UART\_C4, field TDMAS[7] (RW)

\*

\* TDMAS configures the transmit data register empty flag, TDRE, to generate

\* interrupt or DMA requests if TIE is set. If UART\_C2[TIE] is cleared, TDRE DMA and

\* TDRE interrupt request signals are not asserted when the TDRE flag is set,

\* regardless of the state of TDMAS. If UART\_C2[TIE] and TDMAS are both set, then

\* UART\_C2[TCIE] must be cleared, and UART\_D must not be written outside of

\* servicing of a DMA request.

\*

\* Values:

\* - 0b0 - If TIE is set and the TDRE flag is set, the TDRE interrupt request

\* signal is asserted to request interrupt service.

\* - 0b1 - If TIE is set and the TDRE flag is set, the TDRE DMA request signal

\* is asserted to request a DMA transfer.

/\*
/\*! @brief Read current value of the UART\_C4\_TDMAS field. \*/
#define UART\_RD\_C4\_TDMAS(base) ((UART\_C4\_REG(base) & UART\_C4\_TDMAS\_MASK)
>> UART\_C4\_TDMAS\_SHIFT)
#define UART\_BRD\_C4\_TDMAS(base) (BME\_UBFX8(&UART\_C4\_REG(base),
UART\_C4\_TDMAS\_SHIFT, UART\_C4\_TDMAS\_WIDTH))

```

/*! @brief Set the TDMAS field to a new value. */
#define UART_WR_C4_TDMAS(base, value) (UART_RMW_C4(base,
UART_C4_TDMAS_MASK, UART_C4_TDMAS(value)))
#define UART_BWR_C4_TDMAS(base, value) (BME_BFI8(&UART_C4_REG(base),
((uint8_t)(value) << UART_C4_TDMAS_SHIFT), UART_C4_TDMAS_SHIFT,
UART_C4_TDMAS_WIDTH))
/*@}*/

/*
 * MKL25Z4 UART0
 *
 * Universal Asynchronous Receiver/Transmitter
 *
 * Registers defined in this header file:
 * - UART0_BDH - UART Baud Rate Register High
 * - UART0_BDL - UART Baud Rate Register Low
 * - UART0_C1 - UART Control Register 1
 * - UART0_C2 - UART Control Register 2
 * - UART0_S1 - UART Status Register 1
 * - UART0_S2 - UART Status Register 2
 * - UART0_C3 - UART Control Register 3
 * - UART0_D - UART Data Register
 * - UART0_MA1 - UART Match Address Registers 1
 * - UART0_MA2 - UART Match Address Registers 2
 * - UART0_C4 - UART Control Register 4
 * - UART0_C5 - UART Control Register 5
*/
#define UART0_INSTANCE_COUNT (1U) /*!< Number of instances of the UART0
module. */
#define UART0_IDX (0U) /*!< Instance number for UART0. */

/********************* UART0_BDH - UART Baud Rate Register High *****/
***** /
 * UART0_BDH - UART Baud Rate Register High
***** /

/*!!
 * @brief UART0_BDH - UART Baud Rate Register High (RW)
 *
 * Reset value: 0x00U
 *
 * This register, along with UART_BDL, controls the prescale divisor for
UART
 * baud rate generation. The 13-bit baud rate setting [SBR12:SBR0] should
only be
 * updated when the transmitter and receiver are both disabled.
 */
/*!
 * @name Constants and macros for entire UART0_BDH register
*/

```

```

/*@{*/
#define UART0_RD_BDH(base)          (UART0_BDH_REG(base))
#define UART0_WR_BDH(base, value)   (UART0_BDH_REG(base) = (value))
#define UART0_RMW_BDH(base, mask, value) (UART0_WR_BDH(base,
(UART0_RD_BDH(base) & ~mask) | (value)))
#define UART0_SET_BDH(base, value)  (BME_OR8(&UART0_BDH_REG(base),
(uint8_t)(value)))
#define UART0_CLR_BDH(base, value)  (BME_AND8(&UART0_BDH_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_BDH(base, value)  (BME_XOR8(&UART0_BDH_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART0_BDH bitfields
 */

/*!
 * @name Register UART0_BDH, field SBR[4:0] (RW)
 *
 * The 13 bits in SBR[12:0] are referred to collectively as BR, and they
set the
 * modulo divide rate for the baud rate generator. When BR is 1 - 8191,
the baud
 * rate equals baud clock / ((OSR+1) * BR).
 */
/*@{/*/
/*! @brief Read current value of the UART0_BDH_SBR field. */
#define UART0_RD_BDH_SBR(base) ((UART0_BDH_REG(base) &
UART0_BDH_SBR_MASK) >> UART0_BDH_SBR_SHIFT)
#define UART0_BRD_BDH_SBR(base) (BME_UBFX8(&UART0_BDH_REG(base),
UART0_BDH_SBR_SHIFT, UART0_BDH_SBR_WIDTH))

/*! @brief Set the SBR field to a new value. */
#define UART0_WR_BDH_SBR(base, value) (UART0_RMW_BDH(base,
UART0_BDH_SBR_MASK, UART0_BDH_SBR(value)))
#define UART0_BWR_BDH_SBR(base, value) (BME_BFI8(&UART0_BDH_REG(base),
((uint8_t)(value) << UART0_BDH_SBR_SHIFT), UART0_BDH_SBR_SHIFT,
UART0_BDH_SBR_WIDTH))
/*@{/*/

/*!
 * @name Register UART0_BDH, field SBNS[5] (RW)
 *
 * SBNS determines whether data characters are one or two stop bits. This
bit
 * should only be changed when the transmitter and receiver are both
disabled.
 *
 * Values:
 * - 0b0 - One stop bit.
 * - 0b1 - Two stop bit.
 */
/*@{/*/

```

```

/*! @brief Read current value of the UART0_BDH_SBNS field. */
#define UART0_RD_BDH_SBNS(base) ((UART0_BDH_REG(base) &
UART0_BDH_SBNS_MASK) >> UART0_BDH_SBNS_SHIFT)
#define UART0_BRD_BDH_SBNS(base) (BME_UBFX8(&UART0_BDH_REG(base),
UART0_BDH_SBNS_SHIFT, UART0_BDH_SBNS_WIDTH))

/*! @brief Set the SBNS field to a new value. */
#define UART0_WR_BDH_SBNS(base, value) (UART0_RMW_BDH(base,
UART0_BDH_SBNS_MASK, UART0_BDH_SBNS(value)))
#define UART0_BWR_BDH_SBNS(base, value) (BME_BFI8(&UART0_BDH_REG(base),
((uint8_t)(value) << UART0_BDH_SBNS_SHIFT), UART0_BDH_SBNS_SHIFT,
UART0_BDH_SBNS_WIDTH))
/*@}*/

/*!
* @name Register UART0_BDH, field RXEDGIE[6] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from UART_S2[RXEDGIF] disabled (use
polling).
* - 0b1 - Hardware interrupt requested when UART_S2[RXEDGIF] flag is 1.
*/
/*@{*/
/*! @brief Read current value of the UART0_BDH_RXEDGIE field. */
#define UART0_RD_BDH_RXEDGIE(base) ((UART0_BDH_REG(base) &
UART0_BDH_RXEDGIE_MASK) >> UART0_BDH_RXEDGIE_SHIFT)
#define UART0_BRD_BDH_RXEDGIE(base) (BME_UBFX8(&UART0_BDH_REG(base),
UART0_BDH_RXEDGIE_SHIFT, UART0_BDH_RXEDGIE_WIDTH))

/*! @brief Set the RXEDGIE field to a new value. */
#define UART0_WR_BDH_RXEDGIE(base, value) (UART0_RMW_BDH(base,
UART0_BDH_RXEDGIE_MASK, UART0_BDH_RXEDGIE(value)))
#define UART0_BWR_BDH_RXEDGIE(base, value)
(BME_BFI8(&UART0_BDH_REG(base), ((uint8_t)(value) <<
UART0_BDH_RXEDGIE_SHIFT), UART0_BDH_RXEDGIE_SHIFT,
UART0_BDH_RXEDGIE_WIDTH))
/*@}*/

/*!
* @name Register UART0_BDH, field LBKDIE[7] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from UART_S2[LBKDIF] disabled (use
polling).
* - 0b1 - Hardware interrupt requested when UART_S2[LBKDIF] flag is 1.
*/
/*@{*/
/*! @brief Read current value of the UART0_BDH_LBKDIE field. */
#define UART0_RD_BDH_LBKDIE(base) ((UART0_BDH_REG(base) &
UART0_BDH_LBKDIE_MASK) >> UART0_BDH_LBKDIE_SHIFT)
#define UART0_BRD_BDH_LBKDIE(base) (BME_UBFX8(&UART0_BDH_REG(base),
UART0_BDH_LBKDIE_SHIFT, UART0_BDH_LBKDIE_WIDTH))

/*! @brief Set the LBKDIE field to a new value. */

```

```

#define UART0_WR_BDH_LBKDIE(base, value) (UART0_RMW_BDH(base,
UART0_BDH_LBKDIE_MASK, UART0_BDH_LBKDIE(value)))
#define UART0_WR_BDH_LBKDIE(base, value) (BME_BFI8(&UART0_BDH_REG(base),
((uint8_t)(value)) << UART0_BDH_LBKDIE_SHIFT), UART0_BDH_LBKDIE_SHIFT,
UART0_BDH_LBKDIE_WIDTH))
/*@}*/

/********************* ****
 * UART0_BDL - UART Baud Rate Register Low
***** */

/*!!
* @brief UART0_BDL - UART Baud Rate Register Low (RW)
*
* Reset value: 0x04U
*
* This register, along with UART_BDH, control the prescale divisor for
UART
* baud rate generation. The 13-bit baud rate setting [SBR12:SBR0] can
only be
* updated when the transmitter and receiver are both disabled. UART_BDL
is reset to
* a non-zero value, so after reset the baud rate generator remains
disabled
* until the first time the receiver or transmitter is enabled; that is,
UART
* _C2[RE] or UART_C2[TE] bits are written to 1.
*/
/*!!
* @name Constants and macros for entire UART0_BDL register
*/
/*@{*/
#define UART0_RD_BDL(base) (UART0_BDL_REG(base))
#define UART0_WR_BDL(base, value) (UART0_BDL_REG(base) = (value))
#define UART0_RMW_BDL(base, mask, value) (UART0_WR_BDL(base,
(UART0_RD_BDL(base) & ~mask) | (value)))
#define UART0_SET_BDL(base, value) (BME_OR8(&UART0_BDL_REG(base),
(uint8_t)(value)))
#define UART0_CLR_BDL(base, value) (BME_AND8(&UART0_BDL_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_BDL(base, value) (BME_XOR8(&UART0_BDL_REG(base),
(uint8_t)(value)))
/*@}*/

/********************* ****
 * UART0_C1 - UART Control Register 1
***** */

***** */

```

```

/*!
 * @brief UART0_C1 - UART Control Register 1 (RW)
 *
 * Reset value: 0x00U
 *
 * This read/write register controls various optional features of the
UART
 * system. This register should only be altered when the transmitter and
receiver are
 * both disabled.
 */
/*!
 * @name Constants and macros for entire UART0_C1 register
 */
/*@{*/
#define UART0_RD_C1(base)          (UART0_C1_REG(base))
#define UART0_WR_C1(base, value)   (UART0_C1_REG(base) = (value))
#define UART0_RMW_C1(base, mask, value) (UART0_WR_C1(base,
(UART0_RD_C1(base) & ~mask) | (value)))
#define UART0_SET_C1(base, value)   (BME_OR8(&UART0_C1_REG(base),
(uint8_t)(value)))
#define UART0_CLR_C1(base, value)   (BME_AND8(&UART0_C1_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_C1(base, value)   (BME_XOR8(&UART0_C1_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual UART0_C1 bitfields
 */
/*!
 * @name Register UART0_C1, field PT[0] (RW)
 *
 * Provided parity is enabled (PE = 1), this bit selects even or odd
parity. Odd
 * parity means the total number of 1s in the data character, including
the
 * parity bit, is odd. Even parity means the total number of 1s in the
data
 * character, including the parity bit, is even.
 *
 * Values:
 * - 0b0 - Even parity.
 * - 0b1 - Odd parity.
 */
/*@*/
/*! @brief Read current value of the UART0_C1_PT field. */
#define UART0_RD_C1_PT(base) ((UART0_C1_REG(base) & UART0_C1_PT_MASK) >>
UART0_C1_PT_SHIFT)
#define UART0_BRD_C1_PT(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_PT_SHIFT, UART0_C1_PT_WIDTH))

/*! @brief Set the PT field to a new value. */

```

```

#define UART0_WR_C1_PT(base, value)  (UART0_RMW_C1(base, UART0_C1_PT_MASK, \
UART0_C1_PT(value)))
#define UART0_BWR_C1_PT(base, value)  (BME_BFI8(&UART0_C1_REG(base), \
((uint8_t)(value) << UART0_C1_PT_SHIFT), UART0_C1_PT_SHIFT, \
UART0_C1_PT_WIDTH))
/*@}*/

/*
 * @name Register UART0_C1, field PE[1] (RW)
 *
 * Enables hardware parity generation and checking. When parity is
enabled, the
 * bit immediately before the stop bit is treated as the parity bit.
 *
 * Values:
 * - 0b0 - No hardware parity generation or checking.
 * - 0b1 - Parity enabled.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_PE field. */
#define UART0_RD_C1_PE(base)  ((UART0_C1_REG(base) & UART0_C1_PE_MASK) >>
UART0_C1_PE_SHIFT)
#define UART0_BRD_C1_PE(base)  (BME_UBFX8(&UART0_C1_REG(base), \
UART0_C1_PE_SHIFT, UART0_C1_PE_WIDTH))

/*! @brief Set the PE field to a new value. */
#define UART0_WR_C1_PE(base, value)  (UART0_RMW_C1(base, UART0_C1_PE_MASK, \
UART0_C1_PE(value)))
#define UART0_BWR_C1_PE(base, value)  (BME_BFI8(&UART0_C1_REG(base), \
((uint8_t)(value) << UART0_C1_PE_SHIFT), UART0_C1_PE_SHIFT, \
UART0_C1_PE_WIDTH))
/*@}*/

/*
 * @name Register UART0_C1, field ILT[2] (RW)
 *
 * Setting this bit to 1 ensures that the stop bits and logic 1 bits at
the end
 * of a character do not count toward the 10 to 13 bit times of logic
high level
 * needed by the idle line detection logic.
 *
 * Values:
 * - 0b0 - Idle character bit count starts after start bit.
 * - 0b1 - Idle character bit count starts after stop bit.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_ILT field. */
#define UART0_RD_C1_ILT(base)  ((UART0_C1_REG(base) & UART0_C1_ILT_MASK) \
>> UART0_C1_ILT_SHIFT)
#define UART0_BRD_C1_ILT(base)  (BME_UBFX8(&UART0_C1_REG(base), \
UART0_C1_ILT_SHIFT, UART0_C1_ILT_WIDTH))

/*! @brief Set the ILT field to a new value. */

```

```

#define UART0_WR_C1_ILT(base, value) (UART0_RMW_C1(base,
UART0_C1_ILT_MASK, UART0_C1_ILT(value)))
#define UART0_BWR_C1_ILT(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_ILT_SHIFT), UART0_C1_ILT_SHIFT,
UART0_C1_ILT_WIDTH))
/*@}*/

/*!!
 * @name Register UART0_C1, field WAKE[3] (RW)
 *
 * Values:
 * - 0b0 - Idle-line wakeup.
 * - 0b1 - Address-mark wakeup.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_WAKE field. */
#define UART0_RD_C1_WAKE(base) ((UART0_C1_REG(base) & UART0_C1_WAKE_MASK) >> UART0_C1_WAKE_SHIFT)
#define UART0_BRD_C1_WAKE(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_WAKE_SHIFT, UART0_C1_WAKE_WIDTH))

/*! @brief Set the WAKE field to a new value. */
#define UART0_WR_C1_WAKE(base, value) (UART0_RMW_C1(base,
UART0_C1_WAKE_MASK, UART0_C1_WAKE(value)))
#define UART0_BWR_C1_WAKE(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_WAKE_SHIFT), UART0_C1_WAKE_SHIFT,
UART0_C1_WAKE_WIDTH))
/*@}*/

/*!!
 * @name Register UART0_C1, field M[4] (RW)
 *
 * Values:
 * - 0b0 - Receiver and transmitter use 8-bit data characters.
 * - 0b1 - Receiver and transmitter use 9-bit data characters.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_M field. */
#define UART0_RD_C1_M(base) ((UART0_C1_REG(base) & UART0_C1_M_MASK) >> UART0_C1_M_SHIFT)
#define UART0_BRD_C1_M(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_M_SHIFT, UART0_C1_M_WIDTH))

/*! @brief Set the M field to a new value. */
#define UART0_WR_C1_M(base, value) (UART0_RMW_C1(base, UART0_C1_M_MASK,
UART0_C1_M(value)))
#define UART0_BWR_C1_M(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_M_SHIFT), UART0_C1_M_SHIFT,
UART0_C1_M_WIDTH))
/*@}*/

/*!!
 * @name Register UART0_C1, field RSRC[5] (RW)
 *

```

```

 * This bit has no meaning or effect unless the LOOPS bit is set to 1.
When
 * LOOPS is set, the receiver input is internally connected to the UART
 _TX pin and
 * RSRC determines whether this connection is also connected to the
 transmitter
 * output.
 *
 * Values:
 * - 0b0 - Provided LOOPS is set, RSRC is cleared, selects internal loop
 back
 * mode and the UART does not use the UART_RX pins.
 * - 0b1 - Single-wire UART mode where the UART_TX pin is connected to
 the
 * transmitter output and receiver input.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_RSRC field. */
#define UART0_RD_C1_RSRC(base) ((UART0_C1_REG(base) & UART0_C1_RSRC_MASK)
>> UART0_C1_RSRC_SHIFT)
#define UART0_BRD_C1_RSRC(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_RSRC_SHIFT, UART0_C1_RSRC_WIDTH))

/*! @brief Set the RSRC field to a new value. */
#define UART0_WR_C1_RSRC(base, value) (UART0_RMW_C1(base,
UART0_C1_RSRC_MASK, UART0_C1_RSRC(value)))
#define UART0_BWR_C1_RSRC(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_RSRC_SHIFT), UART0_C1_RSRC_SHIFT,
UART0_C1_RSRC_WIDTH))
/*@}*/
/*!
 * @name Register UART0_C1, field DOZEEN[6] (RW)
 *
 * Values:
 * - 0b0 - UART is enabled in Wait mode.
 * - 0b1 - UART is disabled in Wait mode.
 */
/*@{*/
/*! @brief Read current value of the UART0_C1_DOZEEN field. */
#define UART0_RD_C1_DOZEEN(base) ((UART0_C1_REG(base) &
UART0_C1_DOZEEN_MASK) >> UART0_C1_DOZEEN_SHIFT)
#define UART0_BRD_C1_DOZEEN(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_DOZEEN_SHIFT, UART0_C1_DOZEEN_WIDTH))

/*! @brief Set the DOZEEN field to a new value. */
#define UART0_WR_C1_DOZEEN(base, value) (UART0_RMW_C1(base,
UART0_C1_DOZEEN_MASK, UART0_C1_DOZEEN(value)))
#define UART0_BWR_C1_DOZEEN(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_DOZEEN_SHIFT), UART0_C1_DOZEEN_SHIFT,
UART0_C1_DOZEEN_WIDTH))
/*@}*/
/*!

```

```

* @name Register UART0_C1, field LOOPS[7] (RW)
*
* Selects between loop back modes and normal 2-pin full-duplex modes.
When
* LOOPS is set, the transmitter output is internally connected to the
receiver input.
*
* Values:
* - 0b0 - Normal operation - UART_RX and UART_TX use separate pins.
* - 0b1 - Loop mode or single-wire mode where transmitter outputs are
* internally connected to receiver input. (See RSRC bit.) UART_RX
pin is not used by
*     UART .
*/
/*@{*/
/*! @brief Read current value of the UART0_C1_LOOPS field. */
#define UART0_RD_C1_LOOPS(base) ((UART0_C1_REG(base) &
UART0_C1_LOOPS_MASK) >> UART0_C1_LOOPS_SHIFT)
#define UART0_BRD_C1_LOOPS(base) (BME_UBFX8(&UART0_C1_REG(base),
UART0_C1_LOOPS_SHIFT, UART0_C1_LOOPS_WIDTH))

/*! @brief Set the LOOPS field to a new value. */
#define UART0_WR_C1_LOOPS(base, value) (UART0_RMW_C1(base,
UART0_C1_LOOPS_MASK, UART0_C1_LOOPS(value)))
#define UART0_BWR_C1_LOOPS(base, value) (BME_BFI8(&UART0_C1_REG(base),
((uint8_t)(value) << UART0_C1_LOOPS_SHIFT), UART0_C1_LOOPS_SHIFT,
UART0_C1_LOOPS_WIDTH))
/*@}*/

/*****
*****
* UART0_C2 - UART Control Register 2
*****
****/

/*!
* @brief UART0_C2 - UART Control Register 2 (RW)
*
* Reset value: 0x00U
*
* This register can be read or written at any time.
*/
/*!
* @name Constants and macros for entire UART0_C2 register
*/
/*@{*/
#define UART0_RD_C2(base) (UART0_C2_REG(base))
#define UART0_WR_C2(base, value) (UART0_C2_REG(base) = (value))
#define UART0_RMW_C2(base, mask, value) (UART0_WR_C2(base,
(UART0_RD_C2(base) & ~mask) | (value)))
#define UART0_SET_C2(base, value) (BME_OR8(&UART0_C2_REG(base),
(uint8_t)(value)))

```

```

#define UART0_CLR_C2(base, value) (BME_AND8 (&UART0_C2_REG(base),  

(uint8_t) (~(value))))  

#define UART0_TOG_C2(base, value) (BME_XOR8 (&UART0_C2_REG(base),  

(uint8_t) (value)))  

/*@*/
```

/\*
 \* Constants & macros for individual UART0\_C2 bitfields
 \*/

```

/*!  

 * @name Register UART0_C2, field SBK[0] (RW)  

 *  

 * Writing a 1 and then a 0 to SBK queues a break character in the  

transmit data  

 * stream. Additional break characters of 10 to 13, or 13 to 16 if BRK13  

= 1,  

 * bit times of logic 0 are queued as long as SBK is set. Depending on  

the timing  

 * of the set and clear of SBK relative to the information currently  

being  

 * transmitted, a second break character may be queued before software  

clears SBK.  

 *  

 * Values:  

 * - 0b0 - Normal transmitter operation.  

 * - 0b1 - Queue break character(s) to be sent.  

 */
/*@{*/
```

```

/*! @brief Read current value of the UART0_C2_SBK field. */  

#define UART0_RD_C2_SBK(base) ((UART0_C2_REG(base) & UART0_C2_SBK_MASK)  

>> UART0_C2_SBK_SHIFT)  

#define UART0_BRD_C2_SBK(base) (BME_UBFX8 (&UART0_C2_REG(base),  

UART0_C2_SBK_SHIFT, UART0_C2_SBK_WIDTH))
```

```

/*! @brief Set the SBK field to a new value. */  

#define UART0_WR_C2_SBK(base, value) (UART0_RMW_C2(base,  

UART0_C2_SBK_MASK, UART0_C2_SBK(value)))  

#define UART0_BWR_C2_SBK(base, value) (BME_BFI8 (&UART0_C2_REG(base),  

((uint8_t) (value) << UART0_C2_SBK_SHIFT), UART0_C2_SBK_SHIFT,  

UART0_C2_SBK_WIDTH))  

/*@*/
```

/\*
 \* @name Register UART0\_C2, field RWU[1] (RW)
 \*  
 \* This bit can be written to 1 to place the UART receiver in a standby  
state  
 \* where it waits for automatic hardware detection of a selected wakeup  
condition.  
 \* The wakeup condition is an idle line between messages, WAKE = 0, idle-  
line  
 \* wakeup, or a logic 1 in the most significant data bit in a character,  
WAKE = 1,

```

    * address-mark wakeup. Application software sets RWU and, normally, a
selected
    * hardware condition automatically clears RWU.
    *
    * Values:
    * - 0b0 - Normal UART receiver operation.
    * - 0b1 - UART receiver in standby waiting for wakeup condition.
    */
/*@{/*/
/*! @brief Read current value of the UART0_C2_RWU field. */
#define UART0_RD_C2_RWU(base) ((UART0_C2_REG(base) & UART0_C2_RWU_MASK)
>> UART0_C2_RWU_SHIFT)
#define UART0_BRD_C2_RWU(base) (BME_UBFX8(&UART0_C2_REG(base),
UART0_C2_RWU_SHIFT, UART0_C2_RWU_WIDTH))

/*! @brief Set the RWU field to a new value. */
#define UART0_WR_C2_RWU(base, value) (UART0_RMW_C2(base,
UART0_C2_RWU_MASK, UART0_C2_RWU(value)))
#define UART0_BWR_C2_RWU(base, value) (BME_BFI8(&UART0_C2_REG(base),
(uint8_t)(value) << UART0_C2_RWU_SHIFT), UART0_C2_RWU_SHIFT,
UART0_C2_RWU_WIDTH)
/*@{/*/

/*!
    * @name Register UART0_C2, field RE[2] (RW)
    *
    * When the UART receiver is off or LOOPS is set, the UART_RX pin is not
used
    * by the UART . When RE is written to 0, the receiver finishes receiving
the
    * current character (if any).
    *
    * Values:
    * - 0b0 - Receiver disabled.
    * - 0b1 - Receiver enabled.
    */
/*@{/*/
/*! @brief Read current value of the UART0_C2_RE field. */
#define UART0_RD_C2_RE(base) ((UART0_C2_REG(base) & UART0_C2_RE_MASK) >>
UART0_C2_RE_SHIFT)
#define UART0_BRD_C2_RE(base) (BME_UBFX8(&UART0_C2_REG(base),
UART0_C2_RE_SHIFT, UART0_C2_RE_WIDTH))

/*! @brief Set the RE field to a new value. */
#define UART0_WR_C2_RE(base, value) (UART0_RMW_C2(base, UART0_C2_RE_MASK,
UART0_C2_RE(value)))
#define UART0_BWR_C2_RE(base, value) (BME_BFI8(&UART0_C2_REG(base),
(uint8_t)(value) << UART0_C2_RE_SHIFT), UART0_C2_RE_SHIFT,
UART0_C2_RE_WIDTH)
/*@{/*/

/*!
    * @name Register UART0_C2, field TE[3] (RW)
    *

```

```

 * TE must be 1 to use the UART transmitter. When TE is set, the UART
forces the
 * UART _TX pin to act as an output for the UART system. When the UART is
 * configured for single-wire operation (LOOPS = RSRC = 1), TXDIR
controls the
 * direction of traffic on the single UART communication line ( UART _TX
pin). TE can
 * also queue an idle character by clearing TE then setting TE while a
transmission
 * is in progress. When TE is written to 0, the transmitter keeps control
of the
 * port UART _TX pin until any data, queued idle, or queued break
character
 * finishes transmitting before allowing the pin to tristate.
*
* Values:
* - 0b0 - Transmitter disabled.
* - 0b1 - Transmitter enabled.
*/
/*@{*/
/*! @brief Read current value of the UART0_C2_TE field. */
#define UART0_RD_C2_TE(base) ((UART0_C2_REG(base) & UART0_C2_TE_MASK) >>
UART0_C2_TE_SHIFT)
#define UART0_BRD_C2_TE(base) (BME_UBFX8(&UART0_C2_REG(base),
UART0_C2_TE_SHIFT, UART0_C2_TE_WIDTH))

/*! @brief Set the TE field to a new value. */
#define UART0_WR_C2_TE(base, value) (UART0_RMW_C2(base, UART0_C2_TE_MASK,
UART0_C2_TE(value)))
#define UART0_BWR_C2_TE(base, value) (BME_BFI8(&UART0_C2_REG(base),
(uint8_t)(value) << UART0_C2_TE_SHIFT), UART0_C2_TE_SHIFT,
UART0_C2_TE_WIDTH)
/*@}*/

/*
 * @name Register UART0_C2, field ILIE[4] (RW)
*
* Values:
* - 0b0 - Hardware interrupts from IDLE disabled; use polling.
* - 0b1 - Hardware interrupt requested when IDLE flag is 1.
*/
/*@{*/
/*! @brief Read current value of the UART0_C2_ILIE field. */
#define UART0_RD_C2_ILIE(base) ((UART0_C2_REG(base) & UART0_C2_ILIE_MASK)
>> UART0_C2_ILIE_SHIFT)
#define UART0_BRD_C2_ILIE(base) (BME_UBFX8(&UART0_C2_REG(base),
UART0_C2_ILIE_SHIFT, UART0_C2_ILIE_WIDTH))

/*! @brief Set the ILIE field to a new value. */
#define UART0_WR_C2_ILIE(base, value) (UART0_RMW_C2(base,
UART0_C2_ILIE_MASK, UART0_C2_ILIE(value)))
#define UART0_BWR_C2_ILIE(base, value) (BME_BFI8(&UART0_C2_REG(base),
(uint8_t)(value) << UART0_C2_ILIE_SHIFT), UART0_C2_ILIE_SHIFT,
UART0_C2_ILIE_WIDTH))

```

```

/*@}*/

/*
 * @name Register UART0_C2, field RIE[5] (RW)
 *
 * Values:
 * - 0b0 - Hardware interrupts from RDRF disabled; use polling.
 * - 0b1 - Hardware interrupt requested when RDRF flag is 1.
 */
/*@{*/
/*! @brief Read current value of the UART0_C2_RIE field. */
#define UART0_RD_C2_RIE(base) ((UART0_C2_REG(base) & UART0_C2_RIE_MASK) \
>> UART0_C2_RIE_SHIFT)
#define UART0_BRD_C2_RIE(base) (BME_UBFX8(&UART0_C2_REG(base), \
UART0_C2_RIE_SHIFT, UART0_C2_RIE_WIDTH))

/*! @brief Set the RIE field to a new value. */
#define UART0_WR_C2_RIE(base, value) (UART0_RMW_C2(base, \
UART0_C2_RIE_MASK, UART0_C2_RIE(value)))
#define UART0_BWR_C2_RIE(base, value) (BME_BFI8(&UART0_C2_REG(base), \
((uint8_t)(value) << UART0_C2_RIE_SHIFT), UART0_C2_RIE_SHIFT, \
UART0_C2_RIE_WIDTH))
/*@}*/
/*!

 * @name Register UART0_C2, field TCIE[6] (RW)
 *
 * Values:
 * - 0b0 - Hardware interrupts from TC disabled; use polling.
 * - 0b1 - Hardware interrupt requested when TC flag is 1.
 */
/*@{*/
/*! @brief Read current value of the UART0_C2_TCIE field. */
#define UART0_RD_C2_TCIE(base) ((UART0_C2_REG(base) & UART0_C2_TCIE_MASK) \
>> UART0_C2_TCIE_SHIFT)
#define UART0_BRD_C2_TCIE(base) (BME_UBFX8(&UART0_C2_REG(base), \
UART0_C2_TCIE_SHIFT, UART0_C2_TCIE_WIDTH))

/*! @brief Set the TCIE field to a new value. */
#define UART0_WR_C2_TCIE(base, value) (UART0_RMW_C2(base, \
UART0_C2_TCIE_MASK, UART0_C2_TCIE(value)))
#define UART0_BWR_C2_TCIE(base, value) (BME_BFI8(&UART0_C2_REG(base), \
((uint8_t)(value) << UART0_C2_TCIE_SHIFT), UART0_C2_TCIE_SHIFT, \
UART0_C2_TCIE_WIDTH))
/*@}*/
/*!

 * @name Register UART0_C2, field TIE[7] (RW)
 *
 * Values:
 * - 0b0 - Hardware interrupts from TDRE disabled; use polling.
 * - 0b1 - Hardware interrupt requested when TDRE flag is 1.
 */
/*@}*/

```

```

/*! @brief Read current value of the UART0_C2_TIE field. */
#define UART0_RD_C2_TIE(base) ((UART0_C2_REG(base) & UART0_C2_TIE_MASK)
>> UART0_C2_TIE_SHIFT)
#define UART0_BRD_C2_TIE(base) (BME_UBFX8(&UART0_C2_REG(base),
UART0_C2_TIE_SHIFT, UART0_C2_TIE_WIDTH))

/*! @brief Set the TIE field to a new value. */
#define UART0_WR_C2_TIE(base, value) (UART0_RMW_C2(base,
UART0_C2_TIE_MASK, UART0_C2_TIE(value)))
#define UART0_BWR_C2_TIE(base, value) (BME_BFI8(&UART0_C2_REG(base),
((uint8_t)(value) << UART0_C2_TIE_SHIFT), UART0_C2_TIE_SHIFT,
UART0_C2_TIE_WIDTH))
/*@}*/

/********************* ****
 * UART0_S1 - UART Status Register 1
***** */

/*!!
 * @brief UART0_S1 - UART Status Register 1 (RW)
 *
 * Reset value: 0xC0U
 */
/*!!
 * @name Constants and macros for entire UART0_S1 register
 */
/*@{*/
#define UART0_RD_S1(base) (UART0_S1_REG(base))
#define UART0_WR_S1(base, value) (UART0_S1_REG(base) = (value))
#define UART0_RMW_S1(base, mask, value) (UART0_WR_S1(base,
(UART0_RD_S1(base) & ~mask) | (value)))
#define UART0_SET_S1(base, value) (BME_OR8(&UART0_S1_REG(base),
(uint8_t)(value)))
#define UART0_CLR_S1(base, value) (BME_AND8(&UART0_S1_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_S1(base, value) (BME_XOR8(&UART0_S1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART0_S1 bitfields
 */
/*!!
 * @name Register UART0_S1, field PF[0] (W1C)
 *
 * PF is set at the same time as RDRE when parity is enabled (PE = 1) and
 * the
 * parity bit in the received character does not agree with the expected
 * parity
 * value. To clear PF, write a logic one to the PF.

```

```

/*
 * Values:
 * - 0b0 - No parity error.
 * - 0b1 - Parity error.
 */
/*@{*/
/*! @brief Read current value of the UART0_S1_PF field. */
#define UART0_RD_S1_PF(base) ((UART0_S1_REG(base) & UART0_S1_PF_MASK) >>
UART0_S1_PF_SHIFT)
#define UART0_BRD_S1_PF(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_PF_SHIFT, UART0_S1_PF_WIDTH))

/*! @brief Set the PF field to a new value. */
#define UART0_WR_S1_PF(base, value) (UART0_RMW_S1(base, (UART0_S1_PF_MASK
| UART0_S1_FE_MASK | UART0_S1_NF_MASK | UART0_S1_OR_MASK |
UART0_S1_IDLE_MASK), UART0_S1_PF(value)))
#define UART0_BWR_S1_PF(base, value) (BME_BFI8(&UART0_S1_REG(base),
((uint8_t)(value) << UART0_S1_PF_SHIFT), UART0_S1_PF_SHIFT,
UART0_S1_PF_WIDTH))
/*@}*/
/*!
 * @name Register UART0_S1, field FE[1] (W1C)
 *
 * FE is set at the same time as RDRF when the receiver detects a logic 0
where
 * a stop bit was expected. This suggests the receiver was not properly
aligned
 * to a character frame. To clear FE, write a logic one to the FE flag.
 *
 * Values:
 * - 0b0 - No framing error detected. This does not guarantee the framing
is
 *     correct.
 * - 0b1 - Framing error.
 */
/*@*/
/*! @brief Read current value of the UART0_S1_FE field. */
#define UART0_RD_S1_FE(base) ((UART0_S1_REG(base) & UART0_S1_FE_MASK) >>
UART0_S1_FE_SHIFT)
#define UART0_BRD_S1_FE(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_FE_SHIFT, UART0_S1_FE_WIDTH))

/*! @brief Set the FE field to a new value. */
#define UART0_WR_S1_FE(base, value) (UART0_RMW_S1(base, (UART0_S1_FE_MASK
| UART0_S1_PF_MASK | UART0_S1_NF_MASK | UART0_S1_OR_MASK |
UART0_S1_IDLE_MASK), UART0_S1_FE(value)))
#define UART0_BWR_S1_FE(base, value) (BME_BFI8(&UART0_S1_REG(base),
((uint8_t)(value) << UART0_S1_FE_SHIFT), UART0_S1_FE_SHIFT,
UART0_S1_FE_WIDTH))
/*@}*/

/*!
 * @name Register UART0_S1, field NF[2] (W1C)

```

```

/*
 * The advanced sampling technique used in the receiver takes three
samples in
 * each of the received bits. If any of these samples disagrees with the
rest of
 * the samples within any bit time in the frame, the flag NF is set at
the same
 * time as RDRF is set for the character. To clear NF, write logic one to
the NF.
 *
 * Values:
 * - 0b0 - No noise detected.
 * - 0b1 - Noise detected in the received character in UART _D.
 */
/*@{*/
/*! @brief Read current value of the UART0_S1_NF field. */
#define UART0_RD_S1_NF(base) ((UART0_S1_REG(base) & UART0_S1_NF_MASK) >>
UART0_S1_NF_SHIFT)
#define UART0_BRD_S1_NF(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_NF_SHIFT, UART0_S1_NF_WIDTH))

/*! @brief Set the NF field to a new value. */
#define UART0_WR_S1_NF(base, value) (UART0_RMW_S1(base, (UART0_S1_NF_MASK
| UART0_S1_PF_MASK | UART0_S1_FE_MASK | UART0_S1_OR_MASK |
UART0_S1_IDLE_MASK), UART0_S1_NF(value)))
#define UART0_BWR_S1_NF(base, value) (BME_BFI8(&UART0_S1_REG(base),
((uint8_t)(value) << UART0_S1_NF_SHIFT), UART0_S1_NF_SHIFT,
UART0_S1_NF_WIDTH))
/*@}*/
/*!
 * @name Register UART0_S1, field OR[3] (W1C)
 *
 * OR is set when a new serial character is ready to be transferred to
the
 * receive data buffer, but the previously received character has not
been read from
 * UART _D yet. In this case, the new character, and all associated error
 * information, is lost because there is no room to move it into UART _D.
To clear OR,
 * write a logic 1 to the OR flag.
 *
 * Values:
 * - 0b0 - No overrun.
 * - 0b1 - Receive overrun (new UART data lost).
 */
/*@{*/
/*! @brief Read current value of the UART0_S1_OR field. */
#define UART0_RD_S1_OR(base) ((UART0_S1_REG(base) & UART0_S1_OR_MASK) >>
UART0_S1_OR_SHIFT)
#define UART0_BRD_S1_OR(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_OR_SHIFT, UART0_S1_OR_WIDTH))

/*! @brief Set the OR field to a new value. */

```

```

#define UART0_WR_S1_OR(base, value) (UART0_RMW_S1(base, (UART0_S1_OR_MASK
| UART0_S1_PF_MASK | UART0_S1_FE_MASK | UART0_S1_NF_MASK |
UART0_S1_IDLE_MASK), UART0_S1_OR(value)))
#define UART0_BWR_S1_OR(base, value) (BME_BFI8(&UART0_S1_REG(base),
((uint8_t)(value) << UART0_S1_OR_SHIFT), UART0_S1_OR_SHIFT,
UART0_S1_OR_WIDTH))
/*@}*/

/*
 * @name Register UART0_S1, field IDLE[4] (W1C)
 *
 * IDLE is set when the UART receive line becomes idle for a full
character time
 * after a period of activity. When ILT is cleared, the receiver starts
counting
 * idle bit times after the start bit. If the receive character is all
1s, these
 * bit times and the stop bits time count toward the full character time
of
 * logic high, 10 to 13 bit times, needed for the receiver to detect an
idle line.
 * When ILT is set, the receiver doesn't start counting idle bit times
until after
 * the stop bits. The stop bits and any logic high bit times at the end
of the
 * previous character do not count toward the full character time of
logic high
 * needed for the receiver to detect an idle line. To clear IDLE, write
logic 1 to
 * the IDLE flag. After IDLE has been cleared, it cannot become set again
until
 * after a new character has been received and RDRF has been set. IDLE is
set only
 * once even if the receive line remains idle for an extended period.
*
* Values:
* - 0b0 - No idle line detected.
* - 0b1 - Idle line was detected.
*/
/*@{*/
/*! @brief Read current value of the UART0_S1_IDLE field. */
#define UART0_RD_S1_IDLE(base) ((UART0_S1_REG(base) & UART0_S1_IDLE_MASK)
>> UART0_S1_IDLE_SHIFT)
#define UART0_BRD_S1_IDLE(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_IDLE_SHIFT, UART0_S1_IDLE_WIDTH))

/*! @brief Set the IDLE field to a new value. */
#define UART0_WR_S1_IDLE(base, value) (UART0_RMW_S1(base,
(UART0_S1_IDLE_MASK | UART0_S1_PF_MASK | UART0_S1_FE_MASK |
UART0_S1_NF_MASK | UART0_S1_OR_MASK), UART0_S1_IDLE(value)))
#define UART0_BWR_S1_IDLE(base, value) (BME_BFI8(&UART0_S1_REG(base),
((uint8_t)(value) << UART0_S1_IDLE_SHIFT), UART0_S1_IDLE_SHIFT,
UART0_S1_IDLE_WIDTH))
/*@}*/

```

```

/*!
 * @name Register UART0_S1, field RDRF[5] (RO)
 *
 * RDRF becomes set whenever the receive data buffer is full. To clear
RDRF,
 * read the UART data register ( UART _D).
 *
 * Values:
 * - 0b0 - Receive data buffer empty.
 * - 0b1 - Receive data buffer full.
 */
/*@{*/
/*! @brief Read current value of the UART0_S1_RDRF field. */
#define UART0_RD_S1_RDRF(base) ((UART0_S1_REG(base) & UART0_S1_RDRF_MASK) >> UART0_S1_RDRF_SHIFT)
#define UART0_BRD_S1_RDRF(base) (BME_UBX8(&UART0_S1_REG(base), UART0_S1_RDRF_SHIFT, UART0_S1_RDRF_WIDTH))
/*}@*/
```

```

/*!
 * @name Register UART0_S1, field TC[6] (RO)
 *
 * TC is set out of reset and when TDRE is set and no data, preamble, or
break
 * character is being transmitted. TC is cleared automatically by one of
the
 * following: Write to the UART data register ( UART _D) to transmit new
data Queue a
 * preamble by changing TE from 0 to 1 Queue a break character by writing
1 to
 * UART _C2[SBK]
 *
 * Values:
 * - 0b0 - Transmitter active (sending data, a preamble, or a break).
 * - 0b1 - Transmitter idle (transmission activity complete).
 */
/*@{*/
/*! @brief Read current value of the UART0_S1_TC field. */
#define UART0_RD_S1_TC(base) ((UART0_S1_REG(base) & UART0_S1_TC_MASK) >> UART0_S1_TC_SHIFT)
#define UART0_BRD_S1_TC(base) (BME_UBX8(&UART0_S1_REG(base), UART0_S1_TC_SHIFT, UART0_S1_TC_WIDTH))
/*}@*/
```

```

/*!
 * @name Register UART0_S1, field TDRE[7] (RO)
 *
 * TDRE is set out of reset and whenever there is room to write data to
the
 * transmit data buffer. To clear TDRE, write to the UART data register ( UART _D).
 *
 * Values:
```

```

* - 0b0 - Transmit data buffer full.
* - 0b1 - Transmit data buffer empty.
*/
/*@{*/
/*! @brief Read current value of the UART0_S1_TDRE field. */
#define UART0_RD_S1_TDRE(base) ((UART0_S1_REG(base) & UART0_S1_TDRE_MASK)
>> UART0_S1_TDRE_SHIFT)
#define UART0_BRD_S1_TDRE(base) (BME_UBFX8(&UART0_S1_REG(base),
UART0_S1_TDRE_SHIFT, UART0_S1_TDRE_WIDTH))
/*}@*/
```

\*\*\*\*\*  
\*\*\*\*\*

```

* UART0_S2 - UART Status Register 2
```

\*\*\*\*\*  
\*\*\*\*\*

```

/*!
* @brief UART0_S2 - UART Status Register 2 (RW)
*
* Reset value: 0x00U
*
* This register contains one read-only status flag. When using an
internal
* oscillator in a LIN system, it is necessary to raise the break
detection threshold
* one bit time. Under the worst case timing conditions allowed in LIN,
it is
* possible that a 0x00 data character can appear to be 10.26 bit times
long at a
* slave running 14% faster than the master. This would trigger normal
break
* detection circuitry designed to detect a 10-bit break symbol. When the
LBKDE bit is
* set, framing errors are inhibited and the break detection threshold
* increases, preventing false detection of a 0x00 data character as a
LIN break symbol.
*/
/*!
* @name Constants and macros for entire UART0_S2 register
*/
/*@{*/
#define UART0_RD_S2(base) (UART0_S2_REG(base))
#define UART0_WR_S2(base, value) (UART0_S2_REG(base) = (value))
#define UART0_RMW_S2(base, mask, value) (UART0_WR_S2(base,
(UART0_RD_S2(base) & ~mask) | (value)))
#define UART0_SET_S2(base, value) (BME_OR8(&UART0_S2_REG(base),
(uint8_t)(value)))
#define UART0_CLR_S2(base, value) (BME_AND8(&UART0_S2_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_S2(base, value) (BME_XOR8(&UART0_S2_REG(base),
(uint8_t)(value)))
/*}@*/
```

```

/*
 * Constants & macros for individual UART0_S2 bitfields
 */

/*!
 * @name Register UART0_S2, field RAF[0] (RO)
 *
 * RAF is set when the UART receiver detects the beginning of a valid
start bit,
 * and RAF is cleared automatically when the receiver detects an idle
line.
 *
 * Values:
 * - 0b0 - UART receiver idle waiting for a start bit.
 * - 0b1 - UART receiver active ( UART _RXD input not idle).
 */
/*@{*/
/*! @brief Read current value of the UART0_S2_RAF field. */
#define UART0_RD_S2_RAF(base) ((UART0_S2_REG(base) & UART0_S2_RAF_MASK)
>> UART0_S2_RAF_SHIFT)
#define UART0_BRD_S2_RAF(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_RAF_SHIFT, UART0_S2_RAF_WIDTH))
/*}@*/
```

  

```

/*!
 * @name Register UART0_S2, field LBKDE[1] (RW)
 *
 * LBKDE selects a longer break character detection length. While LBKDE
is set,
 * framing error (FE) and receive data register full (RDRF) flags are
prevented
 * from setting.
 *
 * Values:
 * - 0b0 - Break character is detected at length 10 bit times (if M = 0,
SBNS =
 *      0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1,
SBNS = 1
 *      or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1).
 * - 0b1 - Break character is detected at length of 11 bit times (if M =
0, SBNS
 *      = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 14 (if M =
1, SBNS
 *      = 1 or M10 = 1, SNBS = 0) or 15 (if M10 = 1, SNBS = 1).
 */
/*@{*/
/*! @brief Read current value of the UART0_S2_LBKDE field. */
#define UART0_RD_S2_LBKDE(base) ((UART0_S2_REG(base) &
UART0_S2_LBKDE_MASK) >> UART0_S2_LBKDE_SHIFT)
#define UART0_BRD_S2_LBKDE(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_LBKDE_SHIFT, UART0_S2_LBKDE_WIDTH))
```

  

```
/*! @brief Set the LBKDE field to a new value. */
```

```

#define UART0_WR_S2_LBKDE(base, value) (UART0_RMW_S2(base,
UART0_S2_LBKDE_MASK, UART0_S2_LBKDE(value)))
#define UART0_BWR_S2_LBKDE(base, value) (BME_BFI8(&UART0_S2_REG(base),
((uint8_t)(value) << UART0_S2_LBKDE_SHIFT), UART0_S2_LBKDE_SHIFT,
UART0_S2_LBKDE_WIDTH))
/*@}*/

/*
 * @name Register UART0_S2, field BRK13[2] (RW)
 *
 * BRK13 selects a longer transmitted break character length. Detection
of a
 * framing error is not affected by the state of this bit. This bit
should only be
 * changed when the transmitter is disabled.
 *
 * Values:
 * - 0b0 - Break character is transmitted with length of 10 bit times (if
M = 0,
 *      SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if
M = 1,
 *      SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1).
 * - 0b1 - Break character is transmitted with length of 13 bit times (if
M = 0,
 *      SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if
M = 1,
 *      SBNS = 1 or M10 = 1, SNBS = 0) or 16 (if M10 = 1, SNBS = 1).
 */
/*@{*/
/**! @brief Read current value of the UART0_S2_BRK13 field. */
#define UART0_RD_S2_BRK13(base) ((UART0_S2_REG(base) &
UART0_S2_BRK13_MASK) >> UART0_S2_BRK13_SHIFT)
#define UART0_BRD_S2_BRK13(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_BRK13_SHIFT, UART0_S2_BRK13_WIDTH))

/**! @brief Set the BRK13 field to a new value. */
#define UART0_WR_S2_BRK13(base, value) (UART0_RMW_S2(base,
UART0_S2_BRK13_MASK, UART0_S2_BRK13(value)))
#define UART0_BWR_S2_BRK13(base, value) (BME_BFI8(&UART0_S2_REG(base),
((uint8_t)(value) << UART0_S2_BRK13_SHIFT), UART0_S2_BRK13_SHIFT,
UART0_S2_BRK13_WIDTH))
/*@}*/

/*
 * @name Register UART0_S2, field RWUID[3] (RW)
 *
 * RWUID controls whether the idle character that wakes up the receiver
sets the
 * IDLE bit. This bit should only be changed when the receiver is
disabled.
 *
 * Values:
 * - 0b0 - During receive standby state (RWU = 1), the IDLE bit does not
get set

```

```

*      upon detection of an idle character.
* - 0b1 - During receive standby state (RWU = 1), the IDLE bit gets set
upon
*      detection of an idle character.
*/
/*@{*/
/*! @brief Read current value of the UART0_S2_RXUID field. */
#define UART0_RD_S2_RXUID(base) ((UART0_S2_REG(base) &
UART0_S2_RXUID_MASK) >> UART0_S2_RXUID_SHIFT)
#define UART0_BRD_S2_RXUID(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_RXUID_SHIFT, UART0_S2_RXUID_WIDTH))

/*! @brief Set the RXUID field to a new value. */
#define UART0_WR_S2_RXUID(base, value) (UART0_RMW_S2(base,
UART0_S2_RXUID_MASK, UART0_S2_RXUID(value)))
#define UART0_BWR_S2_RXUID(base, value) (BME_BFI8(&UART0_S2_REG(base),
(uint8_t)(value) << UART0_S2_RXUID_SHIFT), UART0_S2_RXUID_SHIFT,
UART0_S2_RXUID_WIDTH)
/*@}*/

/*!
* @name Register UART0_S2, field RXINV[4] (RW)
*
* Setting this bit reverses the polarity of the received data input.
Setting
* RXINV inverts the UART_RXD input for all cases: data bits, start and
stop bits,
* break, and idle.
*
* Values:
* - 0b0 - Receive data not inverted.
* - 0b1 - Receive data inverted.
*/
/*@{*/
/*! @brief Read current value of the UART0_S2_RXINV field. */
#define UART0_RD_S2_RXINV(base) ((UART0_S2_REG(base) &
UART0_S2_RXINV_MASK) >> UART0_S2_RXINV_SHIFT)
#define UART0_BRD_S2_RXINV(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_RXINV_SHIFT, UART0_S2_RXINV_WIDTH))

/*! @brief Set the RXINV field to a new value. */
#define UART0_WR_S2_RXINV(base, value) (UART0_RMW_S2(base,
UART0_S2_RXINV_MASK, UART0_S2_RXINV(value)))
#define UART0_BWR_S2_RXINV(base, value) (BME_BFI8(&UART0_S2_REG(base),
(uint8_t)(value) << UART0_S2_RXINV_SHIFT), UART0_S2_RXINV_SHIFT,
UART0_S2_RXINV_WIDTH)
/*@}*/

/*!
* @name Register UART0_S2, field MSBF[5] (RW)
*
* Setting this bit reverses the order of the bits that are transmitted
and

```

```

    * received on the wire. This bit does not affect the polarity of the
bits, the
    * location of the parity bit or the location of the start or stop bits.
This bit
    * should only be changed when the transmitter and receiver are both
disabled.
    *
    * Values:
    * - 0b0 - LSB (bit0) is the first bit that is transmitted following the
start
        * bit. Further, the first bit received after the start bit is
identified as
        * bit0.
    * - 0b1 - MSB (bit9, bit8, bit7 or bit6) is the first bit that is
transmitted
        * following the start bit depending on the setting of C1[M], C1[PE]
and
        * C4[M10]. Further, the first bit received after the start bit is
identified as
        * bit9, bit8, bit7 or bit6 depending on the setting of C1[M] and
C1[PE].
    */
/*@{*/
/*! @brief Read current value of the UART0_S2_MSBF field. */
#define UART0_RD_S2_MSBF(base) ((UART0_S2_REG(base) & UART0_S2_MSBF_MASK)
>> UART0_S2_MSBF_SHIFT)
#define UART0_BRD_S2_MSBF(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_MSBF_SHIFT, UART0_S2_MSBF_WIDTH))

/*! @brief Set the MSBF field to a new value. */
#define UART0_WR_S2_MSBF(base, value) (UART0_RMW_S2(base,
UART0_S2_MSBF_MASK, UART0_S2_MSBF(value)))
#define UART0_BWR_S2_MSBF(base, value) (BME_BFI8(&UART0_S2_REG(base),
((uint8_t)(value) << UART0_S2_MSBF_SHIFT), UART0_S2_MSBF_SHIFT,
UART0_S2_MSBF_WIDTH))
/*@}*/
/*!
    * @name Register UART0_S2, field RXEDGIF[6] (RW)
    *
    * RXEDGIF is set when an active edge, falling if RXINV = 0, rising if
RXINV=1,
    * on the UART_RX pin occurs. RXEDGIF is cleared by writing a 1 to it.
    *
    * Values:
    * - 0b0 - No active edge on the receive pin has occurred.
    * - 0b1 - An active edge on the receive pin has occurred.
    */
/*@{*/
/*! @brief Read current value of the UART0_S2_RXEDGIF field. */
#define UART0_RD_S2_RXEDGIF(base) ((UART0_S2_REG(base) &
UART0_S2_RXEDGIF_MASK) >> UART0_S2_RXEDGIF_SHIFT)
#define UART0_BRD_S2_RXEDGIF(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_RXEDGIF_SHIFT, UART0_S2_RXEDGIF_WIDTH))

```

```

/*! @brief Set the RXEDGIF field to a new value. */
#define UART0_WR_S2_RXEDGIF(base, value) (UART0_RMW_S2(base,
UART0_S2_RXEDGIF_MASK, UART0_S2_RXEDGIF(value)))
#define UART0_BWR_S2_RXEDGIF(base, value) (BME_BFI8(&UART0_S2_REG(base),
((uint8_t)(value) << UART0_S2_RXEDGIF_SHIFT), UART0_S2_RXEDGIF_SHIFT,
UART0_S2_RXEDGIF_WIDTH))
/*@}*/

/*! 
 * @name Register UART0_S2, field LBKDIF[7] (RW)
 *
 * LBKDIF is set when the LIN break detect circuitry is enabled and a LIN
break
 * character is detected. LBKDIF is cleared by writing a 1 to it.
 *
 * Values:
 * - 0b0 - No LIN break character has been detected.
 * - 0b1 - LIN break character has been detected.
 */
/*@*/
/*! @brief Read current value of the UART0_S2_LBKDIF field. */
#define UART0_RD_S2_LBKDIF(base) ((UART0_S2_REG(base) &
UART0_S2_LBKDIF_MASK) >> UART0_S2_LBKDIF_SHIFT)
#define UART0_BRD_S2_LBKDIF(base) (BME_UBFX8(&UART0_S2_REG(base),
UART0_S2_LBKDIF_SHIFT, UART0_S2_LBKDIF_WIDTH))

/*! @brief Set the LBKDIF field to a new value. */
#define UART0_WR_S2_LBKDIF(base, value) (UART0_RMW_S2(base,
UART0_S2_LBKDIF_MASK, UART0_S2_LBKDIF(value)))
#define UART0_BWR_S2_LBKDIF(base, value) (BME_BFI8(&UART0_S2_REG(base),
((uint8_t)(value) << UART0_S2_LBKDIF_SHIFT), UART0_S2_LBKDIF_SHIFT,
UART0_S2_LBKDIF_WIDTH))
/*@*/

***** 
* UART0_C3 - UART Control Register 3
*****
*/ 

/*! 
 * @brief UART0_C3 - UART Control Register 3 (RW)
 *
 * Reset value: 0x00U
 */
/*! 
 * @name Constants and macros for entire UART0_C3 register
 */
/*@*/
#define UART0_RD_C3(base) (UART0_C3_REG(base))
#define UART0_WR_C3(base, value) (UART0_C3_REG(base) = (value))

```

```

#define UART0_RMW_C3(base, mask, value) (UART0_WR_C3(base,
(UART0_RD_C3(base) & ~(mask)) | (value)))
#define UART0_SET_C3(base, value) (BME_OR8(&UART0_C3_REG(base),
(uint8_t)(value)))
#define UART0_CLR_C3(base, value) (BME_AND8(&UART0_C3_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_C3(base, value) (BME_XOR8(&UART0_C3_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual UART0_C3 bitfields
 */

/*!
 * @name Register UART0_C3, field PEIE[0] (RW)
 *
 * This bit enables the parity error flag (PF) to generate hardware
interrupt
 * requests.
 *
 * Values:
 * - 0b0 - PF interrupts disabled; use polling).
 * - 0b1 - Hardware interrupt requested when PF is set.
 */
/*@{*/
/*! @brief Read current value of the UART0_C3_PEIE field. */
#define UART0_RD_C3_PEIE(base) ((UART0_C3_REG(base) & UART0_C3_PEIE_MASK)
>> UART0_C3_PEIE_SHIFT)
#define UART0_BRD_C3_PEIE(base) (BME_UBFX8(&UART0_C3_REG(base),
UART0_C3_PEIE_SHIFT, UART0_C3_PEIE_WIDTH))

/*! @brief Set the PEIE field to a new value. */
#define UART0_WR_C3_PEIE(base, value) (UART0_RMW_C3(base,
UART0_C3_PEIE_MASK, UART0_C3_PEIE(value)))
#define UART0_BWR_C3_PEIE(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_PEIE_SHIFT), UART0_C3_PEIE_SHIFT,
UART0_C3_PEIE_WIDTH))
/*@}*/

/*
 * @name Register UART0_C3, field FEIE[1] (RW)
 *
 * This bit enables the framing error flag (FE) to generate hardware
interrupt
 * requests.
 *
 * Values:
 * - 0b0 - FE interrupts disabled; use polling.
 * - 0b1 - Hardware interrupt requested when FE is set.
 */
/*@{*/
/*! @brief Read current value of the UART0_C3_FEIE field. */

```

```

#define UART0_RD_C3_FEIE(base) ((UART0_C3_REG(base) & UART0_C3_FEIE_MASK) \
>> UART0_C3_FEIE_SHIFT)
#define UART0_BRD_C3_FEIE(base) (BME_UBFX8(&UART0_C3_REG(base), \
UART0_C3_FEIE_SHIFT, UART0_C3_FEIE_WIDTH))

/*! @brief Set the FEIE field to a new value. */
#define UART0_WR_C3_FEIE(base, value) (UART0_RMW_C3(base, \
UART0_C3_FEIE_MASK, UART0_C3_FEIE(value)))
#define UART0_BWR_C3_FEIE(base, value) (BME_BFI8(&UART0_C3_REG(base), \
((uint8_t)(value) << UART0_C3_FEIE_SHIFT), UART0_C3_FEIE_SHIFT, \
UART0_C3_FEIE_WIDTH))
/*@}*/

/*!
 * @name Register UART0_C3, field NEIE[2] (RW)
 *
 * This bit enables the noise flag (NF) to generate hardware interrupt
requests.
 *
 * Values:
 * - 0b0 - NF interrupts disabled; use polling.
 * - 0b1 - Hardware interrupt requested when NF is set.
 */
/*@{*/
/*! @brief Read current value of the UART0_C3_NEIE field. */
#define UART0_RD_C3_NEIE(base) ((UART0_C3_REG(base) & UART0_C3_NEIE_MASK) \
>> UART0_C3_NEIE_SHIFT)
#define UART0_BRD_C3_NEIE(base) (BME_UBFX8(&UART0_C3_REG(base), \
UART0_C3_NEIE_SHIFT, UART0_C3_NEIE_WIDTH))

/*! @brief Set the NEIE field to a new value. */
#define UART0_WR_C3_NEIE(base, value) (UART0_RMW_C3(base, \
UART0_C3_NEIE_MASK, UART0_C3_NEIE(value)))
#define UART0_BWR_C3_NEIE(base, value) (BME_BFI8(&UART0_C3_REG(base), \
((uint8_t)(value) << UART0_C3_NEIE_SHIFT), UART0_C3_NEIE_SHIFT, \
UART0_C3_NEIE_WIDTH))
/*@}*/

/*!
 * @name Register UART0_C3, field ORIE[3] (RW)
 *
 * This bit enables the overrun flag (OR) to generate hardware interrupt
requests.
 *
 * Values:
 * - 0b0 - OR interrupts disabled; use polling.
 * - 0b1 - Hardware interrupt requested when OR is set.
 */
/*@{*/
/*! @brief Read current value of the UART0_C3_ORIE field. */
#define UART0_RD_C3_ORIE(base) ((UART0_C3_REG(base) & UART0_C3_ORIE_MASK) \
>> UART0_C3_ORIE_SHIFT)
#define UART0_BRD_C3_ORIE(base) (BME_UBFX8(&UART0_C3_REG(base), \
UART0_C3_ORIE_SHIFT, UART0_C3_ORIE_WIDTH))

```

```

/*! @brief Set the ORIE field to a new value. */
#define UART0_WR_C3_ORIE(base, value) (UART0_RMW_C3(base,
UART0_C3_ORIE_MASK, UART0_C3_ORIE(value)))
#define UART0_BWR_C3_ORIE(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_ORIE_SHIFT), UART0_C3_ORIE_SHIFT,
UART0_C3_ORIE_WIDTH))
/*@}*/

/*!!
 * @name Register UART0_C3, field TXINV[4] (RW)
 *
 * Setting this bit reverses the polarity of the transmitted data output.
 * Setting TXINV inverts the UART_TXD output for all cases: data bits,
start and stop
 * bits, break, and idle.
 *
 * Values:
 * - 0b0 - Transmit data not inverted.
 * - 0b1 - Transmit data inverted.
 */
/*@*/
/*! @brief Read current value of the UART0_C3_TXINV field. */
#define UART0_RD_C3_TXINV(base) ((UART0_C3_REG(base) &
UART0_C3_TXINV_MASK) >> UART0_C3_TXINV_SHIFT)
#define UART0_BRD_C3_TXINV(base) (BME_UBFX8(&UART0_C3_REG(base),
UART0_C3_TXINV_SHIFT, UART0_C3_TXINV_WIDTH))

/*! @brief Set the TXINV field to a new value. */
#define UART0_WR_C3_TXINV(base, value) (UART0_RMW_C3(base,
UART0_C3_TXINV_MASK, UART0_C3_TXINV(value)))
#define UART0_BWR_C3_TXINV(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_TXINV_SHIFT), UART0_C3_TXINV_SHIFT,
UART0_C3_TXINV_WIDTH))
/*@*/

/*!!
 * @name Register UART0_C3, field TXDIR[5] (RW)
 *
 * When the is configured for single-wire half-duplex operation (LOOPS =
RSRC =
 * 1), this bit determines the direction of data at the UART_TXD pin.
When
 * clearing TXDIR, the transmitter will finish receiving the current
character (if any)
 * before the receiver starts receiving data from the UART_TXD pin.
 *
 * Values:
 * - 0b0 - UART_TXD pin is an input in single-wire mode.
 * - 0b1 - UART_TXD pin is an output in single-wire mode.
 */
/*@*/
/*! @brief Read current value of the UART0_C3_TXDIR field. */

```

```

#define UART0_RD_C3_TXDIR(base) ((UART0_C3_REG(base) &
UART0_C3_TXDIR_MASK) >> UART0_C3_TXDIR_SHIFT)
#define UART0_BRD_C3_TXDIR(base) (BME_UBFX8(&UART0_C3_REG(base),
UART0_C3_TXDIR_SHIFT, UART0_C3_TXDIR_WIDTH))

/*! @brief Set the TXDIR field to a new value. */
#define UART0_WR_C3_TXDIR(base, value) (UART0_RMW_C3(base,
UART0_C3_TXDIR_MASK, UART0_C3_TXDIR(value)))
#define UART0_BWR_C3_TXDIR(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_TXDIR_SHIFT), UART0_C3_TXDIR_SHIFT,
UART0_C3_TXDIR_WIDTH))
/*@}*/

/*!
 * @name Register UART0_C3, field R9T8[6] (RW)
 *
 * When the UART is configured for 9-bit data (M = 1), T8 may be thought
 * of as a
 * ninth transmit data bit to the left of the msb of the data in the
 * UART_D
 * register. When writing 9-bit data, the entire 9-bit value is
 * transferred to the
 * UART transmit buffer after UART_D is written so T8 should be written,
 * if it
 * needs to change from its previous value, before UART_D is written. If
 * T8 does not
 * need to change in the new value, such as when it is used to generate
 * mark or
 * space parity, it need not be written each time UART_D is written. When
 * the UART
 * is configured for 10-bit data (M10 = 1), R9 can be thought of as a
 * tenth
 * receive data bit. When reading 10-bit data, read R9 and R8 before
 * reading UART_D
 * because reading UART_D completes automatic flag clearing sequences
 * that could
 * allow R8, R9 and UART_D to be overwritten with new data.
 */
/*@{*/
/*! @brief Read current value of the UART0_C3_R9T8 field. */
#define UART0_RD_C3_R9T8(base) ((UART0_C3_REG(base) & UART0_C3_R9T8_MASK)
>> UART0_C3_R9T8_SHIFT)
#define UART0_BRD_C3_R9T8(base) (BME_UBFX8(&UART0_C3_REG(base),
UART0_C3_R9T8_SHIFT, UART0_C3_R9T8_WIDTH))

/*! @brief Set the R9T8 field to a new value. */
#define UART0_WR_C3_R9T8(base, value) (UART0_RMW_C3(base,
UART0_C3_R9T8_MASK, UART0_C3_R9T8(value)))
#define UART0_BWR_C3_R9T8(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_R9T8_SHIFT), UART0_C3_R9T8_SHIFT,
UART0_C3_R9T8_WIDTH))
/*@}*/

/*! */

```

```

* @name Register UART0_C3, field R8T9[7] (RW)
*
* When the UART is configured for 9-bit data (M = 1), R8 can be thought
of as a
* ninth receive data bit to the left of the msb of the buffered data in
the
* UART_D register. When reading 9-bit data, read R8 before reading
UART_D because
* reading UART_D completes automatic flag clearing sequences that could
allow R8
* and UART_D to be overwritten with new data. When the UART is
configured for
* 10-bit data (M10 = 1), T9 may be thought of as a tenth transmit data
bit. When
* writing 10-bit data, the entire 10-bit value is transferred to the
UART
* transmit buffer when UART_D is written so T9 and T8 should be written,
if it needs to
* change from its previous value, before UART_D is written. If T9 and T8
do not
* need to change in the new value, such as when it is used to generate
mark or
* space parity, they need not be written each time UART_D is written.
*/
/*@{*/
/*! @brief Read current value of the UART0_C3_R8T9 field. */
#define UART0_RD_C3_R8T9(base) ((UART0_C3_REG(base) & UART0_C3_R8T9_MASK)
>> UART0_C3_R8T9_SHIFT)
#define UART0_BRD_C3_R8T9(base) (BME_UBFX8(&UART0_C3_REG(base),
UART0_C3_R8T9_SHIFT, UART0_C3_R8T9_WIDTH))

/*! @brief Set the R8T9 field to a new value. */
#define UART0_WR_C3_R8T9(base, value) (UART0_RMW_C3(base,
UART0_C3_R8T9_MASK, UART0_C3_R8T9(value)))
#define UART0_BWR_C3_R8T9(base, value) (BME_BFI8(&UART0_C3_REG(base),
((uint8_t)(value) << UART0_C3_R8T9_SHIFT), UART0_C3_R8T9_SHIFT,
UART0_C3_R8T9_WIDTH))
/*@}*/

/*****
*****
* UART0_D - UART Data Register
*****
****/

/*!
* @brief UART0_D - UART Data Register (RW)
*
* Reset value: 0x00U
*
* This register is actually two separate registers. Reads return the
contents

```

```

    * of the read-only receive data buffer and writes go to the write-only
transmit
    * data buffer. Reads and writes of this register are also involved in
the
    * automatic flag clearing mechanisms for some of the UART status flags.
*/
/*!
    * @name Constants and macros for entire UART0_D register
*/
/*@{*/
#define UART0_RD_D(base)          (UART0_D_REG(base))
#define UART0_WR_D(base, value)   (UART0_D_REG(base) = (value))
#define UART0_RMW_D(base, mask, value) (UART0_WR_D(base,
(UART0_RD_D(base) & ~mask) | (value)))
#define UART0_SET_D(base, value)  (BME_OR8(&UART0_D_REG(base),
(uint8_t)(value)))
#define UART0_CLR_D(base, value)  (BME_AND8(&UART0_D_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_D(base, value)  (BME_XOR8(&UART0_D_REG(base),
(uint8_t)(value)))
/*@}*/
/*
    * Constants & macros for individual UART0_D bitfields
*/
/*!
    * @name Register UART0_D, field R0T0[0] (RW)
    *
    * Read receive data buffer 0 or write transmit data buffer 0.
    */
/*@{*/
/*! @brief Read current value of the UART0_D_R0T0 field. */
#define UART0_RD_D_R0T0(base) ((UART0_D_REG(base) & UART0_D_R0T0_MASK) >>
UART0_D_R0T0_SHIFT)
#define UART0_BRD_D_R0T0(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R0T0_SHIFT, UART0_D_R0T0_WIDTH))

/*! @brief Set the R0T0 field to a new value. */
#define UART0_WR_D_R0T0(base, value) (UART0_RMW_D(base,
UART0_D_R0T0_MASK, UART0_D_R0T0(value)))
#define UART0_BWR_D_R0T0(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R0T0_SHIFT), UART0_D_R0T0_SHIFT,
UART0_D_R0T0_WIDTH))
/*@}*/
/*
    * @name Register UART0_D, field R1T1[1] (RW)
    *
    * Read receive data buffer 1 or write transmit data buffer 1.
    */
/*@{*/
/*! @brief Read current value of the UART0_D_R1T1 field. */

```

```

#define UART0_RD_D_R1T1(base) ((UART0_D_REG(base) & UART0_D_R1T1_MASK) >>
UART0_D_R1T1_SHIFT)
#define UART0_BRD_D_R1T1(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R1T1_SHIFT, UART0_D_R1T1_WIDTH))

/*! @brief Set the R1T1 field to a new value. */
#define UART0_WR_D_R1T1(base, value) (UART0_RMW_D(base,
UART0_D_R1T1_MASK, UART0_D_R1T1(value)))
#define UART0_BWR_D_R1T1(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R1T1_SHIFT), UART0_D_R1T1_SHIFT,
UART0_D_R1T1_WIDTH))
/*@}*/

/*!
 * @name Register UART0_D, field R2T2[2] (RW)
 *
 * Read receive data buffer 2 or write transmit data buffer 2.
 */
/*@{*/
/*! @brief Read current value of the UART0_D_R2T2 field. */
#define UART0_RD_D_R2T2(base) ((UART0_D_REG(base) & UART0_D_R2T2_MASK) >>
UART0_D_R2T2_SHIFT)
#define UART0_BRD_D_R2T2(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R2T2_SHIFT, UART0_D_R2T2_WIDTH))

/*! @brief Set the R2T2 field to a new value. */
#define UART0_WR_D_R2T2(base, value) (UART0_RMW_D(base,
UART0_D_R2T2_MASK, UART0_D_R2T2(value)))
#define UART0_BWR_D_R2T2(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R2T2_SHIFT), UART0_D_R2T2_SHIFT,
UART0_D_R2T2_WIDTH))
/*@}*/

/*!
 * @name Register UART0_D, field R3T3[3] (RW)
 *
 * Read receive data buffer 3 or write transmit data buffer 3.
 */
/*@{*/
/*! @brief Read current value of the UART0_D_R3T3 field. */
#define UART0_RD_D_R3T3(base) ((UART0_D_REG(base) & UART0_D_R3T3_MASK) >>
UART0_D_R3T3_SHIFT)
#define UART0_BRD_D_R3T3(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R3T3_SHIFT, UART0_D_R3T3_WIDTH))

/*! @brief Set the R3T3 field to a new value. */
#define UART0_WR_D_R3T3(base, value) (UART0_RMW_D(base,
UART0_D_R3T3_MASK, UART0_D_R3T3(value)))
#define UART0_BWR_D_R3T3(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R3T3_SHIFT), UART0_D_R3T3_SHIFT,
UART0_D_R3T3_WIDTH))
/*@}*/

/*!

```

```

/* @name Register UART0_D, field R4T4[4] (RW)
*
* Read receive data buffer 4 or write transmit data buffer 4.
*/
/*@{*/
/*! @brief Read current value of the UART0_D_R4T4 field. */
#define UART0_RD_D_R4T4(base) ((UART0_D_REG(base) & UART0_D_R4T4_MASK) >>
UART0_D_R4T4_SHIFT)
#define UART0_BRD_D_R4T4(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R4T4_SHIFT, UART0_D_R4T4_WIDTH))

/*! @brief Set the R4T4 field to a new value. */
#define UART0_WR_D_R4T4(base, value) (UART0_RMW_D(base,
UART0_D_R4T4_MASK, UART0_D_R4T4(value)))
#define UART0_BWR_D_R4T4(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R4T4_SHIFT), UART0_D_R4T4_SHIFT,
UART0_D_R4T4_WIDTH))
/*@}*/
/*!
* @name Register UART0_D, field R5T5[5] (RW)
*
* Read receive data buffer 5 or write transmit data buffer 5.
*/
/*@{*/
/*! @brief Read current value of the UART0_D_R5T5 field. */
#define UART0_RD_D_R5T5(base) ((UART0_D_REG(base) & UART0_D_R5T5_MASK) >>
UART0_D_R5T5_SHIFT)
#define UART0_BRD_D_R5T5(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R5T5_SHIFT, UART0_D_R5T5_WIDTH))

/*! @brief Set the R5T5 field to a new value. */
#define UART0_WR_D_R5T5(base, value) (UART0_RMW_D(base,
UART0_D_R5T5_MASK, UART0_D_R5T5(value)))
#define UART0_BWR_D_R5T5(base, value) (BME_BFI8(&UART0_D_REG(base),
((uint8_t)(value) << UART0_D_R5T5_SHIFT), UART0_D_R5T5_SHIFT,
UART0_D_R5T5_WIDTH))
/*@}*/
/*!
* @name Register UART0_D, field R6T6[6] (RW)
*
* Read receive data buffer 6 or write transmit data buffer 6.
*/
/*@{*/
/*! @brief Read current value of the UART0_D_R6T6 field. */
#define UART0_RD_D_R6T6(base) ((UART0_D_REG(base) & UART0_D_R6T6_MASK) >>
UART0_D_R6T6_SHIFT)
#define UART0_BRD_D_R6T6(base) (BME_UBFX8(&UART0_D_REG(base),
UART0_D_R6T6_SHIFT, UART0_D_R6T6_WIDTH))

/*! @brief Set the R6T6 field to a new value. */
#define UART0_WR_D_R6T6(base, value) (UART0_RMW_D(base,
UART0_D_R6T6_MASK, UART0_D_R6T6(value)))

```

```

#define UART0_BWR_D_R6T6(base, value) (BME_BFI8(&UART0_D_REG(base),  

((uint8_t)(value) << UART0_D_R6T6_SHIFT), UART0_D_R6T6_SHIFT,  

UART0_D_R6T6_WIDTH))  

/*@}*/

/*!  

 * @name Register UART0_D, field R7T7[7] (RW)  

 *  

 * Read receive data buffer 7 or write transmit data buffer 7.  

 */  

/*@{*/  

/*! @brief Read current value of the UART0_D_R7T7 field. */  

#define UART0_RD_D_R7T7(base) ((UART0_D_REG(base) & UART0_D_R7T7_MASK) >>  

UART0_D_R7T7_SHIFT)  

#define UART0_BRD_D_R7T7(base) (BME_UBFX8(&UART0_D_REG(base),  

UART0_D_R7T7_SHIFT, UART0_D_R7T7_WIDTH))

/*! @brief Set the R7T7 field to a new value. */  

#define UART0_WR_D_R7T7(base, value) (UART0_RMW_D(base,  

UART0_D_R7T7_MASK, UART0_D_R7T7(value)))  

#define UART0_BWR_D_R7T7(base, value) (BME_BFI8(&UART0_D_REG(base),  

((uint8_t)(value) << UART0_D_R7T7_SHIFT), UART0_D_R7T7_SHIFT,  

UART0_D_R7T7_WIDTH))  

/*@}/

*****  

* UART0_MA1 - UART Match Address Registers 1  

*****/  

  

/*!  

 * @brief UART0_MA1 - UART Match Address Registers 1 (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The MA1 and MA2 registers are compared to input data addresses when  

the most  

* significant bit is set and the associated C4[MAEN] bit is set. If a  

match  

* occurs, the following data is transferred to the data register. If a  

match fails,  

* the following data is discarded. Software should only write a MA  

register when  

* the associated C4[MAEN] bit is clear.  

*/  

/*!  

 * @name Constants and macros for entire UART0_MA1 register  

*/  

/*@{*/  

#define UART0_RD_MA1(base) (UART0_MA1_REG(base))  

#define UART0_WR_MA1(base, value) (UART0_MA1_REG(base) = (value))

```

```

#define UART0_RMW_MA1(base, mask, value) (UART0_WR_MA1(base,
(UART0_RD_MA1(base) & ~mask) | (value)))
#define UART0_SET_MA1(base, value) (BME_OR8(&UART0_MA1_REG(base),
(uint8_t)(value)))
#define UART0_CLR_MA1(base, value) (BME_AND8(&UART0_MA1_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_MA1(base, value) (BME_XOR8(&UART0_MA1_REG(base),
(uint8_t)(value)))
/*@}*/

/********************* ****
 * UART0_MA2 - UART Match Address Registers 2
***** */

/*!!
 * @brief UART0_MA2 - UART Match Address Registers 2 (RW)
 *
 * Reset value: 0x00U
 *
 * The MA1 and MA2 registers are compared to input data addresses when
the most
 * significant bit is set and the associated C4[MAEN] bit is set. If a
match
 * occurs, the following data is transferred to the data register. If a
match fails,
 * the following data is discarded. Software should only write a MA
register when
 * the associated C4[MAEN] bit is clear.
 */
/*!!
 * @name Constants and macros for entire UART0_MA2 register
 */
/*@{*/
#define UART0_RD_MA2(base) (UART0_MA2_REG(base))
#define UART0_WR_MA2(base, value) (UART0_MA2_REG(base) = (value))
#define UART0_RMW_MA2(base, mask, value) (UART0_WR_MA2(base,
(UART0_RD_MA2(base) & ~mask) | (value)))
#define UART0_SET_MA2(base, value) (BME_OR8(&UART0_MA2_REG(base),
(uint8_t)(value)))
#define UART0_CLR_MA2(base, value) (BME_AND8(&UART0_MA2_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_MA2(base, value) (BME_XOR8(&UART0_MA2_REG(base),
(uint8_t)(value)))
/*@}*/

/********************* ****
 * UART0_C4 - UART Control Register 4
***** */

/*@}*/

```

```

/*!
 * @brief UART0_C4 - UART Control Register 4 (RW)
 *
 * Reset value: 0x0FU
 */
/*!
 * @name Constants and macros for entire UART0_C4 register
 */
/*@{*/
#define UART0_RD_C4(base)          (UART0_C4_REG(base))
#define UART0_WR_C4(base, value)   (UART0_C4_REG(base) = (value))
#define UART0_RMW_C4(base, mask, value) (UART0_WR_C4(base,
(UART0_RD_C4(base) & ~mask) | (value)))
#define UART0_SET_C4(base, value)  (BME_OR8(&UART0_C4_REG(base),
(uint8_t)(value)))
#define UART0_CLR_C4(base, value)  (BME_AND8(&UART0_C4_REG(base),
(uint8_t)(~(value))))
#define UART0_TOG_C4(base, value)  (BME_XOR8(&UART0_C4_REG(base),
(uint8_t)(value)))
/*@}*/
/*
 * Constants & macros for individual UART0_C4 bitfields
 */

/*!
 * @name Register UART0_C4, field OSR[4:0] (RW)
 *
 * This field configures the oversampling ratio for the receiver between
4x
 * (00011) and 32x (11111). Writing an invalid oversampling ratio will
default to an
 * oversampling ratio of 16 (01111). This field should only be changed
when the
 * transmitter and receiver are both disabled.
 */
/*@{*/
/*! @brief Read current value of the UART0_C4_OSР field. */
#define UART0_RD_C4_OSР(base) ((UART0_C4_REG(base) & UART0_C4_OSР_MASK)
>> UART0_C4_OSР_SHIFT)
#define UART0_BRD_C4_OSР(base) (BME_UBFX8(&UART0_C4_REG(base),
UART0_C4_OSР_SHIFT, UART0_C4_OSР_WIDTH))

/*! @brief Set the OSR field to a new value. */
#define UART0_WR_C4_OSР(base, value) (UART0_RMW_C4(base,
UART0_C4_OSР_MASK, UART0_C4_OSР(value)))
#define UART0_BWR_C4_OSР(base, value) (BME_BFI8(&UART0_C4_REG(base),
((uint8_t)(value) << UART0_C4_OSР_SHIFT), UART0_C4_OSР_SHIFT,
UART0_C4_OSР_WIDTH))
/*@}*/

/*
 * @name Register UART0_C4, field M10[5] (RW)

```

```

/*
 * The M10 bit causes a tenth bit to be part of the serial transmission.
This
 * bit should only be changed when the transmitter and receiver are both
disabled.
 *
 * Values:
 * - 0b0 - Receiver and transmitter use 8-bit or 9-bit data characters.
 * - 0b1 - Receiver and transmitter use 10-bit data characters.
 */
/*@{*/
/*! @brief Read current value of the UART0_C4_M10 field. */
#define UART0_RD_C4_M10(base) ((UART0_C4_REG(base) & UART0_C4_M10_MASK)
>> UART0_C4_M10_SHIFT)
#define UART0_BRD_C4_M10(base) (BME_UBFX8(&UART0_C4_REG(base),
UART0_C4_M10_SHIFT, UART0_C4_M10_WIDTH))

/*! @brief Set the M10 field to a new value. */
#define UART0_WR_C4_M10(base, value) (UART0_RMW_C4(base,
UART0_C4_M10_MASK, UART0_C4_M10(value)))
#define UART0_BWR_C4_M10(base, value) (BME_BFI8(&UART0_C4_REG(base),
((uint8_t)(value) << UART0_C4_M10_SHIFT), UART0_C4_M10_SHIFT,
UART0_C4_M10_WIDTH))
/*@}*/

/*
 * @name Register UART0_C4, field MAEN2[6] (RW)
 *
 * Refer to Match address operation for more information.
 *
 * Values:
 * - 0b0 - All data received is transferred to the data buffer if MAEN1
is
 *         cleared.
 * - 0b1 - All data received with the most significant bit cleared, is
 *         discarded. All data received with the most significant bit set, is
compared with
 *         contents of MA2 register. If no match occurs, the data is
discarded. If
 *         match occurs, data is transferred to the data buffer.
 */
/*@{*/
/*! @brief Read current value of the UART0_C4_MAEN2 field. */
#define UART0_RD_C4_MAEN2(base) ((UART0_C4_REG(base) &
UART0_C4_MAEN2_MASK) >> UART0_C4_MAEN2_SHIFT)
#define UART0_BRD_C4_MAEN2(base) (BME_UBFX8(&UART0_C4_REG(base),
UART0_C4_MAEN2_SHIFT, UART0_C4_MAEN2_WIDTH))

/*! @brief Set the MAEN2 field to a new value. */
#define UART0_WR_C4_MAEN2(base, value) (UART0_RMW_C4(base,
UART0_C4_MAEN2_MASK, UART0_C4_MAEN2(value)))
#define UART0_BWR_C4_MAEN2(base, value) (BME_BFI8(&UART0_C4_REG(base),
((uint8_t)(value) << UART0_C4_MAEN2_SHIFT), UART0_C4_MAEN2_SHIFT,
UART0_C4_MAEN2_WIDTH))

```

```

/*@}*/

/*
 * @name Register UART0_C4, field MAEN1[7] (RW)
 *
 * Refer to Match address operation for more information.
 *
 * Values:
 * - 0b0 - All data received is transferred to the data buffer if MAEN2
is
 *     cleared.
 * - 0b1 - All data received with the most significant bit cleared, is
 *     discarded. All data received with the most significant bit set, is
compared with
 *     contents of MA1 register. If no match occurs, the data is
discarded. If
 *     match occurs, data is transferred to the data buffer.
 */
/*@{*/
/*! @brief Read current value of the UART0_C4_MAEN1 field. */
#define UART0_RD_C4_MAEN1(base) ((UART0_C4_REG(base) &
UART0_C4_MAEN1_MASK) >> UART0_C4_MAEN1_SHIFT)
#define UART0_BRD_C4_MAEN1(base) (BME_UBFX8(&UART0_C4_REG(base),
UART0_C4_MAEN1_SHIFT, UART0_C4_MAEN1_WIDTH))

/*! @brief Set the MAEN1 field to a new value. */
#define UART0_WR_C4_MAEN1(base, value) (UART0_RMW_C4(base,
UART0_C4_MAEN1_MASK, UART0_C4_MAEN1(value)))
#define UART0_BWR_C4_MAEN1(base, value) (BME_BFI8(&UART0_C4_REG(base),
(uint8_t)(value) << UART0_C4_MAEN1_SHIFT), UART0_C4_MAEN1_SHIFT,
UART0_C4_MAEN1_WIDTH)
/*@}/

*****  

* UART0_C5 - UART Control Register 5  

*****  

/*!  

 * @brief UART0_C5 - UART Control Register 5 (RW)
 *
 * Reset value: 0x00U
 */
/*!  

 * @name Constants and macros for entire UART0_C5 register
 */
/*@{*/
#define UART0_RD_C5(base) (UART0_C5_REG(base))
#define UART0_WR_C5(base, value) (UART0_C5_REG(base) = (value))
#define UART0_RMW_C5(base, mask, value) (UART0_WR_C5(base,
(UART0_RD_C5(base) & ~mask) | (value)))

```

```

#define UART0_SET_C5(base, value) (BME_OR8 (&UART0_C5_REG(base),  

(uint8_t)(value)))  

#define UART0_CLR_C5(base, value) (BME_AND8 (&UART0_C5_REG(base),  

(uint8_t)(~(value))))  

#define UART0_TOG_C5(base, value) (BME_XOR8 (&UART0_C5_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual UART0_C5 bitfields
 */

/*!  

 * @name Register UART0_C5, field RESYNCDIS[0] (RW)  

 *  

 * When set, disables the resynchronization of the received data word  

when a  

 * data one followed by data zero transition is detected. This bit should  

only be  

 * changed when the receiver is disabled.  

 *  

 * Values:  

 * - 0b0 - Resynchronization during received data word is supported  

 * - 0b1 - Resynchronization during received data word is disabled  

 */
/*@{*/  

/*! @brief Read current value of the UART0_C5_RESYNCDIS field. */  

#define UART0_RD_C5_RESYNCDIS(base) ((UART0_C5_REG(base) &  

UART0_C5_RESYNCDIS_MASK) >> UART0_C5_RESYNCDIS_SHIFT)  

#define UART0_BRD_C5_RESYNCDIS(base) (BME_UBFX8(&UART0_C5_REG(base),  

UART0_C5_RESYNCDIS_SHIFT, UART0_C5_RESYNCDIS_WIDTH))  

/*! @brief Set the RESYNCDIS field to a new value. */  

#define UART0_WR_C5_RESYNCDIS(base, value) (UART0_RMW_C5(base,  

UART0_C5_RESYNCDIS_MASK, UART0_C5_RESYNCDIS(value)))  

#define UART0_BWR_C5_RESYNCDIS(base, value)  

(BME_BFI8(&UART0_C5_REG(base), ((uint8_t)(value) <<  

UART0_C5_RESYNCDIS_SHIFT), UART0_C5_RESYNCDIS_SHIFT,  

UART0_C5_RESYNCDIS_WIDTH))  

/*@}/
```

```

/*!  

 * @name Register UART0_C5, field BOTHEDGE[1] (RW)  

 *  

 * Enables sampling of the received data on both edges of the baud rate  

clock,  

 * effectively doubling the number of times the receiver samples the  

input data  

 * for a given oversampling ratio. This bit must be set for oversampling  

ratios  

 * between x4 and x7 and is optional for higher oversampling ratios. This  

bit should  

 * only be changed when the receiver is disabled.  

 *

```

```

* Values:
* - 0b0 - Receiver samples input data using the rising edge of the baud
rate
*     clock.
* - 0b1 - Receiver samples input data using the rising and falling edge
of the
*     baud rate clock.
*/
/*@{/*/
/*! @brief Read current value of the UART0_C5_BOTHEDGE field. */
#define UART0_RD_C5_BOTHEDGE(base) ((UART0_C5_REG(base) &
UART0_C5_BOTHEDGE_MASK) >> UART0_C5_BOTHEDGE_SHIFT)
#define UART0_BRD_C5_BOTHEDGE(base) (BME_UBFX8(&UART0_C5_REG(base),
UART0_C5_BOTHEDGE_SHIFT, UART0_C5_BOTHEDGE_WIDTH))

/*! @brief Set the BOTHEDGE field to a new value. */
#define UART0_WR_C5_BOTHEDGE(base, value) (UART0_RMW_C5(base,
UART0_C5_BOTHEDGE_MASK, UART0_C5_BOTHEDGE(value)))
#define UART0_BWR_C5_BOTHEDGE(base, value) (BME_BFI8(&UART0_C5_REG(base),
((uint8_t)(value) << UART0_C5_BOTHEDGE_SHIFT), UART0_C5_BOTHEDGE_SHIFT,
UART0_C5_BOTHEDGE_WIDTH))
/*@{/*/

/*!!
* @name Register UART0_C5, field RDMAE[5] (RW)
*
* RDMAE configures the receiver data register full flag, S1[RDRF], to
generate
* a DMA request.
*
* Values:
* - 0b0 - DMA request disabled.
* - 0b1 - DMA request enabled.
*/
/*@{/*/
/*! @brief Read current value of the UART0_C5_RDMAE field. */
#define UART0_RD_C5_RDMAE(base) ((UART0_C5_REG(base) &
UART0_C5_RDMAE_MASK) >> UART0_C5_RDMAE_SHIFT)
#define UART0_BRD_C5_RDMAE(base) (BME_UBFX8(&UART0_C5_REG(base),
UART0_C5_RDMAE_SHIFT, UART0_C5_RDMAE_WIDTH))

/*! @brief Set the RDMAE field to a new value. */
#define UART0_WR_C5_RDMAE(base, value) (UART0_RMW_C5(base,
UART0_C5_RDMAE_MASK, UART0_C5_RDMAE(value)))
#define UART0_BWR_C5_RDMAE(base, value) (BME_BFI8(&UART0_C5_REG(base),
((uint8_t)(value) << UART0_C5_RDMAE_SHIFT), UART0_C5_RDMAE_SHIFT,
UART0_C5_RDMAE_WIDTH))
/*@{/*/

/*!!
* @name Register UART0_C5, field TDMAE[7] (RW)
*
* TDMAE configures the transmit data register empty flag, S1[TDRE], to
generate

```

```

* a DMA request.
*
* Values:
* - 0b0 - DMA request disabled.
* - 0b1 - DMA request enabled.
*/
/*@{ */
/*! @brief Read current value of the UART0_C5_TDMAE field. */
#define UART0_RD_C5_TDMAE(base) ((UART0_C5_REG(base) &
UART0_C5_TDMAE_MASK) >> UART0_C5_TDMAE_SHIFT)
#define UART0_BRD_C5_TDMAE(base) (BME_UBFX8(&UART0_C5_REG(base),
UART0_C5_TDMAE_SHIFT, UART0_C5_TDMAE_WIDTH))

/*! @brief Set the TDMAE field to a new value. */
#define UART0_WR_C5_TDMAE(base, value) (UART0_RMW_C5(base,
UART0_C5_TDMAE_MASK, UART0_C5_TDMAE(value)))
#define UART0_BWR_C5_TDMAE(base, value) (BME_BFI8(&UART0_C5_REG(base),
((uint8_t)(value) << UART0_C5_TDMAE_SHIFT), UART0_C5_TDMAE_SHIFT,
UART0_C5_TDMAE_WIDTH))
/*@} */

/*
* MKL25Z4 USB
*
* Universal Serial Bus, OTG Capable Controller
*
* Registers defined in this header file:
* - USB_PERID - Peripheral ID register
* - USB_IDCOMP - Peripheral ID Complement register
* - USB_REV - Peripheral Revision register
* - USB_ADDINFO - Peripheral Additional Info register
* - USB_OTGSTAT - OTG Interrupt Status register
* - USB_OTGICR - OTG Interrupt Control Register
* - USB_OTGSTAT - OTG Status register
* - USB_OTGCTL - OTG Control register
* - USB_ISTAT - Interrupt Status register
* - USB_INTEN - Interrupt Enable register
* - USB_ERRSTAT - Error Interrupt Status register
* - USB_ERREN - Error Interrupt Enable register
* - USB_STAT - Status register
* - USB_CTL - Control register
* - USB_ADDR - Address register
* - USB_BDTPAGE1 - BDT Page Register 1
* - USB_FRMNUML - Frame Number Register Low
* - USB_FRMNUMH - Frame Number Register High
* - USB_TOKEN - Token register
* - USB_SOFTHLD - SOF Threshold Register
* - USB_BDTPAGE2 - BDT Page Register 2
* - USB_BDTPAGE3 - BDT Page Register 3
* - USB_ENDPT - Endpoint Control register
* - USB_USBCTRL - USB Control register
* - USB_OBSERVE - USB OTG Observe register
* - USB_CONTROL - USB OTG Control register
* - USB_USBTRC0 - USB Transceiver Control Register 0

```

```

* - USB_USBFRMADJUST - Frame Adjust Register
*/
#define USB_INSTANCE_COUNT (1U) /*!< Number of instances of the USB
module. */
#define USB0_IDX (0U) /*!< Instance number for USB0. */

*****  

*****  

* USB_PERID - Peripheral ID register  

*****  

*****/  

  

/*!  

* @brief USB_PERID - Peripheral ID register (RO)  

*  

* Reset value: 0x04U  

*  

* Reads back the value of 0x04. This value is defined for the USB
peripheral.  

*/  

/*!  

* @name Constants and macros for entire USB_PERID register  

*/  

/*@{ */  

#define USB_RD_PERID(base)          (USB_PERID_REG(base))  

/*}@*/  

  

/*  

* Constants & macros for individual USB_PERID bitfields  

*/  

  

/*!  

* @name Register USB_PERID, field ID[5:0] (RO)  

*  

* This field always reads 0x4h.  

*/  

/*@{ */  

/*! @brief Read current value of the USB_PERID_ID field. */  

#define USB_RD_PERID_ID(base) ((USB_PERID_REG(base) & USB_PERID_ID_MASK)  

>> USB_PERID_ID_SHIFT)  

#define USB_BRD_PERID_ID(base) (BME_UBFX8(&USB_PERID_REG(base),  

USB_PERID_ID_SHIFT, USB_PERID_ID_WIDTH))  

/*}@*/  

  

*****  

*****  

* USB_IDCOMP - Peripheral ID Complement register  

*****  

*****/  

  

/*!

```

```

* @brief USB_IDCOMP - Peripheral ID Complement register (RO)
*
* Reset value: 0xFB0
*
* Reads back the complement of the Peripheral ID register. For the USB
* peripheral, the value is 0xFB.
*/
/*!
* @name Constants and macros for entire USB_IDCOMP register
*/
/*@{ */
#define USB_RD_IDCOMP(base)      (USB_IDCOMP_REG(base))
/*}@*/



/*
* Constants & macros for individual USB_IDCOMP bitfields
*/



/*!
* @name Register USB_IDCOMP, field NID[5:0] (RO)
*
* Ones complement of peripheral identification bits.
*/
/*@{ */
/*! @brief Read current value of the USB_IDCOMP_NID field. */
#define USB_RD_IDCOMP_NID(base) ((USB_IDCOMP_REG(base) &
USB_IDCOMP_NID_MASK) >> USB_IDCOMP_NID_SHIFT)
#define USB_BRD_IDCOMP_NID(base) (BME_UBFX8(&USB_IDCOMP_REG(base),
USB_IDCOMP_NID_SHIFT, USB_IDCOMP_NID_WIDTH))
/*}@*/



*****  

*****  

* USB_REV - Peripheral Revision register  

*****  

*****  




/*!
* @brief USB_REV - Peripheral Revision register (RO)
*
* Reset value: 0x33U
*
* Contains the revision number of the USB module.
*/
/*!
* @name Constants and macros for entire USB_REV register
*/
/*@{ */
#define USB_RD_REV(base)        (USB_REV_REG(base))
/*}@*/



*****  

*****
```

```

 * USB_ADDINFO - Peripheral Additional Info register
 ****
 */

/*!
 * @brief USB_ADDINFO - Peripheral Additional Info register (RO)
 *
 * Reset value: 0x01U
 *
 * Reads back the value of the fixed Interrupt Request Level (IRQNUM)
 * along with
 * the Host Enable bit.
 */
/*!
 * @name Constants and macros for entire USB_ADDINFO register
 */
/*@{*/
#define USB_RD_ADDINFO(base)      (USB_ADDINFO_REG(base))
/*}@*/



/*
 * Constants & macros for individual USB_ADDINFO bitfields
 */

/*!
 * @name Register USB_ADDINFO, field IEHOST[0] (RO)
 *
 * When this bit is set, the USB peripheral is operating in host mode.
 */
/*@{*/
/*! @brief Read current value of the USB_ADDINFO_IEHOST field. */
#define USB_RD_ADDINFO_IEHOST(base) ((USB_ADDINFO_REG(base) &
USB_ADDINFO_IEHOST_MASK) >> USB_ADDINFO_IEHOST_SHIFT)
#define USB_BRD_ADDINFO_IEHOST(base) (BME_UBFX8(&USB_ADDINFO_REG(base),
USB_ADDINFO_IEHOST_SHIFT, USB_ADDINFO_IEHOST_WIDTH))
/*}@*/



/*!
 * @name Register USB_ADDINFO, field IRQNUM[7:3] (RO)
 */
/*@{*/
/*! @brief Read current value of the USB_ADDINFO_IRQNUM field. */
#define USB_RD_ADDINFO_IRQNUM(base) ((USB_ADDINFO_REG(base) &
USB_ADDINFO_IRQNUM_MASK) >> USB_ADDINFO_IRQNUM_SHIFT)
#define USB_BRD_ADDINFO_IRQNUM(base) (BME_UBFX8(&USB_ADDINFO_REG(base),
USB_ADDINFO_IRQNUM_SHIFT, USB_ADDINFO_IRQNUM_WIDTH))
/*}@*/



/*
 * USB_OTGISTAT - OTG Interrupt Status register
 ****
 */

```

```
*****
****/


/*!
 * @brief USB_OTGISTAT - OTG Interrupt Status register (RW)
 *
 * Reset value: 0x00U
 *
 * Records changes of the ID sense and VBUS signals. Software can read
this
 * register to determine the event that triggers interrupt. Only bits
that have
 * changed since the last software read are set. Writing a one to a bit
clears the
 * associated interrupt.
 */
/*!
 * @name Constants and macros for entire USB_OTGISTAT register
 */
/*@{*/
#define USB_RD_OTGISTAT(base)      (USB_OTGISTAT_REG(base))
#define USB_WR_OTGISTAT(base, value) (USB_OTGISTAT_REG(base) = (value))
#define USB_RMW_OTGISTAT(base, mask, value) (USB_WR_OTGISTAT(base,
(USB_RD_OTGISTAT(base) & ~mask) | (value)))
#define USB_SET_OTGISTAT(base, value) (BME_OR8(&USB_OTGISTAT_REG(base),
(uint8_t)(value)))
#define USB_CLR_OTGISTAT(base, value) (BME_AND8(&USB_OTGISTAT_REG(base),
(uint8_t)(~value)))
#define USB_TOG_OTGISTAT(base, value) (BME_XOR8(&USB_OTGISTAT_REG(base),
(uint8_t)(value)))
/*@}*/
/*@
 * Constants & macros for individual USB_OTGISTAT bitfields
*/
/*!
 * @name Register USB_OTGISTAT, field AVBUSCHG[0] (RW)
 *
 * This bit is set when a change in VBUS is detected on an A device.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_AVBUSCHG field. */
#define USB_RD_OTGISTAT_AVBUSCHG(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_AVBUSCHG_MASK) >> USB_OTGISTAT_AVBUSCHG_SHIFT)
#define USB_BRD_OTGISTAT_AVBUSCHG(base)
(BME_UBX8(&USB_OTGISTAT_REG(base), USB_OTGISTAT_AVBUSCHG_SHIFT,
USB_OTGISTAT_AVBUSCHG_WIDTH))

/*! @brief Set the AVBUSCHG field to a new value. */
#define USB_WR_OTGISTAT_AVBUSCHG(base, value) (USB_RMW_OTGISTAT(base,
USB_OTGISTAT_AVBUSCHG_MASK, USB_OTGISTAT_AVBUSCHG(value)))

```

```

#define USB_BWR_OTGISTAT_AVBUSCHG(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_AVBUSCHG_SHIFT), USB_OTGISTAT_AVBUSCHG_SHIFT,
USB_OTGISTAT_AVBUSCHG_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGISTAT, field B_SESS_CHG[2] (RW)
 *
 * This bit is set when a change in VBUS is detected on a B device.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_B_SESS_CHG field. */
#define USB_RD_OTGISTAT_B_SESS_CHG(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_B_SESS_CHG_MASK) >> USB_OTGISTAT_B_SESS_CHG_SHIFT)
#define USB_BRD_OTGISTAT_B_SESS_CHG(base)
(BME_UBFX8(&USB_OTGISTAT_REG(base), USB_OTGISTAT_B_SESS_CHG_SHIFT,
USB_OTGISTAT_B_SESS_CHG_WIDTH))

/*! @brief Set the B_SESS_CHG field to a new value. */
#define USB_WR_OTGISTAT_B_SESS_CHG(base, value) (USB_RMW_OTGISTAT(base,
USB_OTGISTAT_B_SESS_CHG_MASK, USB_OTGISTAT_B_SESS_CHG(value)))
#define USB_BWR_OTGISTAT_B_SESS_CHG(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_B_SESS_CHG_SHIFT), USB_OTGISTAT_B_SESS_CHG_SHIFT,
USB_OTGISTAT_B_SESS_CHG_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGISTAT, field SESSVLDCHG[3] (RW)
 *
 * This bit is set when a change in VBUS is detected indicating a session
valid
 * or a session no longer valid.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_SESSVLDCHG field. */
#define USB_RD_OTGISTAT_SESSVLDCHG(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_SESSVLDCHG_MASK) >> USB_OTGISTAT_SESSVLDCHG_SHIFT)
#define USB_BRD_OTGISTAT_SESSVLDCHG(base)
(BME_UBFX8(&USB_OTGISTAT_REG(base), USB_OTGISTAT_SESSVLDCHG_SHIFT,
USB_OTGISTAT_SESSVLDCHG_WIDTH))

/*! @brief Set the SESSVLDCHG field to a new value. */
#define USB_WR_OTGISTAT_SESSVLDCHG(base, value) (USB_RMW_OTGISTAT(base,
USB_OTGISTAT_SESSVLDCHG_MASK, USB_OTGISTAT_SESSVLDCHG(value)))
#define USB_BWR_OTGISTAT_SESSVLDCHG(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_SESSVLDCHG_SHIFT), USB_OTGISTAT_SESSVLDCHG_SHIFT,
USB_OTGISTAT_SESSVLDCHG_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGISTAT, field LINE_STATE_CHG[5] (RW)

```

```

/*
 * This bit is set when the USB line state changes. The interrupt
associated
 * with this bit can be used to detect Reset, Resume, Connect, and Data
Line Pulse
 * signaling
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_LINE_STATE_CHG field.
*/
#define USB_RD_OTGISTAT_LINE_STATE_CHG(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_LINE_STATE_CHG_MASK) >> USB_OTGISTAT_LINE_STATE_CHG_SHIFT)
#define USB_BRD_OTGISTAT_LINE_STATE_CHG(base)
(BME_UBFX8(&USB_OTGISTAT_REG(base), USB_OTGISTAT_LINE_STATE_CHG_SHIFT,
USB_OTGISTAT_LINE_STATE_CHG_WIDTH))

/*! @brief Set the LINE_STATE_CHG field to a new value. */
#define USB_WR_OTGISTAT_LINE_STATE_CHG(base, value)
(USB_RMW_OTGISTAT(base, USB_OTGISTAT_LINE_STATE_CHG_MASK,
USB_OTGISTAT_LINE_STATE_CHG(value)))
#define USB_BWR_OTGISTAT_LINE_STATE_CHG(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_LINE_STATE_CHG_SHIFT), USB_OTGISTAT_LINE_STATE_CHG_SHIFT,
USB_OTGISTAT_LINE_STATE_CHG_WIDTH))
/*}@*/
```

```

/*!
 * @name Register USB_OTGISTAT, field ONEMSEC[6] (RW)
 *
 * This bit is set when the 1 millisecond timer expires. This bit stays
asserted
 * until cleared by software. The interrupt must be serviced every
millisecond
 * to avoid losing 1msec counts.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_ONEMSEC field. */
#define USB_RD_OTGISTAT_ONEMSEC(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_ONEMSEC_MASK) >> USB_OTGISTAT_ONEMSEC_SHIFT)
#define USB_BRD_OTGISTAT_ONEMSEC(base)
(BME_UBFX8(&USB_OTGISTAT_REG(base), USB_OTGISTAT_ONEMSEC_SHIFT,
USB_OTGISTAT_ONEMSEC_WIDTH))

/*! @brief Set the ONEMSEC field to a new value. */
#define USB_WR_OTGISTAT_ONEMSEC(base, value) (USB_RMW_OTGISTAT(base,
USB_OTGISTAT_ONEMSEC_MASK, USB_OTGISTAT_ONEMSEC(value)))
#define USB_BWR_OTGISTAT_ONEMSEC(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_ONEMSEC_SHIFT), USB_OTGISTAT_ONEMSEC_SHIFT,
USB_OTGISTAT_ONEMSEC_WIDTH))
/*}@*/
```

```

/*!
 * @name Register USB_OTGISTAT, field IDCHG[7] (RW)

```

```

/*
 * This bit is set when a change in the ID Signal from the USB connector
is
 * sensed.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGISTAT_IDCHG field. */
#define USB_RD_OTGISTAT_IDCHG(base) ((USB_OTGISTAT_REG(base) &
USB_OTGISTAT_IDCHG_MASK) >> USB_OTGISTAT_IDCHG_SHIFT)
#define USB_BRD_OTGISTAT_IDCHG(base) (BME_UBFX8(&USB_OTGISTAT_REG(base),
USB_OTGISTAT_IDCHG_SHIFT, USB_OTGISTAT_IDCHG_WIDTH))

/*! @brief Set the IDCHG field to a new value. */
#define USB_WR_OTGISTAT_IDCHG(base, value) (USB_RMW_OTGISTAT(base,
USB_OTGISTAT_IDCHG_MASK, USB_OTGISTAT_IDCHG(value)))
#define USB_BWR_OTGISTAT_IDCHG(base, value)
(BME_BFI8(&USB_OTGISTAT_REG(base), ((uint8_t)(value) <<
USB_OTGISTAT_IDCHG_SHIFT), USB_OTGISTAT_IDCHG_SHIFT,
USB_OTGISTAT_IDCHG_WIDTH))
/*@}*/
/******
 * USB_OTGICR - OTG Interrupt Control Register
*****/
/*!
 * @brief USB_OTGICR - OTG Interrupt Control Register (RW)
 *
 * Reset value: 0x00U
 *
 * Enables the corresponding interrupt status bits defined in the OTG
Interrupt
 * Status Register.
 */
/*!
 * @name Constants and macros for entire USB_OTGICR register
 */
/*@{*/
#define USB_RD_OTGICR(base) (USB_OTGICR_REG(base))
#define USB_WR_OTGICR(base, value) (USB_OTGICR_REG(base) = (value))
#define USB_RMW_OTGICR(base, mask, value) (USB_WR_OTGICR(base,
(USB_RD_OTGICR(base) & ~mask) | (value)))
#define USB_SET_OTGICR(base, value) (BME_OR8(&USB_OTGICR_REG(base),
(uint8_t)(value)))
#define USB_CLR_OTGICR(base, value) (BME_AND8(&USB_OTGICR_REG(base),
(uint8_t)(~value)))
#define USB_TOG_OTGICR(base, value) (BME_XOR8(&USB_OTGICR_REG(base),
(uint8_t)(value)))
/*@}*/
/*

```

```

 * Constants & macros for individual USB_OTGICR bitfields
 */

/*!
 * @name Register USB_OTGICR, field AVBUSEN[0] (RW)
 *
 * Values:
 * - 0b0 - Disables the AVBUSCHG interrupt.
 * - 0b1 - Enables the AVBUSCHG interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGICR_AVBUSEN field. */
#define USB_RD_OTGICR_AVBUSEN(base) ((USB_OTGICR_REG(base) &
USB_OTGICR_AVBUSEN_MASK) >> USB_OTGICR_AVBUSEN_SHIFT)
#define USB_BRD_OTGICR_AVBUSEN(base) (BME_UBXF8(&USB_OTGICR_REG(base),
USB_OTGICR_AVBUSEN_SHIFT, USB_OTGICR_AVBUSEN_WIDTH))

/*! @brief Set the AVBUSEN field to a new value. */
#define USB_WR_OTGICR_AVBUSEN(base, value) (USB_RMW_OTGICR(base,
USB_OTGICR_AVBUSEN_MASK, USB_OTGICR_AVBUSEN(value)))
#define USB_BWR_OTGICR_AVBUSEN(base, value)
(BME_BFI8(&USB_OTGICR_REG(base), ((uint8_t)(value) <<
USB_OTGICR_AVBUSEN_SHIFT), USB_OTGICR_AVBUSEN_SHIFT,
USB_OTGICR_AVBUSEN_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGICR, field BSESSSEN[2] (RW)
 *
 * Values:
 * - 0b0 - Disables the B_SESS_CHG interrupt.
 * - 0b1 - Enables the B_SESS_CHG interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGICR_BSESSSEN field. */
#define USB_RD_OTGICR_BSESSSEN(base) ((USB_OTGICR_REG(base) &
USB_OTGICR_BSESSSEN_MASK) >> USB_OTGICR_BSESSSEN_SHIFT)
#define USB_BRD_OTGICR_BSESSSEN(base) (BME_UBXF8(&USB_OTGICR_REG(base),
USB_OTGICR_BSESSSEN_SHIFT, USB_OTGICR_BSESSSEN_WIDTH))

/*! @brief Set the BSESSSEN field to a new value. */
#define USB_WR_OTGICR_BSESSSEN(base, value) (USB_RMW_OTGICR(base,
USB_OTGICR_BSESSSEN_MASK, USB_OTGICR_BSESSSEN(value)))
#define USB_BWR_OTGICR_BSESSSEN(base, value)
(BME_BFI8(&USB_OTGICR_REG(base), ((uint8_t)(value) <<
USB_OTGICR_BSESSSEN_SHIFT), USB_OTGICR_BSESSSEN_SHIFT,
USB_OTGICR_BSESSSEN_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGICR, field SESSVLDEN[3] (RW)
 *
 * Values:
 * - 0b0 - Disables the SESSVLDCHG interrupt.

```

```

* - 0b1 - Enables the SESSVLDCHG interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_OTGICR_SESSVLDEN field. */
#define USB_RD_OTGICR_SESSVLDEN(base) ((USB_OTGICR_REG(base) &
USB_OTGICR_SESSVLDEN_MASK) >> USB_OTGICR_SESSVLDEN_SHIFT)
#define USB_BRD_OTGICR_SESSVLDEN(base) (BME_UBX8(&USB_OTGICR_REG(base),
USB_OTGICR_SESSVLDEN_SHIFT, USB_OTGICR_SESSVLDEN_WIDTH))

/*! @brief Set the SESSVLDEN field to a new value. */
#define USB_WR_OTGICR_SESSVLDEN(base, value) (USB_RMW_OTGICR(base,
USB_OTGICR_SESSVLDEN_MASK, USB_OTGICR_SESSVLDEN(value)))
#define USB_BWR_OTGICR_SESSVLDEN(base, value)
(BME_BFI8(&USB_OTGICR_REG(base), ((uint8_t)(value) <<
USB_OTGICR_SESSVLDEN_SHIFT), USB_OTGICR_SESSVLDEN_SHIFT,
USB_OTGICR_SESSVLDEN_WIDTH))
/*}@ */

/*!
* @name Register USB_OTGICR, field LINESTATEEN[5] (RW)
*
* Values:
* - 0b0 - Disables the LINE_STAT_CHG interrupt.
* - 0b1 - Enables the LINE_STAT_CHG interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_OTGICR_LINESTATEEN field. */
#define USB_RD_OTGICR_LINESTATEEN(base) ((USB_OTGICR_REG(base) &
USB_OTGICR_LINESTATEEN_MASK) >> USB_OTGICR_LINESTATEEN_SHIFT)
#define USB_BRD_OTGICR_LINESTATEEN(base)
(BME_UBX8(&USB_OTGICR_REG(base), USB_OTGICR_LINESTATEEN_SHIFT,
USB_OTGICR_LINESTATEEN_WIDTH))

/*! @brief Set the LINESTATEEN field to a new value. */
#define USB_WR_OTGICR_LINESTATEEN(base, value) (USB_RMW_OTGICR(base,
USB_OTGICR_LINESTATEEN_MASK, USB_OTGICR_LINESTATEEN(value)))
#define USB_BWR_OTGICR_LINESTATEEN(base, value)
(BME_BFI8(&USB_OTGICR_REG(base), ((uint8_t)(value) <<
USB_OTGICR_LINESTATEEN_SHIFT), USB_OTGICR_LINESTATEEN_SHIFT,
USB_OTGICR_LINESTATEEN_WIDTH))
/*}@ */

/*!
* @name Register USB_OTGICR, field ONEMSECEN[6] (RW)
*
* Values:
* - 0b0 - Disables the 1ms timer interrupt.
* - 0b1 - Enables the 1ms timer interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_OTGICR_ONEMSECEN field. */
#define USB_RD_OTGICR_ONEMSECEN(base) ((USB_OTGICR_REG(base) &
USB_OTGICR_ONEMSECEN_MASK) >> USB_OTGICR_ONEMSECEN_SHIFT)

```

```

#define USB_BRD_OTGICR_ONEMSECEN(base) (BME_UBFX8(&USB_OTGICR_REG(base),  

USB_OTGICR_ONEMSECEN_SHIFT, USB_OTGICR_ONEMSECEN_WIDTH))

/*! @brief Set the ONEMSECEN field to a new value. */  

#define USB_WR_OTGICR_ONEMSECEN(base, value) (USB_RMW_OTGICR(base,  

USB_OTGICR_ONEMSECEN_MASK, USB_OTGICR_ONEMSECEN(value)))  

#define USB_BWR_OTGICR_ONEMSECEN(base, value)  

(BME_BFI8(&USB_OTGICR_REG(base), ((uint8_t)(value) <<  

USB_OTGICR_ONEMSECEN_SHIFT), USB_OTGICR_ONEMSECEN_SHIFT,  

USB_OTGICR_ONEMSECEN_WIDTH))  

/*@}*/

/*!  

 * @name Register USB_OTGICR, field IDEN[7] (RW)  

 *  

 * Values:  

 * - 0b0 - The ID interrupt is disabled  

 * - 0b1 - The ID interrupt is enabled  

 */  

/*@{*/  

/*! @brief Read current value of the USB_OTGICR_IDEN field. */  

#define USB_RD_OTGICR_IDEN(base) ((USB_OTGICR_REG(base) &  

USB_OTGICR_IDEN_MASK) >> USB_OTGICR_IDEN_SHIFT)  

#define USB_BRD_OTGICR_IDEN(base) (BME_UBFX8(&USB_OTGICR_REG(base),  

USB_OTGICR_IDEN_SHIFT, USB_OTGICR_IDEN_WIDTH))

/*! @brief Set the IDEN field to a new value. */  

#define USB_WR_OTGICR_IDEN(base, value) (USB_RMW_OTGICR(base,  

USB_OTGICR_IDEN_MASK, USB_OTGICR_IDEN(value)))  

#define USB_BWR_OTGICR_IDEN(base, value) (BME_BFI8(&USB_OTGICR_REG(base),  

((uint8_t)(value) << USB_OTGICR_IDEN_SHIFT), USB_OTGICR_IDEN_SHIFT,  

USB_OTGICR_IDEN_WIDTH))  

/*@}/
```

\*\*\*\*\*

\* USB\_OTGSTAT - OTG Status register

\*\*\*\*\*

\*/!

\* @brief USB\_OTGSTAT - OTG Status register (RW)

\* Reset value: 0x00U

\* Displays the actual value from the external comparator outputs of the  
ID pin  
\* and VBUS.

\*/

\*/!

\* @name Constants and macros for entire USB\_OTGSTAT register

\*/

/\*@{\*/

```

#define USB_RD_OTGSTAT(base)      (USB_OTGSTAT_REG(base))
#define USB_WR_OTGSTAT(base, value) (USB_OTGSTAT_REG(base) = (value))
#define USB_RMW_OTGSTAT(base, mask, value) (USB_WR_OTGSTAT(base,
(USB_RD_OTGSTAT(base) & ~mask) | (value)))
#define USB_SET_OTGSTAT(base, value) (BME_OR8(&USB_OTGSTAT_REG(base),
(uint8_t)(value)))
#define USB_CLR_OTGSTAT(base, value) (BME_AND8(&USB_OTGSTAT_REG(base),
(uint8_t)(~value)))
#define USB_TOG_OTGSTAT(base, value) (BME_XOR8(&USB_OTGSTAT_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual USB_OTGSTAT bitfields
 */

/*!!
 * @name Register USB_OTGSTAT, field AVBUSVLD[0] (RW)
 *
 * Values:
 * - 0b0 - The VBUS voltage is below the A VBUS Valid threshold.
 * - 0b1 - The VBUS voltage is above the A VBUS Valid threshold.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGSTAT_AVBUSVLD field. */
#define USB_RD_OTGSTAT_AVBUSVLD(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_AVBUSVLD_MASK) >> USB_OTGSTAT_AVBUSVLD_SHIFT)
#define USB_BRD_OTGSTAT_AVBUSVLD(base) (BME_UBFX8(&USB_OTGSTAT_REG(base),
USB_OTGSTAT_AVBUSVLD_SHIFT, USB_OTGSTAT_AVBUSVLD_WIDTH))

/*! @brief Set the AVBUSVLD field to a new value. */
#define USB_WR_OTGSTAT_AVBUSVLD(base, value) (USB_RMW_OTGSTAT(base,
USB_OTGSTAT_AVBUSVLD_MASK, USB_OTGSTAT_AVBUSVLD(value)))
#define USB_BWR_OTGSTAT_AVBUSVLD(base, value)
(BME_BFI8(&USB_OTGSTAT_REG(base), ((uint8_t)(value) <<
USB_OTGSTAT_AVBUSVLD_SHIFT), USB_OTGSTAT_AVBUSVLD_SHIFT,
USB_OTGSTAT_AVBUSVLD_WIDTH))
/*@}*/

/*!!
 * @name Register USB_OTGSTAT, field BSESEND[2] (RW)
 *
 * Values:
 * - 0b0 - The VBUS voltage is above the B session end threshold.
 * - 0b1 - The VBUS voltage is below the B session end threshold.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGSTAT_BSESEND field. */
#define USB_RD_OTGSTAT_BSESEND(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_BSESEND_MASK) >> USB_OTGSTAT_BSESEND_SHIFT)
#define USB_BRD_OTGSTAT_BSESEND(base) (BME_UBFX8(&USB_OTGSTAT_REG(base),
USB_OTGSTAT_BSESEND_SHIFT, USB_OTGSTAT_BSESEND_WIDTH))

/*! @brief Set the BSESEND field to a new value. */

```

```

#define USB_WR_OTGSTAT_BSESSEND(base, value) (USB_RMW_OTGSTAT(base,
USB_OTGSTAT_BSESSEND_MASK, USB_OTGSTAT_BSESSEND(value)))
#define USB_BWR_OTGSTAT_BSESSEND(base, value)
(BME_BFI8(&USB_OTGSTAT_REG(base), ((uint8_t)(value) <<
USB_OTGSTAT_BSESSEND_SHIFT), USB_OTGSTAT_BSESSEND_SHIFT,
USB_OTGSTAT_BSESSEND_WIDTH))
/*@}*/

/*
 * @name Register USB_OTGSTAT, field SESS_VLD[3] (RW)
 *
 * Values:
 * - 0b0 - The VBUS voltage is below the B session valid threshold
 * - 0b1 - The VBUS voltage is above the B session valid threshold.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGSTAT_SESS_VLD field. */
#define USB_RD_OTGSTAT_SESS_VLD(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_SESS_VLD_MASK) >> USB_OTGSTAT_SESS_VLD_SHIFT)
#define USB_BRD_OTGSTAT_SESS_VLD(base) (BME_UBFX8(&USB_OTGSTAT_REG(base),
USB_OTGSTAT_SESS_VLD_SHIFT, USB_OTGSTAT_SESS_VLD_WIDTH))

/*! @brief Set the SESS_VLD field to a new value. */
#define USB_WR_OTGSTAT_SESS_VLD(base, value) (USB_RMW_OTGSTAT(base,
USB_OTGSTAT_SESS_VLD_MASK, USB_OTGSTAT_SESS_VLD(value)))
#define USB_BWR_OTGSTAT_SESS_VLD(base, value)
(BME_BFI8(&USB_OTGSTAT_REG(base), ((uint8_t)(value) <<
USB_OTGSTAT_SESS_VLD_SHIFT), USB_OTGSTAT_SESS_VLD_SHIFT,
USB_OTGSTAT_SESS_VLD_WIDTH))
/*@}*/

/*
 * @name Register USB_OTGSTAT, field LINESTATESTABLE[5] (RW)
 *
 * Indicates that the internal signals that control the LINE_STATE_CHG
field of
 * OTGISTAT are stable for at least 1 millisecond. First read
LINE_STATE_CHG
 * field and then read this field. If this field reads as 1, then the
value of
 * LINE_STATE_CHG can be considered stable.
 *
 * Values:
 * - 0b0 - The LINE_STAT_CHG bit is not yet stable.
 * - 0b1 - The LINE_STAT_CHG bit has been debounced and is stable.
 */
/*@{*/
/*! @brief Read current value of the USB_OTGSTAT_LINESTATESTABLE field. */
*/
#define USB_RD_OTGSTAT_LINESTATESTABLE(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_LINESTATESTABLE_MASK) >> USB_OTGSTAT_LINESTATESTABLE_SHIFT)
#define USB_BRD_OTGSTAT_LINESTATESTABLE(base)
(BME_UBFX8(&USB_OTGSTAT_REG(base), USB_OTGSTAT_LINESTATESTABLE_SHIFT,
USB_OTGSTAT_LINESTATESTABLE_WIDTH))

```

```

/*! @brief Set the LINESTATESTABLE field to a new value. */
#define USB_WR_OTGSTAT_LINESTATESTABLE(base, value)
(USB_RMW_OTGSTAT(base, USB_OTGSTAT_LINESTATESTABLE_MASK,
USB_OTGSTAT_LINESTATESTABLE(value)))
#define USB_BWR_OTGSTAT_LINESTATESTABLE(base, value)
(BME_BFI8(&USB_OTGSTAT_REG(base), ((uint8_t)(value) <<
USB_OTGSTAT_LINESTATESTABLE_SHIFT), USB_OTGSTAT_LINESTATESTABLE_SHIFT,
USB_OTGSTAT_LINESTATESTABLE_WIDTH))
/*@}*/

/*!
 * @name Register USB_OTGSTAT, field ONEMSECEN[6] (RW)
 *
 * This bit is reserved for the 1ms count, but it is not useful to
software.
 */
/*@*/
/*! @brief Read current value of the USB_OTGSTAT_ONEMSECEN field. */
#define USB_RD_OTGSTAT_ONEMSECEN(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_ONEMSECEN_MASK) >> USB_OTGSTAT_ONEMSECEN_SHIFT)
#define USB_BRD_OTGSTAT_ONEMSECEN(base)
(BME_UBFX8(&USB_OTGSTAT_REG(base), USB_OTGSTAT_ONEMSECEN_SHIFT,
USB_OTGSTAT_ONEMSECEN_WIDTH))

/*! @brief Set the ONEMSECEN field to a new value. */
#define USB_WR_OTGSTAT_ONEMSECEN(base, value) (USB_RMW_OTGSTAT(base,
USB_OTGSTAT_ONEMSECEN_MASK, USB_OTGSTAT_ONEMSECEN(value)))
#define USB_BWR_OTGSTAT_ONEMSECEN(base, value)
(BME_BFI8(&USB_OTGSTAT_REG(base), ((uint8_t)(value) <<
USB_OTGSTAT_ONEMSECEN_SHIFT), USB_OTGSTAT_ONEMSECEN_SHIFT,
USB_OTGSTAT_ONEMSECEN_WIDTH))
/*@*/

/*!
 * @name Register USB_OTGSTAT, field ID[7] (RW)
 *
 * Indicates the current state of the ID pin on the USB connector
 *
 * Values:
 * - 0b0 - Indicates a Type A cable is plugged into the USB connector.
 * - 0b1 - Indicates no cable is attached or a Type B cable is plugged
into the
 *         USB connector.
 */
/*@*/
/*! @brief Read current value of the USB_OTGSTAT_ID field. */
#define USB_RD_OTGSTAT_ID(base) ((USB_OTGSTAT_REG(base) &
USB_OTGSTAT_ID_MASK) >> USB_OTGSTAT_ID_SHIFT)
#define USB_BRD_OTGSTAT_ID(base) (BME_UBFX8(&USB_OTGSTAT_REG(base),
USB_OTGSTAT_ID_SHIFT, USB_OTGSTAT_ID_WIDTH))

/*! @brief Set the ID field to a new value. */

```

```

#define USB_WR_OTGSTAT_ID(base, value) (USB_RMW_OTGSTAT(base,
USB_OTGSTAT_ID_MASK, USB_OTGSTAT_ID(value)))
#define USB_BWR_OTGSTAT_ID(base, value) (BME_BFI8(&USB_OTGSTAT_REG(base),
((uint8_t)(value) << USB_OTGSTAT_ID_SHIFT), USB_OTGSTAT_ID_SHIFT,
USB_OTGSTAT_ID_WIDTH))
/*@}*/

*****  

* USB_OTGCTL - OTG Control register  

*****  

/*!  

 * @brief USB_OTGCTL - OTG Control register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * Controls the operation of VBUS and Data Line termination resistors.  

 */  

/*!  

 * @name Constants and macros for entire USB_OTGCTL register  

 */  

/*@*/  

#define USB_RD_OTGCTL(base) (USB_OTGCTL_REG(base))  

#define USB_WR_OTGCTL(base, value) (USB_OTGCTL_REG(base) = (value))  

#define USB_RMW_OTGCTL(base, mask, value) (USB_WR_OTGCTL(base,  

(USB_RD_OTGCTL(base) & ~mask) | (value)))  

#define USB_SET_OTGCTL(base, value) (BME_OR8(&USB_OTGCTL_REG(base),  

(uint8_t)(value)))  

#define USB_CLR_OTGCTL(base, value) (BME_AND8(&USB_OTGCTL_REG(base),  

(uint8_t)(~value)))  

#define USB_TOG_OTGCTL(base, value) (BME_XOR8(&USB_OTGCTL_REG(base),  

(uint8_t)(value)))  

/*@*/  

/*  

 * Constants & macros for individual USB_OTGCTL bitfields  

 */  

/*!  

 * @name Register USB_OTGCTL, field OTGEN[2] (RW)  

 *  

 * Values:  

 * - 0b0 - If USB_EN is 1 and HOST_MODE is 0 in the Control Register  

(CTL), then  

 *       the D+ Data Line pull-up resistors are enabled. If HOST_MODE is 1  

the D+  

 *       and D- Data Line pull-down resistors are engaged.  

 * - 0b1 - The pull-up and pull-down controls in this register are used.  

 */  

/*@*/  

/*! @brief Read current value of the USB_OTGCTL_OTGEN field. */

```

```

#define USB_RD_OTGCTL_OTGEN(base) ((USB_OTGCTL_REG(base) &
USB_OTGCTL_OTGEN_MASK) >> USB_OTGCTL_OTGEN_SHIFT)
#define USB_BRD_OTGCTL_OTGEN(base) (BME_UBFX8(&USB_OTGCTL_REG(base),
USB_OTGCTL_OTGEN_SHIFT, USB_OTGCTL_OTGEN_WIDTH))

/*! @brief Set the OTGEN field to a new value. */
#define USB_WR_OTGCTL_OTGEN(base, value) (USB_RMW_OTGCTL(base,
USB_OTGCTL_OTGEN_MASK, USB_OTGCTL_OTGEN(value)))
#define USB_BWR_OTGCTL_OTGEN(base, value)
(BME_BFI8(&USB_OTGCTL_REG(base), ((uint8_t)(value) <<
USB_OTGCTL_OTGEN_SHIFT), USB_OTGCTL_OTGEN_SHIFT, USB_OTGCTL_OTGEN_WIDTH))
/*@}*/

/*!
* @name Register USB_OTGCTL, field DMLOW[4] (RW)
*
* Values:
* - 0b0 - D- pulldown resistor is not enabled.
* - 0b1 - D- pulldown resistor is enabled.
*/
/*@{*/
/*! @brief Read current value of the USB_OTGCTL_DMLOW field. */
#define USB_RD_OTGCTL_DMLOW(base) ((USB_OTGCTL_REG(base) &
USB_OTGCTL_DMLOW_MASK) >> USB_OTGCTL_DMLOW_SHIFT)
#define USB_BRD_OTGCTL_DMLOW(base) (BME_UBFX8(&USB_OTGCTL_REG(base),
USB_OTGCTL_DMLOW_SHIFT, USB_OTGCTL_DMLOW_WIDTH))

/*! @brief Set the DMLOW field to a new value. */
#define USB_WR_OTGCTL_DMLOW(base, value) (USB_RMW_OTGCTL(base,
USB_OTGCTL_DMLOW_MASK, USB_OTGCTL_DMLOW(value)))
#define USB_BWR_OTGCTL_DMLOW(base, value)
(BME_BFI8(&USB_OTGCTL_REG(base), ((uint8_t)(value) <<
USB_OTGCTL_DMLOW_SHIFT), USB_OTGCTL_DMLOW_SHIFT, USB_OTGCTL_DMLOW_WIDTH))
/*@}*/

/*!
* @name Register USB_OTGCTL, field DPLOW[5] (RW)
*
* This bit should always be enabled together with bit 4 (DMLOW)
*
* Values:
* - 0b0 - D+ pulldown resistor is not enabled.
* - 0b1 - D+ pulldown resistor is enabled.
*/
/*@{*/
/*! @brief Read current value of the USB_OTGCTL_DPLOW field. */
#define USB_RD_OTGCTL_DPLOW(base) ((USB_OTGCTL_REG(base) &
USB_OTGCTL_DPLOW_MASK) >> USB_OTGCTL_DPLOW_SHIFT)
#define USB_BRD_OTGCTL_DPLOW(base) (BME_UBFX8(&USB_OTGCTL_REG(base),
USB_OTGCTL_DPLOW_SHIFT, USB_OTGCTL_DPLOW_WIDTH))

/*! @brief Set the DPLOW field to a new value. */
#define USB_WR_OTGCTL_DPLOW(base, value) (USB_RMW_OTGCTL(base,
USB_OTGCTL_DPLOW_MASK, USB_OTGCTL_DPLOW(value)))

```

```

#define USB_BWR_OTGCTL_DPLOW(base, value)
(BME_BFI8(&USB_OTGCTL_REG(base), ((uint8_t)(value) <<
USB_OTGCTL_DPLOW_SHIFT), USB_OTGCTL_DPLOW_SHIFT, USB_OTGCTL_DPLOW_WIDTH))
/*@}*/

/*!!
 * @name Register USB_OTGCTL, field DPHIGH[7] (RW)
 *
 * Values:
 * - 0b0 - D+ pullup resistor is not enabled
 * - 0b1 - D+ pullup resistor is enabled
 */
/*@{*/
/*! @brief Read current value of the USB_OTGCTL_DPHIGH field. */
#define USB_RD_OTGCTL_DPHIGH(base) ((USB_OTGCTL_REG(base) &
USB_OTGCTL_DPHIGH_MASK) >> USB_OTGCTL_DPHIGH_SHIFT)
#define USB_BRD_OTGCTL_DPHIGH(base) (BME_UBFX8(&USB_OTGCTL_REG(base),
USB_OTGCTL_DPHIGH_SHIFT, USB_OTGCTL_DPHIGH_WIDTH))

/*!! @brief Set the DPHIGH field to a new value. */
#define USB_WR_OTGCTL_DPHIGH(base, value) (USB_RMW_OTGCTL(base,
USB_OTGCTL_DPHIGH_MASK, USB_OTGCTL_DPHIGH(value)))
#define USB_BWR_OTGCTL_DPHIGH(base, value)
(BME_BFI8(&USB_OTGCTL_REG(base), ((uint8_t)(value) <<
USB_OTGCTL_DPHIGH_SHIFT), USB_OTGCTL_DPHIGH_SHIFT,
USB_OTGCTL_DPHIGH_WIDTH))
/*@}*/

*****  

****  

 * USB_ISTAT - Interrupt Status register  

*****  

****/  

  

/*!!
 * @brief USB_ISTAT - Interrupt Status register (W1C)
 *
 * Reset value: 0x00U
 *
 * Contains fields for each of the interrupt sources within the USB
Module. Each
 * of these fields are qualified with their respective interrupt enable
bits.
 * All fields of this register are logically OR'd together along with the
OTG
 * Interrupt Status Register (OTGSTAT) to form a single interrupt source
for the
 * processor's interrupt controller. After an interrupt bit has been set
it may only
 * be cleared by writing a one to the respective interrupt bit. This
register
 * contains the value of 0x00 after a reset.
 */

```

```

/*!
 * @name Constants and macros for entire USB_ISTAT register
 */
/*@{*/
#define USB_RD_ISTAT(base)          (USB_ISTAT_REG(base))
#define USB_WR_ISTAT(base, value)   (USB_ISTAT_REG(base) = (value))
#define USB_RMW_ISTAT(base, mask, value) (USB_WR_ISTAT(base,
(USB_RD_ISTAT(base) & ~mask) | (value)))
#define USB_SET_ISTAT(base, value)  (BME_OR8(&USB_ISTAT_REG(base),
(uint8_t)(value)))
#define USB_CLR_ISTAT(base, value)  (BME_AND8(&USB_ISTAT_REG(base),
(uint8_t)(~(value))))
#define USB_TOG_ISTAT(base, value)  (BME_XOR8(&USB_ISTAT_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual USB_ISTAT bitfields
 */

/*!
 * @name Register USB_ISTAT, field USBRST[0] (W1C)
 *
 * This bit is set when the USB Module has decoded a valid USB reset.
 * informs the processor that it should write 0x00 into the address
 * register and
 * enable endpoint 0. USBRST is set after a USB reset has been detected
 * for 2.5
 * microseconds. It is not asserted again until the USB reset condition
 * has been
 * removed and then reasserted.
 */
/*@{*/
/*! @brief Read current value of the USB_ISTAT_USBRST field. */
#define USB_RD_ISTAT_USBRST(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_USBRST_MASK) >> USB_ISTAT_USBRST_SHIFT)
#define USB_BRD_ISTAT_USBRST(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_USBRST_SHIFT, USB_ISTAT_USBRST_WIDTH))

/*! @brief Set the USBRST field to a new value. */
#define USB_WR_ISTAT_USBRST(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK | USB_ISTAT_SOFTOK_MASK |
USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK | USB_ISTAT_RESUME_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_USBRST(value)))
#define USB_BWR_ISTAT_USBRST(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_USBRST_SHIFT), USB_ISTAT_USBRST_SHIFT,
USB_ISTAT_USBRST_WIDTH))
/*}@*/
/*
 * @name Register USB_ISTAT, field ERROR[1] (W1C)
*

```

```

 * This bit is set when any of the error conditions within Error
Interrupt
 * Status (ERRSTAT) register occur. The processor must then read the
ERRSTAT register
 * to determine the source of the error.
 */
/*@{*/
/*! @brief Read current value of the USB_ISTAT_ERROR field. */
#define USB_RD_ISTAT_ERROR(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_ERROR_MASK) >> USB_ISTAT_ERROR_SHIFT)
#define USB_BRD_ISTAT_ERROR(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_ERROR_SHIFT, USB_ISTAT_ERROR_WIDTH))

/*! @brief Set the ERROR field to a new value. */
#define USB_WR_ISTAT_ERROR(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_ERROR_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_SOFTOK_MASK |
USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK | USB_ISTAT_RESUME_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_ERROR(value)))
#define USB_BWR_ISTAT_ERROR(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_ERROR_SHIFT), USB_ISTAT_ERROR_SHIFT,
USB_ISTAT_ERROR_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_ISTAT, field SOFTOK[2] (W1C)
 *
 * This bit is set when the USB Module receives a Start Of Frame (SOF)
token. In
 * Host mode this field is set when the SOF threshold is reached, so that
 * software can prepare for the next SOF.
 */
/*@{*/
/*! @brief Read current value of the USB_ISTAT_SOFTOK field. */
#define USB_RD_ISTAT_SOFTOK(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_SOFTOK_MASK) >> USB_ISTAT_SOFTOK_SHIFT)
#define USB_BRD_ISTAT_SOFTOK(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_SOFTOK_SHIFT, USB_ISTAT_SOFTOK_WIDTH))

/*! @brief Set the SOFTOK field to a new value. */
#define USB_WR_ISTAT_SOFTOK(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_SOFTOK_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK | USB_ISTAT_RESUME_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_SOFTOK(value)))
#define USB_BWR_ISTAT_SOFTOK(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_SOFTOK_SHIFT), USB_ISTAT_SOFTOK_SHIFT,
USB_ISTAT_SOFTOK_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_ISTAT, field TOKDNE[3] (W1C)
 *
 * This bit is set when the current token being processed has completed.
The

```

```

    * processor must immediately read the STATUS (STAT) register to
determine the
    * EndPoint and BD used for this token. Clearing this bit (by writing a
one) causes
    * STAT to be cleared or the STAT holding register to be loaded into the
STAT
    * register.
*/
/*@{/*/
/*! @brief Read current value of the USB_ISTAT_TOKDNE field. */
#define USB_RD_ISTAT_TOKDNE(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_TOKDNE_MASK) >> USB_ISTAT_TOKDNE_SHIFT)
#define USB_BRD_ISTAT_TOKDNE(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_TOKDNE_SHIFT, USB_ISTAT_TOKDNE_WIDTH))

/*! @brief Set the TOKDNE field to a new value. */
#define USB_WR_ISTAT_TOKDNE(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_TOKDNE_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_SOFTOK_MASK | USB_ISTAT_SLEEP_MASK | USB_ISTAT_RESUME_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_TOKDNE(value)))
#define USB_BWR_ISTAT_TOKDNE(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_TOKDNE_SHIFT), USB_ISTAT_TOKDNE_SHIFT,
USB_ISTAT_TOKDNE_WIDTH))
/*@{/*/

/*!
 * @name Register USB_ISTAT, field SLEEP[4] (W1C)
 *
 * This bit is set when the USB Module detects a constant idle on the USB
bus
 * for 3 ms. The sleep timer is reset by activity on the USB bus.
*/
/*@{/*/
/*! @brief Read current value of the USB_ISTAT_SLEEP field. */
#define USB_RD_ISTAT_SLEEP(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_SLEEP_MASK) >> USB_ISTAT_SLEEP_SHIFT)
#define USB_BRD_ISTAT_SLEEP(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_SLEEP_SHIFT, USB_ISTAT_SLEEP_WIDTH))

/*! @brief Set the SLEEP field to a new value. */
#define USB_WR_ISTAT_SLEEP(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_SLEEP_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_SOFTOK_MASK | USB_ISTAT_TOKDNE_MASK | USB_ISTAT_RESUME_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_SLEEP(value)))
#define USB_BWR_ISTAT_SLEEP(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_SLEEP_SHIFT), USB_ISTAT_SLEEP_SHIFT,
USB_ISTAT_SLEEP_WIDTH))
/*@{/*/

/*!
 * @name Register USB_ISTAT, field RESUME[5] (W1C)
 *
 * This bit is set depending upon the DP/DM signals, and can be used to
signal

```

```

 * remote wake-up signaling on the USB bus. When not in suspend mode this
 * interrupt must be disabled.
 */
/*@{*/
/*! @brief Read current value of the USB_ISTAT_RESUME field. */
#define USB_RD_ISTAT_RESUME(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_RESUME_MASK) >> USB_ISTAT_RESUME_SHIFT)
#define USB_BRD_ISTAT_RESUME(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_RESUME_SHIFT, USB_ISTAT_RESUME_WIDTH))

/*! @brief Set the RESUME field to a new value. */
#define USB_WR_ISTAT_RESUME(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_RESUME_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_SOFTOK_MASK | USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK |
USB_ISTAT_ATTACH_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_RESUME(value)))
#define USB_BWR_ISTAT_RESUME(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_RESUME_SHIFT), USB_ISTAT_RESUME_SHIFT,
USB_ISTAT_RESUME_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_ISTAT, field ATTACH[6] (W1C)
 *
 * This bit is set when the USB Module detects an attach of a USB device.
 * signal is only valid if HOSTMODEEN is true. This interrupt signifies
 * that a peripheral is now present and must be configured.
 */
/*@{*/
/*! @brief Read current value of the USB_ISTAT_ATTACH field. */
#define USB_RD_ISTAT_ATTACH(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_ATTACH_MASK) >> USB_ISTAT_ATTACH_SHIFT)
#define USB_BRD_ISTAT_ATTACH(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_ATTACH_SHIFT, USB_ISTAT_ATTACH_WIDTH))

/*! @brief Set the ATTACH field to a new value. */
#define USB_WR_ISTAT_ATTACH(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_ATTACH_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_SOFTOK_MASK | USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK |
USB_ISTAT_RESUME_MASK | USB_ISTAT_STALL_MASK), USB_ISTAT_ATTACH(value)))
#define USB_BWR_ISTAT_ATTACH(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_ATTACH_SHIFT), USB_ISTAT_ATTACH_SHIFT,
USB_ISTAT_ATTACH_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_ISTAT, field STALL[7] (W1C)
 *
 * In Target mode this bit is asserted when a STALL handshake is sent by
 * the SIE. In Host mode this bit is set when the USB Module detects a STALL
 * acknowledge

```

```

    * during the handshake phase of a USB transaction. This interrupt can be
used to
    * determine whether the last USB transaction was completed successfully
or
    * stalled.
*/
/*@{ */
/*! @brief Read current value of the USB_ISTAT_STALL field. */
#define USB_RD_ISTAT_STALL(base) ((USB_ISTAT_REG(base) &
USB_ISTAT_STALL_MASK) >> USB_ISTAT_STALL_SHIFT)
#define USB_BRD_ISTAT_STALL(base) (BME_UBFX8(&USB_ISTAT_REG(base),
USB_ISTAT_STALL_SHIFT, USB_ISTAT_STALL_WIDTH))

/*! @brief Set the STALL field to a new value. */
#define USB_WR_ISTAT_STALL(base, value) (USB_RMW_ISTAT(base,
(USB_ISTAT_STALL_MASK | USB_ISTAT_USBRST_MASK | USB_ISTAT_ERROR_MASK |
USB_ISTAT_SOFTOK_MASK | USB_ISTAT_TOKDNE_MASK | USB_ISTAT_SLEEP_MASK |
USB_ISTAT_RESUME_MASK | USB_ISTAT_ATTACH_MASK), USB_ISTAT_STALL(value)))
#define USB_BWR_ISTAT_STALL(base, value) (BME_BFI8(&USB_ISTAT_REG(base),
((uint8_t)(value) << USB_ISTAT_STALL_SHIFT), USB_ISTAT_STALL_SHIFT,
USB_ISTAT_STALL_WIDTH))
/*}@*/ */

/********************* USB_INTEN - Interrupt Enable register *****************/
***** */
    * USB_INTEN - Interrupt Enable register
***** */

/*!
    * @brief USB_INTEN - Interrupt Enable register (RW)
    *
    * Reset value: 0x00U
    *
    * Contains enable fields for each of the interrupt sources within the
USB
    * Module. Setting any of these bits enables the respective interrupt
source in the
    * ISTAT register. This register contains the value of 0x00 after a
reset.
    */
/*!
    * @name Constants and macros for entire USB_INTEN register
    */
/*@{ */
#define USB_RD_INTEN(base)      (USB_INTEN_REG(base))
#define USB_WR_INTEN(base, value) (USB_INTEN_REG(base) = (value))
#define USB_RMW_INTEN(base, mask, value) (USB_WR_INTEN(base,
(USB_RD_INTEN(base) & ~mask)) | (value)))
#define USB_SET_INTEN(base, value) (BME_OR8(&USB_INTEN_REG(base),
(uint8_t)(value)))
#define USB_CLR_INTEN(base, value) (BME_AND8(&USB_INTEN_REG(base),
(uint8_t)(~(value)))))


```

```

#define USB_TOG_INTEN(base, value) (BME_XOR8(&USB_INTEN_REG(base),  

    (uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual USB_INTEN bitfields
 */

/*!  

 * @name Register USB_INTEN, field USBRSTEN[0] (RW)
 *  

 * Values:  

 * - 0b0 - Disables the USBRST interrupt.  

 * - 0b1 - Enables the USBRST interrupt.
 */
/*@{*/  

/*! @brief Read current value of the USB_INTEN_USBRSTEN field. */  

#define USB_RD_INTEN_USBRSTEN(base) ((USB_INTEN_REG(base) &  

    USB_INTEN_USBRSTEN_MASK) >> USB_INTEN_USBRSTEN_SHIFT)  

#define USB_BRD_INTEN_USBRSTEN(base) (BME_UBFX8(&USB_INTEN_REG(base),  

    USB_INTEN_USBRSTEN_SHIFT, USB_INTEN_USBRSTEN_WIDTH))  

/*! @brief Set the USBRSTEN field to a new value. */  

#define USB_WR_INTEN_USBRSTEN(base, value) (USB_RMW_INTEN(base,  

    USB_INTEN_USBRSTEN_MASK, USB_INTEN_USBRSTEN(value)))  

#define USB_BWR_INTEN_USBRSTEN(base, value)  

(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<  

    USB_INTEN_USBRSTEN_SHIFT), USB_INTEN_USBRSTEN_SHIFT,  

    USB_INTEN_USBRSTEN_WIDTH))  

/*@}*/

/*!  

 * @name Register USB_INTEN, field ERRORREN[1] (RW)
 *  

 * Values:  

 * - 0b0 - Disables the ERROR interrupt.  

 * - 0b1 - Enables the ERROR interrupt.
 */
/*@{*/  

/*! @brief Read current value of the USB_INTEN_ERRORREN field. */  

#define USB_RD_INTEN_ERRORREN(base) ((USB_INTEN_REG(base) &  

    USB_INTEN_ERRORREN_MASK) >> USB_INTEN_ERRORREN_SHIFT)  

#define USB_BRD_INTEN_ERRORREN(base) (BME_UBFX8(&USB_INTEN_REG(base),  

    USB_INTEN_ERRORREN_SHIFT, USB_INTEN_ERRORREN_WIDTH))  

/*! @brief Set the ERRORREN field to a new value. */  

#define USB_WR_INTEN_ERRORREN(base, value) (USB_RMW_INTEN(base,  

    USB_INTEN_ERRORREN_MASK, USB_INTEN_ERRORREN(value)))  

#define USB_BWR_INTEN_ERRORREN(base, value)  

(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<  

    USB_INTEN_ERRORREN_SHIFT), USB_INTEN_ERRORREN_SHIFT,  

    USB_INTEN_ERRORREN_WIDTH))  

/*@}*/

```

```

/*!
 * @name Register USB_INTEN, field SOFTOKEN[2] (RW)
 *
 * Values:
 * - 0b0 - Disables the SOFTOK interrupt.
 * - 0b1 - Enables the SOFTOK interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_INTEN_SOFTOKEN field. */
#define USB_RD_INTEN_SOFTOKEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_SOFTOKEN_MASK) >> USB_INTEN_SOFTOKEN_SHIFT)
#define USB_BRD_INTEN_SOFTOKEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_SOFTOKEN_SHIFT, USB_INTEN_SOFTOKEN_WIDTH))

/*! @brief Set the SOFTOKEN field to a new value. */
#define USB_WR_INTEN_SOFTOKEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_SOFTOKEN_MASK, USB_INTEN_SOFTOKEN(value)))
#define USB_BWR_INTEN_SOFTOKEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_SOFTOKEN_SHIFT), USB_INTEN_SOFTOKEN_SHIFT,
USB_INTEN_SOFTOKEN_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_INTEN, field TOKDNEEN[3] (RW)
 *
 * Values:
 * - 0b0 - Disables the TOKDNE interrupt.
 * - 0b1 - Enables the TOKDNE interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_INTEN_TOKDNEEN field. */
#define USB_RD_INTEN_TOKDNEEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_TOKDNEEN_MASK) >> USB_INTEN_TOKDNEEN_SHIFT)
#define USB_BRD_INTEN_TOKDNEEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_TOKDNEEN_SHIFT, USB_INTEN_TOKDNEEN_WIDTH))

/*! @brief Set the TOKDNEEN field to a new value. */
#define USB_WR_INTEN_TOKDNEEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_TOKDNEEN_MASK, USB_INTEN_TOKDNEEN(value)))
#define USB_BWR_INTEN_TOKDNEEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_TOKDNEEN_SHIFT), USB_INTEN_TOKDNEEN_SHIFT,
USB_INTEN_TOKDNEEN_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_INTEN, field SLEEPEN[4] (RW)
 *
 * Values:
 * - 0b0 - Disables the SLEEP interrupt.
 * - 0b1 - Enables the SLEEP interrupt.
 */
/*@{*/

```

```

/*! @brief Read current value of the USB_INTEN_SLEEPEN field. */
#define USB_RD_INTEN_SLEEPEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_SLEEPEN_MASK) >> USB_INTEN_SLEEPEN_SHIFT)
#define USB_BRD_INTEN_SLEEPEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_SLEEPEN_SHIFT, USB_INTEN_SLEEPEN_WIDTH))

/*! @brief Set the SLEEPEN field to a new value. */
#define USB_WR_INTEN_SLEEPEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_SLEEPEN_MASK, USB_INTEN_SLEEPEN(value)))
#define USB_BWR_INTEN_SLEEPEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_SLEEPEN_SHIFT), USB_INTEN_SLEEPEN_SHIFT,
USB_INTEN_SLEEPEN_WIDTH))
/*@}*/

/*
 * @name Register USB_INTEN, field RESUMEEN[5] (RW)
 *
 * Values:
 * - 0b0 - Disables the RESUME interrupt.
 * - 0b1 - Enables the RESUME interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_INTEN_RESUMEEN field. */
#define USB_RD_INTEN_RESUMEEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_RESUMEEN_MASK) >> USB_INTEN_RESUMEEN_SHIFT)
#define USB_BRD_INTEN_RESUMEEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_RESUMEEN_SHIFT, USB_INTEN_RESUMEEN_WIDTH))

/*! @brief Set the RESUMEEN field to a new value. */
#define USB_WR_INTEN_RESUMEEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_RESUMEEN_MASK, USB_INTEN_RESUMEEN(value)))
#define USB_BWR_INTEN_RESUMEEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_RESUMEEN_SHIFT), USB_INTEN_RESUMEEN_SHIFT,
USB_INTEN_RESUMEEN_WIDTH))
/*@}*/

/*
 * @name Register USB_INTEN, field ATTACHEN[6] (RW)
 *
 * Values:
 * - 0b0 - Disables the ATTACH interrupt.
 * - 0b1 - Enables the ATTACH interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_INTEN_ATTACHEN field. */
#define USB_RD_INTEN_ATTACHEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_ATTACHEN_MASK) >> USB_INTEN_ATTACHEN_SHIFT)
#define USB_BRD_INTEN_ATTACHEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_ATTACHEN_SHIFT, USB_INTEN_ATTACHEN_WIDTH))

/*! @brief Set the ATTACHEN field to a new value. */

```

```

#define USB_WR_INTEN_ATTACHEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_ATTACHEN_MASK, USB_INTEN_ATTACHEN(value)))
#define USB_BWR_INTEN_ATTACHEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_ATTACHEN_SHIFT), USB_INTEN_ATTACHEN_SHIFT,
USB_INTEN_ATTACHEN_WIDTH))
/*@}*/

/*
 * @name Register USB_INTEN, field STALLEN[7] (RW)
 *
 * Values:
 * - 0b0 - Disables the STALL interrupt.
 * - 0b1 - Enables the STALL interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_INTEN_STALLEN field. */
#define USB_RD_INTEN_STALLEN(base) ((USB_INTEN_REG(base) &
USB_INTEN_STALLEN_MASK) >> USB_INTEN_STALLEN_SHIFT)
#define USB_BRD_INTEN_STALLEN(base) (BME_UBFX8(&USB_INTEN_REG(base),
USB_INTEN_STALLEN_SHIFT, USB_INTEN_STALLEN_WIDTH))

/*! @brief Set the STALLEN field to a new value. */
#define USB_WR_INTEN_STALLEN(base, value) (USB_RMW_INTEN(base,
USB_INTEN_STALLEN_MASK, USB_INTEN_STALLEN(value)))
#define USB_BWR_INTEN_STALLEN(base, value)
(BME_BFI8(&USB_INTEN_REG(base), ((uint8_t)(value) <<
USB_INTEN_STALLEN_SHIFT), USB_INTEN_STALLEN_SHIFT,
USB_INTEN_STALLEN_WIDTH))
/*@}*/

*****
* USB_ERRSTAT - Error Interrupt Status register
*****
*/!

/*!
 * @brief USB_ERRSTAT - Error Interrupt Status register (RW)
 *
 * Reset value: 0x00U
 *
 * Contains enable bits for each of the error sources within the USB
Module.
 * Each of these bits are qualified with their respective error enable
bits. All
 * bits of this register are logically OR'd together and the result
placed in the
 * ERROR bit of the ISTAT register. After an interrupt bit has been set
it may only
 * be cleared by writing a one to the respective interrupt bit. Each bit
is set

```

```

    * as soon as the error conditions is detected. Therefore, the interrupt
does not
    * typically correspond with the end of a token being processed. This
register
    * contains the value of 0x00 after a reset.
*/
/*!
    * @name Constants and macros for entire USB_ERRSTAT register
*/
/*@{*/
#define USB_RD_ERRSTAT(base)      (USB_ERRSTAT_REG(base))
#define USB_WR_ERRSTAT(base, value) (USB_ERRSTAT_REG(base) = (value))
#define USB_RMW_ERRSTAT(base, mask, value) (USB_WR_ERRSTAT(base,
(USB_RD_ERRSTAT(base) & ~mask) | (value)))
#define USB_SET_ERRSTAT(base, value) (BME_OR8(&USB_ERRSTAT_REG(base),
(uint8_t)(value)))
#define USB_CLR_ERRSTAT(base, value) (BME_AND8(&USB_ERRSTAT_REG(base),
(uint8_t)(~value)))
#define USB_TOG_ERRSTAT(base, value) (BME_XOR8(&USB_ERRSTAT_REG(base),
(uint8_t)(value)))
/*@}*/
/*
    * Constants & macros for individual USB_ERRSTAT bitfields
*/
/*!
    * @name Register USB_ERRSTAT, field PIDERR[0] (W1C)
*
    * This bit is set when the PID check field fails.
*/
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_PIDERR field. */
#define USB_RD_ERRSTAT_PIDERR(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_PIDERR_MASK) >> USB_ERRSTAT_PIDERR_SHIFT)
#define USB_BRD_ERRSTAT_PIDERR(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_PIDERR_SHIFT, USB_ERRSTAT_PIDERR_WIDTH))

/*! @brief Set the PIDERR field to a new value. */
#define USB_WR_ERRSTAT_PIDERR(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_PIDERR_MASK | USB_ERRSTAT_CRC5EOF_MASK |
USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_DFN8_MASK | USB_ERRSTAT_BTOERR_MASK |
USB_ERRSTAT_DMAERR_MASK | USB_ERRSTAT_BTERR_MASK),
USB_ERRSTAT_PIDERR(value)))
#define USB_BWR_ERRSTAT_PIDERR(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_PIDERR_SHIFT), USB_ERRSTAT_PIDERR_SHIFT,
USB_ERRSTAT_PIDERR_WIDTH))
/*@}*/
/*
    * @name Register USB_ERRSTAT, field CRC5EOF[1] (W1C)
*/

```

```

    * This error interrupt has two functions. When the USB Module is
operating in
    * peripheral mode (HOSTMODEEN=0), this interrupt detects CRC5 errors in
the token
    * packets generated by the host. If set the token packet was rejected
due to a
    * CRC5 error. When the USB Module is operating in host mode
(HOSTMODEEN=1), this
    * interrupt detects End Of Frame (EOF) error conditions. This occurs
when the
    * USB Module is transmitting or receiving data and the SOF counter
reaches zero.
    * This interrupt is useful when developing USB packet scheduling
software to
    * ensure that no USB transactions cross the start of the next frame.
    */
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_CRC5EOF field. */
#define USB_RD_ERRSTAT_CRC5EOF(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_CRC5EOF_MASK) >> USB_ERRSTAT_CRC5EOF_SHIFT)
#define USB_BRD_ERRSTAT_CRC5EOF(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_CRC5EOF_SHIFT, USB_ERRSTAT_CRC5EOF_WIDTH))

/*! @brief Set the CRC5EOF field to a new value. */
#define USB_WR_ERRSTAT_CRC5EOF(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_DFN8_MASK | USB_ERRSTAT_BTOERR_MASK
| USB_ERRSTAT_DMAERR_MASK | USB_ERRSTAT_BTSERR_MASK),
USB_ERRSTAT_CRC5EOF(value)))
#define USB_BWR_ERRSTAT_CRC5EOF(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_CRC5EOF_SHIFT), USB_ERRSTAT_CRC5EOF_SHIFT,
USB_ERRSTAT_CRC5EOF_WIDTH))
/*@}*/
/*!
 * @name Register USB_ERRSTAT, field CRC16[2] (W1C)
 *
 * This bit is set when a data packet is rejected due to a CRC16 error.
 */
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_CRC16 field. */
#define USB_RD_ERRSTAT_CRC16(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_CRC16_MASK) >> USB_ERRSTAT_CRC16_SHIFT)
#define USB_BRD_ERRSTAT_CRC16(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_CRC16_SHIFT, USB_ERRSTAT_CRC16_WIDTH))

/*! @brief Set the CRC16 field to a new value. */
#define USB_WR_ERRSTAT_CRC16(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_DFN8_MASK |
USB_ERRSTAT_BTOERR_MASK | USB_ERRSTAT_DMAERR_MASK |
USB_ERRSTAT_BTSERR_MASK), USB_ERRSTAT_CRC16(value)))

```

```

#define USB_BWR_ERRSTAT_CRC16(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_CRC16_SHIFT), USB_ERRSTAT_CRC16_SHIFT,
USB_ERRSTAT_CRC16_WIDTH))
/*@}*/

/*
 * @name Register USB_ERRSTAT, field DFN8[3] (W1C)
 *
 * This bit is set if the data field received was not 8 bits in length.
USB
 * Specification 1.0 requires that data fields be an integral number of
bytes. If the
 * data field was not an integral number of bytes, this bit is set.
 */
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_DFN8 field. */
#define USB_RD_ERRSTAT_DFN8(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_DFN8_MASK) >> USB_ERRSTAT_DFN8_SHIFT)
#define USB_BRD_ERRSTAT_DFN8(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_DFN8_SHIFT, USB_ERRSTAT_DFN8_WIDTH))

/*! @brief Set the DFN8 field to a new value. */
#define USB_WR_ERRSTAT_DFN8(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_DFN8_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_CRC16_MASK |
USB_ERRSTAT_BTOERR_MASK | USB_ERRSTAT_DMAERR_MASK |
USB_ERRSTAT_BTSERR_MASK), USB_ERRSTAT_DFN8(value)))
#define USB_BWR_ERRSTAT_DFN8(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_DFN8_SHIFT), USB_ERRSTAT_DFN8_SHIFT, USB_ERRSTAT_DFN8_WIDTH))
/*@}*/

/*
 * @name Register USB_ERRSTAT, field BTOERR[4] (W1C)
 *
 * This bit is set when a bus turnaround timeout error occurs. The USB
module
 * contains a bus turnaround timer that keeps track of the amount of time
elapsed
 * between the token and data phases of a SETUP or OUT TOKEN or the data
and
 * handshake phases of a IN TOKEN. If more than 16 bit times are counted
from the
 * previous EOP before a transition from IDLE, a bus turnaround timeout
error occurs.
 */
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_BTOERR field. */
#define USB_RD_ERRSTAT_BTOERR(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_BTOERR_MASK) >> USB_ERRSTAT_BTOERR_SHIFT)
#define USB_BRD_ERRSTAT_BTOERR(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_BTOERR_SHIFT, USB_ERRSTAT_BTOERR_WIDTH))

```

```

/*! @brief Set the BTOERR field to a new value. */
#define USB_WR_ERRSTAT_BTOERR(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_BTOERR_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_DFN8_MASK
| USB_ERRSTAT_DMAERR_MASK | USB_ERRSTAT_BTSERR_MASK),
USB_ERRSTAT_BTOERR(value)))
#define USB_BWR_ERRSTAT_BTOERR(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_BTOERR_SHIFT), USB_ERRSTAT_BTOERR_SHIFT,
USB_ERRSTAT_BTOERR_WIDTH))
/*@}*/

/*
 * @name Register USB_ERRSTAT, field DMAERR[5] (W1C)
 *
 * This bit is set if the USB Module has requested a DMA access to read a
new
 * BDT but has not been given the bus before it needs to receive or
transmit data.
 * If processing a TX transfer this would cause a transmit data underflow
 * condition. If processing a RX transfer this would cause a receive data
overflow
 * condition. This interrupt is useful when developing device arbitration
hardware for
 * the microprocessor and the USB module to minimize bus request and bus
grant
 * latency. This bit is also set if a data packet to or from the host is
larger
 * than the buffer size allocated in the BDT. In this case the data
packet is
 * truncated as it is put in buffer memory.
 */
/*@*/
/*! @brief Read current value of the USB_ERRSTAT_DMAERR field. */
#define USB_RD_ERRSTAT_DMAERR(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_DMAERR_MASK) >> USB_ERRSTAT_DMAERR_SHIFT)
#define USB_BRD_ERRSTAT_DMAERR(base) (BME_UBFX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_DMAERR_SHIFT, USB_ERRSTAT_DMAERR_WIDTH))

/*! @brief Set the DMAERR field to a new value. */
#define USB_WR_ERRSTAT_DMAERR(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_DMAERR_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_DFN8_MASK
| USB_ERRSTAT_BTOERR_MASK | USB_ERRSTAT_BTSERR_MASK),
USB_ERRSTAT_DMAERR(value)))
#define USB_BWR_ERRSTAT_DMAERR(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_DMAERR_SHIFT), USB_ERRSTAT_DMAERR_SHIFT,
USB_ERRSTAT_DMAERR_WIDTH))
/*@*/

/*
 * @name Register USB_ERRSTAT, field BTSERR[7] (W1C)
*

```

```

    * This bit is set when a bit stuff error is detected. If set, the
corresponding
    * packet is rejected due to the error.
    */
/*@{*/
/*! @brief Read current value of the USB_ERRSTAT_BTERR field. */
#define USB_RD_ERRSTAT_BTERR(base) ((USB_ERRSTAT_REG(base) &
USB_ERRSTAT_BTERR_MASK) >> USB_ERRSTAT_BTERR_SHIFT)
#define USB_BRD_ERRSTAT_BTERR(base) (BME_UBX8(&USB_ERRSTAT_REG(base),
USB_ERRSTAT_BTERR_SHIFT, USB_ERRSTAT_BTERR_WIDTH))

/*! @brief Set the BTERR field to a new value. */
#define USB_WR_ERRSTAT_BTERR(base, value) (USB_RMW_ERRSTAT(base,
(USB_ERRSTAT_BTERR_MASK | USB_ERRSTAT_PIDERR_MASK |
USB_ERRSTAT_CRC5EOF_MASK | USB_ERRSTAT_CRC16_MASK | USB_ERRSTAT_DFN8_MASK
| USB_ERRSTAT_BTOERR_MASK | USB_ERRSTAT_DMAERR_MASK),
USB_ERRSTAT_BTERR(value)))
#define USB_BWR_ERRSTAT_BTERR(base, value)
(BME_BFI8(&USB_ERRSTAT_REG(base), ((uint8_t)(value) <<
USB_ERRSTAT_BTERR_SHIFT), USB_ERRSTAT_BTERR_SHIFT,
USB_ERRSTAT_BTERR_WIDTH))
/*@}*/

/*****
*****  

* USB_ERREN - Error Interrupt Enable register  

*****  

*****/  

/*!  

 * @brief USB_ERREN - Error Interrupt Enable register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * Contains enable bits for each of the error interrupt sources within  

the USB  

 * module. Setting any of these bits enables the respective interrupt  

source in  

 * ERRSTAT. Each bit is set as soon as the error conditions is detected.  

Therefore,  

 * the interrupt does not typically correspond with the end of a token  

being  

 * processed. This register contains the value of 0x00 after a reset.  

*/  

/*!  

 * @name Constants and macros for entire USB_ERREN register  

*/  

/*@{*/
#define USB_RD_ERREN(base) (USB_ERREN_REG(base))
#define USB_WR_ERREN(base, value) (USB_ERREN_REG(base) = (value))
#define USB_RMW_ERREN(base, mask, value) (USB_WR_ERREN(base,
(USB_RD_ERREN(base) & ~mask) | (value)))

```

```

#define USB_SET_ERREN(base, value) (BME_OR8 (&USB_ERREN_REG(base),  

(uint8_t)(value)))  

#define USB_CLR_ERREN(base, value) (BME_AND8 (&USB_ERREN_REG(base),  

(uint8_t)(~(value))))  

#define USB_TOG_ERREN(base, value) (BME_XOR8 (&USB_ERREN_REG(base),  

(uint8_t)(value)))  

/*@}*/

/*
 * Constants & macros for individual USB_ERREN bitfields
 */

/*!
 * @name Register USB_ERREN, field PIDERREN[0] (RW)
 *
 * Values:
 * - 0b0 - Disables the PIDERR interrupt.
 * - 0b1 - Enters the PIDERR interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_ERREN_PIDERREN field. */  

#define USB_RD_ERREN_PIDERREN(base) ((USB_ERREN_REG(base) &  

USB_ERREN_PIDERREN_MASK) >> USB_ERREN_PIDERREN_SHIFT)  

#define USB_BRD_ERREN_PIDERREN(base) (BME_UBFX8(&USB_ERREN_REG(base),  

USB_ERREN_PIDERREN_SHIFT, USB_ERREN_PIDERREN_WIDTH))  

/*! @brief Set the PIDERREN field to a new value. */  

#define USB_WR_ERREN_PIDERREN(base, value) (USB_RMW_ERREN(base,  

USB_ERREN_PIDERREN_MASK, USB_ERREN_PIDERREN(value)))  

#define USB_BWR_ERREN_PIDERREN(base, value)  

(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<  

USB_ERREN_PIDERREN_SHIFT), USB_ERREN_PIDERREN_SHIFT,  

USB_ERREN_PIDERREN_WIDTH))  

/*@}*/

/*!
 * @name Register USB_ERREN, field CRC5EOFEN[1] (RW)
 *
 * Values:
 * - 0b0 - Disables the CRC5/EOF interrupt.
 * - 0b1 - Enables the CRC5/EOF interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_ERREN_CRC5EOFEN field. */  

#define USB_RD_ERREN_CRC5EOFEN(base) ((USB_ERREN_REG(base) &  

USB_ERREN_CRC5EOFEN_MASK) >> USB_ERREN_CRC5EOFEN_SHIFT)  

#define USB_BRD_ERREN_CRC5EOFEN(base) (BME_UBFX8(&USB_ERREN_REG(base),  

USB_ERREN_CRC5EOFEN_SHIFT, USB_ERREN_CRC5EOFEN_WIDTH))  

/*! @brief Set the CRC5EOFEN field to a new value. */  

#define USB_WR_ERREN_CRC5EOFEN(base, value) (USB_RMW_ERREN(base,  

USB_ERREN_CRC5EOFEN_MASK, USB_ERREN_CRC5EOFEN(value)))  

#define USB_BWR_ERREN_CRC5EOFEN(base, value)  

(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<

```

```

USB_ERREN_CRC5EOFEN_SHIFT), USB_ERREN_CRC5EOFEN_SHIFT,
USB_ERREN_CRC5EOFEN_WIDTH))
/*@}*/

/*!
 * @name Register USB_ERREN, field CRC16EN[2] (RW)
 *
 * Values:
 * - 0b0 - Disables the CRC16 interrupt.
 * - 0b1 - Enables the CRC16 interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_ERREN_CRC16EN field. */
#define USB_RD_ERREN_CRC16EN(base) ((USB_ERREN_REG(base) &
USB_ERREN_CRC16EN_MASK) >> USB_ERREN_CRC16EN_SHIFT)
#define USB_BRD_ERREN_CRC16EN(base) (BME_UBXF8(&USB_ERREN_REG(base),
USB_ERREN_CRC16EN_SHIFT, USB_ERREN_CRC16EN_WIDTH))

/*! @brief Set the CRC16EN field to a new value. */
#define USB_WR_ERREN_CRC16EN(base, value) (USB_RMW_ERREN(base,
USB_ERREN_CRC16EN_MASK, USB_ERREN_CRC16EN(value)))
#define USB_BWR_ERREN_CRC16EN(base, value)
(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<
USB_ERREN_CRC16EN_SHIFT), USB_ERREN_CRC16EN_SHIFT,
USB_ERREN_CRC16EN_WIDTH))
/*@}*/

/*!
 * @name Register USB_ERREN, field DFN8EN[3] (RW)
 *
 * Values:
 * - 0b0 - Disables the DFN8 interrupt.
 * - 0b1 - Enables the DFN8 interrupt.
 */
/*@{*/
/*! @brief Read current value of the USB_ERREN_DFN8EN field. */
#define USB_RD_ERREN_DFN8EN(base) ((USB_ERREN_REG(base) &
USB_ERREN_DFN8EN_MASK) >> USB_ERREN_DFN8EN_SHIFT)
#define USB_BRD_ERREN_DFN8EN(base) (BME_UBXF8(&USB_ERREN_REG(base),
USB_ERREN_DFN8EN_SHIFT, USB_ERREN_DFN8EN_WIDTH))

/*! @brief Set the DFN8EN field to a new value. */
#define USB_WR_ERREN_DFN8EN(base, value) (USB_RMW_ERREN(base,
USB_ERREN_DFN8EN_MASK, USB_ERREN_DFN8EN(value)))
#define USB_BWR_ERREN_DFN8EN(base, value) (BME_BFI8(&USB_ERREN_REG(base),
((uint8_t)(value) << USB_ERREN_DFN8EN_SHIFT), USB_ERREN_DFN8EN_SHIFT,
USB_ERREN_DFN8EN_WIDTH))
/*@}*/

/*!
 * @name Register USB_ERREN, field BTOERREN[4] (RW)
 *
 * Values:
 * - 0b0 - Disables the BTOERR interrupt.
 */

```

```

* - 0b1 - Enables the BTOERR interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_ERREN_BTOERREN field. */
#define USB_RD_ERREN_BTOERREN(base) ((USB_ERREN_REG(base) &
USB_ERREN_BTOERREN_MASK) >> USB_ERREN_BTOERREN_SHIFT)
#define USB_BRD_ERREN_BTOERREN(base) (BME_UBFX8(&USB_ERREN_REG(base),
USB_ERREN_BTOERREN_SHIFT, USB_ERREN_BTOERREN_WIDTH))

/*! @brief Set the BTOERREN field to a new value. */
#define USB_WR_ERREN_BTOERREN(base, value) (USB_RMW_ERREN(base,
USB_ERREN_BTOERREN_MASK, USB_ERREN_BTOERREN(value)))
#define USB_BWR_ERREN_BTOERREN(base, value)
(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<
USB_ERREN_BTOERREN_SHIFT), USB_ERREN_BTOERREN_SHIFT,
USB_ERREN_BTOERREN_WIDTH))
/*}@ */

/*!
* @name Register USB_ERREN, field DMAERREN[5] (RW)
*
* Values:
* - 0b0 - Disables the DMAERR interrupt.
* - 0b1 - Enables the DMAERR interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_ERREN_DMAERREN field. */
#define USB_RD_ERREN_DMAERREN(base) ((USB_ERREN_REG(base) &
USB_ERREN_DMAERREN_MASK) >> USB_ERREN_DMAERREN_SHIFT)
#define USB_BRD_ERREN_DMAERREN(base) (BME_UBFX8(&USB_ERREN_REG(base),
USB_ERREN_DMAERREN_SHIFT, USB_ERREN_DMAERREN_WIDTH))

/*! @brief Set the DMAERREN field to a new value. */
#define USB_WR_ERREN_DMAERREN(base, value) (USB_RMW_ERREN(base,
USB_ERREN_DMAERREN_MASK, USB_ERREN_DMAERREN(value)))
#define USB_BWR_ERREN_DMAERREN(base, value)
(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<
USB_ERREN_DMAERREN_SHIFT), USB_ERREN_DMAERREN_SHIFT,
USB_ERREN_DMAERREN_WIDTH))
/*}@ */

/*!
* @name Register USB_ERREN, field BTSERREN[7] (RW)
*
* Values:
* - 0b0 - Disables the BTSERR interrupt.
* - 0b1 - Enables the BTSERR interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_ERREN_BTSERREN field. */
#define USB_RD_ERREN_BTSERREN(base) ((USB_ERREN_REG(base) &
USB_ERREN_BTSERREN_MASK) >> USB_ERREN_BTSERREN_SHIFT)
#define USB_BRD_ERREN_BTSERREN(base) (BME_UBFX8(&USB_ERREN_REG(base),
USB_ERREN_BTSERREN_SHIFT, USB_ERREN_BTSERREN_WIDTH))

```

```

/*! @brief Set the BTSERREN field to a new value. */
#define USB_WR_ERREN_BTSERREN(base, value) (USB_RMW_ERREN(base,
USB_ERREN_BTSERREN_MASK, USB_ERREN_BTSERREN(value)))
#define USB_BWR_ERREN_BTSERREN(base, value)
(BME_BFI8(&USB_ERREN_REG(base), ((uint8_t)(value) <<
USB_ERREN_BTSERREN_SHIFT), USB_ERREN_BTSERREN_SHIFT,
USB_ERREN_BTSERREN_WIDTH))
/*@}*/

/********************* USB_STAT - Status register ********************
****

* USB_STAT - Status register

***** */

/*!
* @brief USB_STAT - Status register (RO)
*
* Reset value: 0x00U
*
* Reports the transaction status within the USB module. When the
processor's
* interrupt controller has received a TOKDNE, interrupt the Status
Register must
* be read to determine the status of the previous endpoint
communication. The
* data in the status register is valid when TOKDNE interrupt is
asserted. The
* Status register is actually a read window into a status FIFO
maintained by the USB
* module. When the USB module uses a BD, it updates the Status register.
If
* another USB transaction is performed before the TOKDNE interrupt is
serviced, the
* USB module stores the status of the next transaction in the STAT FIFO.
Thus
* STAT is actually a four byte FIFO that allows the processor core to
process one
* transaction while the SIE is processing the next transaction. Clearing
the
* TOKDNE bit in the ISTAT register causes the SIE to update STAT with
the contents
* of the next STAT value. If the data in the STAT holding register is
valid, the
* SIE immediately reasserts to TOKDNE interrupt.
*/
/*!
* @name Constants and macros for entire USB_STAT register
*/
/*@{*/
#define USB_RD_STAT(base)          (USB_STAT_REG(base))
/*@}*/

```

```

/*
 * Constants & macros for individual USB_STAT bitfields
 */

/*!
 * @name Register USB_STAT, field ODD[2] (RO)
 *
 * This bit is set if the last buffer descriptor updated was in the odd
bank of
 * the BDT.
 */
/*@{*/
/*! @brief Read current value of the USB_STAT_ODD field. */
#define USB_RD_STAT_ODD(base) ((USB_STAT_REG(base) & USB_STAT_ODD_MASK) >> USB_STAT_ODD_SHIFT)
#define USB_BRD_STAT_ODD(base) (BME_UBFX8(&USB_STAT_REG(base), USB_STAT_ODD_SHIFT, USB_STAT_ODD_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_STAT, field TX[3] (RO)
 *
 * Values:
 * - 0b0 - The most recent transaction was a receive operation.
 * - 0b1 - The most recent transaction was a transmit operation.
 */
/*@{*/
/*! @brief Read current value of the USB_STAT_TX field. */
#define USB_RD_STAT_TX(base) ((USB_STAT_REG(base) & USB_STAT_TX_MASK) >> USB_STAT_TX_SHIFT)
#define USB_BRD_STAT_TX(base) (BME_UBFX8(&USB_STAT_REG(base), USB_STAT_TX_SHIFT, USB_STAT_TX_WIDTH))
/*}@*/ */

/*!
 * @name Register USB_STAT, field ENDP[7:4] (RO)
 *
 * This four-bit field encodes the endpoint address that received or
transmitted
 * the previous token. This allows the processor core to determine the
BDT entry
 * that was updated by the last USB transaction.
 */
/*@{*/
/*! @brief Read current value of the USB_STAT_ENDP field. */
#define USB_RD_STAT_ENDP(base) ((USB_STAT_REG(base) & USB_STAT_ENDP_MASK) >> USB_STAT_ENDP_SHIFT)
#define USB_BRD_STAT_ENDP(base) (BME_UBFX8(&USB_STAT_REG(base), USB_STAT_ENDP_SHIFT, USB_STAT_ENDP_WIDTH))
/*}@*/ */

*****
```

```

 * USB_CTL - Control register
 ****
 */

/*!
 * @brief USB_CTL - Control register (RW)
 *
 * Reset value: 0x00U
 *
 * Provides various control and configuration information for the USB
module.
 */
/*!
 * @name Constants and macros for entire USB_CTL register
 */
/*@*/
#define USB_RD_CTL(base)          (USB_CTL_REG(base))
#define USB_WR_CTL(base, value)   (USB_CTL_REG(base) = (value))
#define USB_RMW_CTL(base, mask, value) (USB_WR_CTL(base,
(USB_RD_CTL(base) & ~mask) | (value)))
#define USB_SET_CTL(base, value)  (BME_OR8(&USB_CTL_REG(base),
(uint8_t)(value)))
#define USB_CLR_CTL(base, value)  (BME_AND8(&USB_CTL_REG(base),
(uint8_t)(~value)))
#define USB_TOG_CTL(base, value)  (BME_XOR8(&USB_CTL_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual USB_CTL bitfields
 */
/*!
 * @name Register USB_CTL, field USBENSOFEN[0] (RW)
 *
 * Setting this bit causes the SIE to reset all of its ODD bits to the
BDTs.
 * Therefore, setting this bit resets much of the logic in the SIE. When
host mode
 * is enabled, clearing this bit causes the SIE to stop sending SOF
tokens.
 *
 * Values:
 * - 0b0 - Disables the USB Module.
 * - 0b1 - Enables the USB Module.
 */
/*@*/
/*! @brief Read current value of the USB_CTL_USBENSOFEN field. */
#define USB_RD_CTL_USBENSOFEN(base) ((USB_CTL_REG(base) &
USB_CTL_USBENSOFEN_MASK) >> USB_CTL_USBENSOFEN_SHIFT)
#define USB_BRD_CTL_USBENSOFEN(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_USBENSOFEN_SHIFT, USB_CTL_USBENSOFEN_WIDTH))

```

```

/*! @brief Set the USBENOFEN field to a new value. */
#define USB_WR_CTL_USBENOFEN(base, value) (USB_RMW_CTL(base,
USB_CTL_USBENOFEN_MASK, USB_CTL_USBENOFEN(value)))
#define USB_BWR_CTL_USBENOFEN(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_USBENOFEN_SHIFT), USB_CTL_USBENOFEN_SHIFT,
USB_CTL_USBENOFEN_WIDTH))
/*@}*/

/*!!
 * @name Register USB_CTL, field ODDRST[1] (RW)
 *
 * Setting this bit to 1 resets all the BDT ODD ping/pong fields to 0,
which
 * then specifies the EVEN BDT bank.
*/
/*@{*/
/*! @brief Read current value of the USB_CTL_ODDRST field. */
#define USB_RD_CTL_ODDRST(base) ((USB_CTL_REG(base) &
USB_CTL_ODDRST_MASK) >> USB_CTL_ODDRST_SHIFT)
#define USB_BRD_CTL_ODDRST(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_ODDRST_SHIFT, USB_CTL_ODDRST_WIDTH))

/*! @brief Set the ODDRST field to a new value. */
#define USB_WR_CTL_ODDRST(base, value) (USB_RMW_CTL(base,
USB_CTL_ODDRST_MASK, USB_CTL_ODDRST(value)))
#define USB_BWR_CTL_ODDRST(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_ODDRST_SHIFT), USB_CTL_ODDRST_SHIFT,
USB_CTL_ODDRST_WIDTH))
/*@}*/

/*!!
 * @name Register USB_CTL, field RESUME[2] (RW)
 *
 * When set to 1 this bit enables the USB Module to execute resume
signaling.
 * This allows the USB Module to perform remote wake-up. Software must
set RESUME
 * to 1 for the required amount of time and then clear it to 0. If the
HOSTMODEEN
 * bit is set, the USB module appends a Low Speed End of Packet to the
Resume
 * signaling when the RESUME bit is cleared. For more information on
RESUME
 * signaling see Section 7.1.4.5 of the USB specification version 1.0.
*/
/*@{*/
/*! @brief Read current value of the USB_CTL_RESUME field. */
#define USB_RD_CTL_RESUME(base) ((USB_CTL_REG(base) &
USB_CTL_RESUME_MASK) >> USB_CTL_RESUME_SHIFT)
#define USB_BRD_CTL_RESUME(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_RESUME_SHIFT, USB_CTL_RESUME_WIDTH))

/*! @brief Set the RESUME field to a new value. */

```

```

#define USB_WR_CTL_RESUME(base, value) (USB_RMW_CTL(base,
USB_CTL_RESUME_MASK, USB_CTL_RESUME(value)))
#define USB_BWR_CTL_RESUME(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_RESUME_SHIFT), USB_CTL_RESUME_SHIFT,
USB_CTL_RESUME_WIDTH))
/*@}*/

/*
 * @name Register USB_CTL, field HOSTMODEEN[3] (RW)
 *
 * When set to 1, this bit enables the USB Module to operate in Host
mode. In
 * host mode, the USB module performs USB transactions under the
programmed control
 * of the host processor.
 */
/*@{*/
/*! @brief Read current value of the USB_CTL_HOSTMODEEN field. */
#define USB_RD_CTL_HOSTMODEEN(base) ((USB_CTL_REG(base) &
USB_CTL_HOSTMODEEN_MASK) >> USB_CTL_HOSTMODEEN_SHIFT)
#define USB_BRD_CTL_HOSTMODEEN(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_HOSTMODEEN_SHIFT, USB_CTL_HOSTMODEEN_WIDTH))

/*! @brief Set the HOSTMODEEN field to a new value. */
#define USB_WR_CTL_HOSTMODEEN(base, value) (USB_RMW_CTL(base,
USB_CTL_HOSTMODEEN_MASK, USB_CTL_HOSTMODEEN(value)))
#define USB_BWR_CTL_HOSTMODEEN(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_HOSTMODEEN_SHIFT), USB_CTL_HOSTMODEEN_SHIFT,
USB_CTL_HOSTMODEEN_WIDTH))
/*@}*/

/*
 * @name Register USB_CTL, field RESET[4] (RW)
 *
 * Setting this bit enables the USB Module to generate USB reset
signaling. This
 * allows the USB Module to reset USB peripherals. This control signal is
only
 * valid in Host mode (HOSTMODEEN=1). Software must set RESET to 1 for
the
 * required amount of time and then clear it to 0 to end reset signaling.
For more
 * information on reset signaling see Section 7.1.4.3 of the USB
specification version
 * 1.0.
 */
/*@{*/
/*! @brief Read current value of the USB_CTL_RESET field. */
#define USB_RD_CTL_RESET(base) ((USB_CTL_REG(base) & USB_CTL_RESET_MASK)
>> USB_CTL_RESET_SHIFT)
#define USB_BRD_CTL_RESET(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_RESET_SHIFT, USB_CTL_RESET_WIDTH))

/*! @brief Set the RESET field to a new value. */

```

```

#define USB_WR_CTL_RESET(base, value) (USB_RMW_CTL(base,
USB_CTL_RESET_MASK, USB_CTL_RESET(value)))
#define USB_BWR_CTL_RESET(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_RESET_SHIFT), USB_CTL_RESET_SHIFT,
USB_CTL_RESET_WIDTH))
/*@}*/

/*
 * @name Register USB_CTL, field TXSUSPENDTOKENBUSY[5] (RW)
 *
 * In Host mode, TOKEN_BUSY is set when the USB module is busy executing
a USB
 * token. Software must not write more token commands to the Token
Register when
 * TOKEN_BUSY is set.. Software should check this field before writing
any tokens
 * to the Token Register to ensure that token commands are not lost. In
Target
 * mode, TXD_SUSPEND is set when the SIE has disabled packet transmission
and
 * reception. Clearing this bit allows the SIE to continue token
processing. This bit
 * is set by the SIE when a SETUP Token is received allowing software to
dequeue
 * any pending packet transactions in the BDT before resuming token
processing.
 */
/*@*/
/*! @brief Read current value of the USB_CTL_TXSUSPENDTOKENBUSY field. */
#define USB_RD_CTL_TXSUSPENDTOKENBUSY(base) ((USB_CTL_REG(base) &
USB_CTL_TXSUSPENDTOKENBUSY_MASK) >> USB_CTL_TXSUSPENDTOKENBUSY_SHIFT)
#define USB_BRD_CTL_TXSUSPENDTOKENBUSY(base)
(BME_UBFX8(&USB_CTL_REG(base), USB_CTL_TXSUSPENDTOKENBUSY_SHIFT,
USB_CTL_TXSUSPENDTOKENBUSY_WIDTH))

/*! @brief Set the TXSUSPENDTOKENBUSY field to a new value. */
#define USB_WR_CTL_TXSUSPENDTOKENBUSY(base, value) (USB_RMW_CTL(base,
USB_CTL_TXSUSPENDTOKENBUSY_MASK, USB_CTL_TXSUSPENDTOKENBUSY(value)))
#define USB_BWR_CTL_TXSUSPENDTOKENBUSY(base, value)
(BME_BFI8(&USB_CTL_REG(base), ((uint8_t)(value) <<
USB_CTL_TXSUSPENDTOKENBUSY_SHIFT), USB_CTL_TXSUSPENDTOKENBUSY_SHIFT,
USB_CTL_TXSUSPENDTOKENBUSY_WIDTH))
/*@*/

/*
 * @name Register USB_CTL, field SE0[6] (RW)
 */
/*@*/
/*! @brief Read current value of the USB_CTL_SE0 field. */
#define USB_RD_CTL_SE0(base) ((USB_CTL_REG(base) & USB_CTL_SE0_MASK) >>
USB_CTL_SE0_SHIFT)
#define USB_BRD_CTL_SE0(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_SE0_SHIFT, USB_CTL_SE0_WIDTH))

```

```

/*! @brief Set the SE0 field to a new value. */
#define USB_WR_CTL_SE0(base, value) (USB_RMW_CTL(base, USB_CTL_SE0_MASK,
USB_CTL_SE0(value)))
#define USB_BWR_CTL_SE0(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_SE0_SHIFT), USB_CTL_SE0_SHIFT,
USB_CTL_SE0_WIDTH))
/*@}*/

/*!!
 * @name Register USB_CTL, field JSTATE[7] (RW)
 *
 * The polarity of this signal is affected by the current state of LSEN .
 */
/*@{*/
/*! @brief Read current value of the USB_CTL_JSTATE field. */
#define USB_RD_CTL_JSTATE(base) ((USB_CTL_REG(base) &
USB_CTL_JSTATE_MASK) >> USB_CTL_JSTATE_SHIFT)
#define USB_BRD_CTL_JSTATE(base) (BME_UBFX8(&USB_CTL_REG(base),
USB_CTL_JSTATE_SHIFT, USB_CTL_JSTATE_WIDTH))

/*! @brief Set the JSTATE field to a new value. */
#define USB_WR_CTL_JSTATE(base, value) (USB_RMW_CTL(base,
USB_CTL_JSTATE_MASK, USB_CTL_JSTATE(value)))
#define USB_BWR_CTL_JSTATE(base, value) (BME_BFI8(&USB_CTL_REG(base),
((uint8_t)(value) << USB_CTL_JSTATE_SHIFT), USB_CTL_JSTATE_SHIFT,
USB_CTL_JSTATE_WIDTH))
/*@}*/

*****  

*****  

* USB_ADDR - Address register  

*****  

*****  

* @brief USB_ADDR - Address register (RW)  

* Reset value: 0x00U  

* Holds the unique USB address that the USB module decodes when in Peripheral mode (HOSTMODEEN=0). When operating in Host mode (HOSTMODEEN=1) the USB module  

* transmits this address with a TOKEN packet. This enables the USB module to  

* uniquely address an USB peripheral. In either mode, USB_EN within the control  

* register must be 1. The Address register is reset to 0x00 after the reset input  

* becomes active or the USB module decodes a USB reset signal. This action  

* initializes the Address register to decode address 0x00 as required by the USB

```

```

 * specification.
 */
/*!
 * @name Constants and macros for entire USB_ADDR register
 */
/*@{*/
#define USB_RD_ADDR(base)      (USB_ADDR_REG(base))
#define USB_WR_ADDR(base, value) (USB_ADDR_REG(base) = (value))
#define USB_RMW_ADDR(base, mask, value) (USB_WR_ADDR(base,
(USB_RD_ADDR(base) & ~mask) | (value)))
#define USB_SET_ADDR(base, value) (BME_OR8(&USB_ADDR_REG(base),
(uint8_t)(value)))
#define USB_CLR_ADDR(base, value) (BME_AND8(&USB_ADDR_REG(base),
(uint8_t)(~(value))))
#define USB_TOG_ADDR(base, value) (BME_XOR8(&USB_ADDR_REG(base),
(uint8_t)(value)))
/*}@*/
/*
 * Constants & macros for individual USB_ADDR bitfields
 */

/*!
 * @name Register USB_ADDR, field ADDR[6:0] (RW)
 *
 * Defines the USB address that the USB module decodes in peripheral
mode, or
 * transmits when in host mode.
 */
/*@{*/
/*! @brief Read current value of the USB_ADDR_ADDR field. */
#define USB_RD_ADDR_ADDR(base) ((USB_ADDR_REG(base) & USB_ADDR_ADDR_MASK)
>> USB_ADDR_ADDR_SHIFT)
#define USB_BRD_ADDR_ADDR(base) (BME_UBFX8(&USB_ADDR_REG(base),
USB_ADDR_ADDR_SHIFT, USB_ADDR_ADDR_WIDTH))

/*! @brief Set the ADDR field to a new value. */
#define USB_WR_ADDR_ADDR(base, value) (USB_RMW_ADDR(base,
USB_ADDR_ADDR_MASK, USB_ADDR_ADDR(value)))
#define USB_BWR_ADDR_ADDR(base, value) (BME_BFI8(&USB_ADDR_REG(base),
((uint8_t)(value) << USB_ADDR_ADDR_SHIFT), USB_ADDR_ADDR_SHIFT,
USB_ADDR_ADDR_WIDTH))
/*}@*/

/*!
 * @name Register USB_ADDR, field LSEN[7] (RW)
 *
 * Informs the USB module that the next token command written to the
token
 * register must be performed at low speed. This enables the USB module
to perform the
 * necessary preamble required for low-speed data transmissions.
 */
/*@{*/

```

```

/*! @brief Read current value of the USB_ADDR_LSEN field. */
#define USB_RD_ADDR_LSEN(base) ((USB_ADDR_REG(base) & USB_ADDR_LSEN_MASK)
>> USB_ADDR_LSEN_SHIFT)
#define USB_BRD_ADDR_LSEN(base) (BME_UBFX8(&USB_ADDR_REG(base),
USB_ADDR_LSEN_SHIFT, USB_ADDR_LSEN_WIDTH))

/*! @brief Set the LSEN field to a new value. */
#define USB_WR_ADDR_LSEN(base, value) (USB_RMW_ADDR(base,
USB_ADDR_LSEN_MASK, USB_ADDR_LSEN(value)))
#define USB_BWR_ADDR_LSEN(base, value) (BME_BFI8(&USB_ADDR_REG(base),
((uint8_t)(value) << USB_ADDR_LSEN_SHIFT), USB_ADDR_LSEN_SHIFT,
USB_ADDR_LSEN_WIDTH))
/*@}*/

/********************* USB_BDTPAGE1 - BDT Page Register 1 *****************/
***** */

/* @brief USB_BDTPAGE1 - BDT Page Register 1 (RW)
 *
 * Reset value: 0x00U
 *
 * Provides address bits 15 through 9 of the base address where the
current
 * Buffer Descriptor Table (BDT) resides in system memory. The 32-bit BDT
Base
 * Address is always aligned on 512-byte boundaries, so bits 8 through 0
of the base
 * address are always zero.
 */
/*!
 * @name Constants and macros for entire USB_BDTPAGE1 register
 */
/*@{ */

#define USB_RD_BDTPAGE1(base) (USB_BDTPAGE1_REG(base))
#define USB_WR_BDTPAGE1(base, value) (USB_BDTPAGE1_REG(base) = (value))
#define USB_RMW_BDTPAGE1(base, mask, value) (USB_WR_BDTPAGE1(base,
(USB_RD_BDTPAGE1(base) & ~mask) | (value)))
#define USB_SET_BDTPAGE1(base, value) (BME_OR8(&USB_BDTPAGE1_REG(base),
(uint8_t)(value)))
#define USB_CLR_BDTPAGE1(base, value) (BME_AND8(&USB_BDTPAGE1_REG(base),
(uint8_t)(~value)))
#define USB_TOG_BDTPAGE1(base, value) (BME_XOR8(&USB_BDTPAGE1_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual USB_BDTPAGE1 bitfields
 */

```

```

/*!
 * @name Register USB_BDTPAGE1, field BDTBA[7:1] (RW)
 *
 * Provides address bits 15 through 9 of the BDT base address.
 */
/*@{*/
/*! @brief Read current value of the USB_BDTPAGE1_BDTBA field. */
#define USB_RD_BDTPAGE1_BDTBA(base) ((USB_BDTPAGE1_REG(base) &
USB_BDTPAGE1_BDTBA_MASK) >> USB_BDTPAGE1_BDTBA_SHIFT)
#define USB_BRD_BDTPAGE1_BDTBA(base) (BME_UBFX8(&USB_BDTPAGE1_REG(base),
USB_BDTPAGE1_BDTBA_SHIFT, USB_BDTPAGE1_BDTBA_WIDTH))

/*! @brief Set the BDTBA field to a new value. */
#define USB_WR_BDTPAGE1_BDTBA(base, value) (USB_RMW_BDTPAGE1(base,
USB_BDTPAGE1_BDTBA_MASK, USB_BDTPAGE1_BDTBA(value)))
#define USB_BWR_BDTPAGE1_BDTBA(base, value)
(BME_BFI8(&USB_BDTPAGE1_REG(base), ((uint8_t)(value) <<
USB_BDTPAGE1_BDTBA_SHIFT), USB_BDTPAGE1_BDTBA_SHIFT,
USB_BDTPAGE1_BDTBA_WIDTH))
/*@}*/
/******
 * USB_FRMNUML - Frame Number Register Low
*****/
/*!
 * @brief USB_FRMNUML - Frame Number Register Low (RW)
 *
 * Reset value: 0x00U
 *
 * Contains an 11-bit value used to compute the address where the current
Buffer
 * Descriptor Table (BDT) resides in system memory.
*/
/*!
 * @name Constants and macros for entire USB_FRMNUML register
*/
/*@{*/
#define USB_RD_FRMNUML(base) (USB_FRMNUML_REG(base))
#define USB_WR_FRMNUML(base, value) (USB_FRMNUML_REG(base) = (value))
#define USB_RMW_FRMNUML(base, mask, value) (USB_WR_FRMNUML(base,
(USB_RD_FRMNUML(base) & ~mask) | (value)))
#define USB_SET_FRMNUML(base, value) (BME_OR8(&USB_FRMNUML_REG(base),
(uint8_t)(value)))
#define USB_CLR_FRMNUML(base, value) (BME_AND8(&USB_FRMNUML_REG(base),
(uint8_t)(~value))))
#define USB_TOG_FRMNUML(base, value) (BME_XOR8(&USB_FRMNUML_REG(base),
(uint8_t)(value)))
/*@}*/

```

```

*****
 * USB_FRMNUMH - Frame Number Register High
*****
**** */

/*!
 * @brief USB_FRMNUMH - Frame Number Register High (RW)
 *
 * Reset value: 0x00U
 *
 * Contains an 11-bit value used to compute the address where the current
Buffer
 * Descriptor Table (BDT) resides in system memory.
 */
/*!
 * @name Constants and macros for entire USB_FRMNUMH register
 */
/*@{ */
#define USB_RD_FRMNUMH(base)      (USB_FRMNUMH_REG(base))
#define USB_WR_FRMNUMH(base, value) (USB_FRMNUMH_REG(base) = (value))
#define USB_RMW_FRMNUMH(base, mask, value) (USB_WR_FRMNUMH(base,
(USB_RD_FRMNUMH(base) & ~mask) | (value)))
#define USB_SET_FRMNUMH(base, value) (BME_OR8(&USB_FRMNUMH_REG(base),
(uint8_t)(value)))
#define USB_CLR_FRMNUMH(base, value) (BME_AND8(&USB_FRMNUMH_REG(base),
(uint8_t)(~value)))
#define USB_TOG_FRMNUMH(base, value) (BME_XOR8(&USB_FRMNUMH_REG(base),
(uint8_t)(value)))
/*@} */

/*
 * Constants & macros for individual USB_FRMNUMH bitfields
 */

/*!
 * @name Register USB_FRMNUMH, field FRM[2:0] (RW)
 *
 * This 3-bit field and the 8-bit field in the Frame Number Register Low
are
 * used to compute the address where the current Buffer Descriptor Table
(BDT)
 * resides in system memory.
 */
/*@{ */
/*! @brief Read current value of the USB_FRMNUMH_FRM field. */
#define USB_RD_FRMNUMH_FRM(base) ((USB_FRMNUMH_REG(base) &
USB_FRMNUMH_FRM_MASK) >> USB_FRMNUMH_FRM_SHIFT)
#define USB_BRD_FRMNUMH_FRM(base) (BME_UBFX8(&USB_FRMNUMH_REG(base),
USB_FRMNUMH_FRM_SHIFT, USB_FRMNUMH_FRM_WIDTH))

/*! @brief Set the FRM field to a new value. */

```

```

#define USB_WR_FRMNUMH_FRM(base, value) (USB_RMW_FRMNUMH(base, USB_FRMNUMH_FRM_MASK, USB_FRMNUMH_FRM(value)))
#define USB_BWR_FRMNUMH_FRM(base, value)
(BME_BFI8(&USB_FRMNUMH_REG(base), ((uint8_t)(value) << USB_FRMNUMH_FRM_SHIFT), USB_FRMNUMH_FRM_SHIFT, USB_FRMNUMH_FRM_WIDTH))
/*@}*/

/********************* USB_TOKEN - Token register *****************/
***** */

/* @brief USB_TOKEN - Token register (RW)
 *
 * Reset value: 0x00U
 *
 * Used to initiate USB transactions when in host mode (HOSTMODEEN=1).
When the
 * software needs to execute a USB transaction to a peripheral, it writes
the
 * TOKEN type and endpoint to this register. After this register has been
written,
 * the USB module begins the specified USB transaction to the address
contained in
 * the address register. The processor core must always check that the
 * TOKEN_BUSY bit in the control register is not 1 before writing to the
Token Register.
 * This ensures that the token commands are not overwritten before they
can be
 * executed. The address register and endpoint control register 0 are
also used when
 * performing a token command and therefore must also be written before
the
 * Token Register. The address register is used to select the USB
peripheral address
 * transmitted by the token command. The endpoint control register
determines the
 * handshake and retry policies used during the transfer.
 */
/*!
 * @name Constants and macros for entire USB_TOKEN register
 */
/*@{*/
#define USB_RD_TOKEN(base) (USB_TOKEN_REG(base))
#define USB_WR_TOKEN(base, value) (USB_TOKEN_REG(base) = (value))
#define USB_RMW_TOKEN(base, mask, value) (USB_WR_TOKEN(base,
(USB_RD_TOKEN(base) & ~mask)) | (value)))
#define USB_SET_TOKEN(base, value) (BME_OR8(&USB_TOKEN_REG(base),
(uint8_t)(value)))
#define USB_CLR_TOKEN(base, value) (BME_AND8(&USB_TOKEN_REG(base),
(uint8_t)(~(value))))

```

```

#define USB_TOG_TOKEN(base, value) (BME_XOR8 (&USB_TOKEN_REG(base),  

(uint8_t)(value)))  

/*@*/
```

```

/*
 * Constants & macros for individual USB_TOKEN bitfields
 */

/*!
 * @name Register USB_TOKEN, field TOKENENDPT[3:0] (RW)
 *
 * Holds the Endpoint address for the token command. The four bit value
written
 * must be a valid endpoint.
 */
/*@{*/
/*! @brief Read current value of the USB_TOKEN_TOKENENDPT field. */  

#define USB_RD_TOKEN_TOKENENDPT(base) ((USB_TOKEN_REG(base) &  

USB_TOKEN_TOKENENDPT_MASK) >> USB_TOKEN_TOKENENDPT_SHIFT)  

#define USB_BRD_TOKEN_TOKENENDPT(base) (BME_UBFX8(&USB_TOKEN_REG(base),  

USB_TOKEN_TOKENENDPT_SHIFT, USB_TOKEN_TOKENENDPT_WIDTH))
```

```

/*! @brief Set the TOKENENDPT field to a new value. */  

#define USB_WR_TOKEN_TOKENENDPT(base, value) (USB_RMW_TOKEN(base,  

USB_TOKEN_TOKENENDPT_MASK, USB_TOKEN_TOKENENDPT(value)))  

#define USB_BWR_TOKEN_TOKENENDPT(base, value)  

(BME_BFI8(&USB_TOKEN_REG(base), ((uint8_t)(value) <<  

USB_TOKEN_TOKENENDPT_SHIFT), USB_TOKEN_TOKENENDPT_SHIFT,  

USB_TOKEN_TOKENENDPT_WIDTH))  

/*@*/
```

```

/*!
 * @name Register USB_TOKEN, field TOKENPID[7:4] (RW)
 *
 * Contains the token type executed by the USB module.
 *
 * Values:
 * - 0b0001 - OUT Token. USB Module performs an OUT (TX) transaction.
 * - 0b1001 - IN Token. USB Module performs an In (RX) transaction.
 * - 0b1101 - SETUP Token. USB Module performs a SETUP (TX) transaction
 */
/*@{*/
/*! @brief Read current value of the USB_TOKEN_TOKENPID field. */  

#define USB_RD_TOKEN_TOKENPID(base) ((USB_TOKEN_REG(base) &  

USB_TOKEN_TOKENPID_MASK) >> USB_TOKEN_TOKENPID_SHIFT)  

#define USB_BRD_TOKEN_TOKENPID(base) (BME_UBFX8(&USB_TOKEN_REG(base),  

USB_TOKEN_TOKENPID_SHIFT, USB_TOKEN_TOKENPID_WIDTH))
```

```

/*! @brief Set the TOKENPID field to a new value. */  

#define USB_WR_TOKEN_TOKENPID(base, value) (USB_RMW_TOKEN(base,  

USB_TOKEN_TOKENPID_MASK, USB_TOKEN_TOKENPID(value)))  

#define USB_BWR_TOKEN_TOKENPID(base, value)  

(BME_BFI8(&USB_TOKEN_REG(base), ((uint8_t)(value) <<
```

```

USB_TOKEN_TOKENPID_SHIFT), USB_TOKEN_TOKENPID_SHIFT,
USB_TOKEN_TOKENPID_WIDTH))
/*@}*/
*****  

* USB_SOFTHLD - SOF Threshold Register  

*****  

*****/  

/*!  

 * @brief USB_SOFTHLD - SOF Threshold Register (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * The SOF Threshold Register is used only in Host mode (HOSTMODEEN=1).  

When in  

 * Host mode, the 14-bit SOF counter counts the interval between SOF  

frames. The  

 * SOF must be transmitted every 1ms so therefore the SOF counter is  

loaded with  

 * a value of 12000. When the SOF counter reaches zero, a Start Of Frame  

(SOF)  

 * token is transmitted. The SOF threshold register is used to program  

the number  

 * of USB byte times before the SOF to stop initiating token packet  

transactions.  

 * This register must be set to a value that ensures that other packets  

are not  

 * actively being transmitted when the SOF time counts to zero. When the  

SOF  

 * counter reaches the threshold value, no more tokens are transmitted  

until after the  

 * SOF has been transmitted. The value programmed into the threshold  

register  

 * must reserve enough time to ensure the worst case transaction  

completes. In  

 * general the worst case transaction is an IN token followed by a data  

packet from  

 * the target followed by the response from the host. The actual time  

required is  

 * a function of the maximum packet size on the bus. Typical values for  

the SOF  

 * threshold are: 64-byte packets=74; 32-byte packets=42; 16-byte  

packets=26;  

 * 8-byte packets=18.  

*/  

/*!  

 * @name Constants and macros for entire USB_SOFTHLD register  

*/  

/*@{*/
#define USB_RD_SOFTHLD(base)      (USB_SOFTHLD_REG(base))  

#define USB_WR_SOFTHLD(base, value) (USB_SOFTHLD_REG(base) = (value))

```

```

#define USB_RMW_SOFTHLD(base, mask, value) (USB_WR_SOFTHLD(base, (USB_RD_SOFTHLD(base) & ~(mask)) | (value)))
#define USB_SET_SOFTHLD(base, value) (BME_OR8(&USB_SOFTHLD_REG(base), (uint8_t)(value)))
#define USB_CLR_SOFTHLD(base, value) (BME_AND8(&USB_SOFTHLD_REG(base), (uint8_t)(~(value))))
#define USB_TOG_SOFTHLD(base, value) (BME_XOR8(&USB_SOFTHLD_REG(base), (uint8_t)(value)))
/*@}*/

/*********************  

*****  

 * USB_BDTPAGE2 - BDT Page Register 2  

*****  

****/  

  

/*!  

 * @brief USB_BDTPAGE2 - BDT Page Register 2 (RW)  

 *  

 * Reset value: 0x00U  

 *  

 * Contains an 8-bit value used to compute the address where the current  

Buffer  

 * Descriptor Table (BDT) resides in system memory.  

 */  

/*!  

 * @name Constants and macros for entire USB_BDTPAGE2 register  

 */  

/*@*/  

#define USB_RD_BDTPAGE2(base) (USB_BDTPAGE2_REG(base))
#define USB_WR_BDTPAGE2(base, value) (USB_BDTPAGE2_REG(base) = (value))
#define USB_RMW_BDTPAGE2(base, mask, value) (USB_WR_BDTPAGE2(base, (USB_RD_BDTPAGE2(base) & ~(mask)) | (value)))
#define USB_SET_BDTPAGE2(base, value) (BME_OR8(&USB_BDTPAGE2_REG(base), (uint8_t)(value)))
#define USB_CLR_BDTPAGE2(base, value) (BME_AND8(&USB_BDTPAGE2_REG(base), (uint8_t)(~(value))))
#define USB_TOG_BDTPAGE2(base, value) (BME_XOR8(&USB_BDTPAGE2_REG(base), (uint8_t)(value)))
/*@*/  

  

/*********************  

*****  

 * USB_BDTPAGE3 - BDT Page Register 3  

*****  

****/  

  

/*!  

 * @brief USB_BDTPAGE3 - BDT Page Register 3 (RW)  

 *  

 * Reset value: 0x00U  

 */

```

```

 * Contains an 8-bit value used to compute the address where the current
Buffer
 * Descriptor Table (BDT) resides in system memory.
 */
/*!
 * @name Constants and macros for entire USB_BDTPAGE3 register
 */
/*@{ */
#define USB_RD_BDTPAGE3(base)      (USB_BDTPAGE3_REG(base))
#define USB_WR_BDTPAGE3(base, value) (USB_BDTPAGE3_REG(base) = (value))
#define USB_RMW_BDTPAGE3(base, mask, value) (USB_WR_BDTPAGE3(base,
(USB_RD_BDTPAGE3(base) & ~mask) | (value)))
#define USB_SET_BDTPAGE3(base, value) (BME_OR8(&USB_BDTPAGE3_REG(base),
(uint8_t)(value)))
#define USB_CLR_BDTPAGE3(base, value) (BME_AND8(&USB_BDTPAGE3_REG(base),
(uint8_t)(~value)))
#define USB_TOG_BDTPAGE3(base, value) (BME_XOR8(&USB_BDTPAGE3_REG(base),
(uint8_t)(value)))
/*@} */

/********************* ENDPT - Endpoint Control register *****

 * USB-ENDPT - Endpoint Control register
 *****/
/*!
 * @brief USB-ENDPT - Endpoint Control register (RW)
 *
 * Reset value: 0x00U
 *
 * Contains the endpoint control bits for each of the 16 endpoints
available
 * within the USB module for a decoded address. The format for these
registers is
 * shown in the following figure. Endpoint 0 (ENDPT0) is associated with
control
 * pipe 0, which is required for all USB functions. Therefore, after a
USBRST
 * interrupt occurs the processor core should set ENDPT0 to contain 0x0D.
In Host mode
 * ENDPT0 is used to determine the handshake, retry and low speed
 * characteristics of the host transfer. For Control, Bulk and Interrupt
transfers, the EPHSHK
 * bit should be 1. For Isochronous transfers it should be 0. Common
values to
 * use for ENDPT0 in host mode are 0x4D for Control, Bulk, and Interrupt
transfers,
 * and 0x4C for Isochronous transfers.
 */
/*!
 * @name Constants and macros for entire USB-ENDPT register
 */

```

```

/*@{*/
#define USB_RD_ENDPT(base, index) (USB_ENDPT_REG(base, index))
#define USB_WR_ENDPT(base, index, value) (USB_ENDPT_REG(base, index) = (value))
#define USB_RMW_ENDPT(base, index, mask, value) (USB_WR_ENDPT(base, index, (USB_RD_ENDPT(base, index) & ~mask) | (value)))
#define USB_SET_ENDPT(base, index, value) (BME_OR8(&USB_ENDPT_REG(base, index), (uint8_t)(value)))
#define USB_CLR_ENDPT(base, index, value) (BME_AND8(&USB_ENDPT_REG(base, index), (uint8_t)(~(value))))
#define USB_TOG_ENDPT(base, index, value) (BME_XOR8(&USB_ENDPT_REG(base, index), (uint8_t)(value)))
/*}@*/
```

/\*
 \* Constants & macros for individual USB\_ENDPT bitfields
 \*/

```

/*!
 * @name Register USB_ENDPT, field EPHSHK[0] (RW)
 *
 * When set this bit enables an endpoint to perform handshaking during a
 * transaction to this endpoint. This bit is generally 1 unless the
endpoint is
 * Isochronous.
 */
/*@{*/
/*! @brief Read current value of the USB_ENDPT_EPHSHK field. */
#define USB_RD_ENDPT_EPHSHK(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_EPHSHK_MASK) >> USB_ENDPT_EPHSHK_SHIFT)
#define USB_BRD_ENDPT_EPHSHK(base, index) (BME_UBFX8(&USB_ENDPT_REG(base,
index), USB_ENDPT_EPHSHK_SHIFT, USB_ENDPT_EPHSHK_WIDTH))

/*! @brief Set the EPHSHK field to a new value. */
#define USB_WR_ENDPT_EPHSHK(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_EPHSHK_MASK, USB_ENDPT_EPHSHK(value)))
#define USB_BWR_ENDPT_EPHSHK(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_EPHSHK_SHIFT), USB_ENDPT_EPHSHK_SHIFT, USB_ENDPT_EPHSHK_WIDTH))
/*}@*/
```

/\*
 \* @name Register USB\_ENDPT, field EPSTALL[1] (RW)
 \*
 \* When set this bit indicates that the endpoint is stalled. This bit has
 \* priority over all other control bits in the EndPoint Enable Register,
but it is only
 \* valid if EPTXEN=1 or EPRXEN=1. Any access to this endpoint causes the
USB
 \* Module to return a STALL handshake. After an endpoint is stalled it
requires
 \* intervention from the Host Controller.
 \*/
/\*@{\*/

```

/*! @brief Read current value of the USB-ENDPT-EPSTALL field. */
#define USB_RD_ENDPT_EPSTALL(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_EPSTALL_MASK) >> USB_ENDPT_EPSTALL_SHIFT)
#define USB_BRD_ENDPT_EPSTALL(base, index)
(BME_UBFX8(&USB_ENDPT_REG(base, index), USB_ENDPT_EPSTALL_SHIFT,
USB_ENDPT_EPSTALL_WIDTH))

/*! @brief Set the EPSTALL field to a new value. */
#define USB_WR_ENDPT_EPSTALL(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_EPSTALL_MASK, USB_ENDPT_EPSTALL(value)))
#define USB_BWR_ENDPT_EPSTALL(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_EPSTALL_SHIFT), USB_ENDPT_EPSTALL_SHIFT,
USB_ENDPT_EPSTALL_WIDTH))
/*@}*/

/*!
* @name Register USB-ENDPT, field EPTXEN[2] (RW)
*
* This bit, when set, enables the endpoint for TX transfers.
*/
/*@{*/
/*! @brief Read current value of the USB-ENDPT-EPTXEN field. */
#define USB_RD_ENDPT_EPTXEN(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_EPTXEN_MASK) >> USB_ENDPT_EPTXEN_SHIFT)
#define USB_BRD_ENDPT_EPTXEN(base, index) (BME_UBFX8(&USB_ENDPT_REG(base,
index), USB_ENDPT_EPTXEN_SHIFT, USB_ENDPT_EPTXEN_WIDTH))

/*! @brief Set the EPTXEN field to a new value. */
#define USB_WR_ENDPT_EPTXEN(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_EPTXEN_MASK, USB_ENDPT_EPTXEN(value)))
#define USB_BWR_ENDPT_EPTXEN(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_EPTXEN_SHIFT), USB_ENDPT_EPTXEN_SHIFT, USB_ENDPT_EPTXEN_WIDTH))
/*@}*/

/*!
* @name Register USB-ENDPT, field EPRXEN[3] (RW)
*
* This bit, when set, enables the endpoint for RX transfers.
*/
/*@{*/
/*! @brief Read current value of the USB-ENDPT-EPRXEN field. */
#define USB_RD_ENDPT_EPRXEN(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_EPRXEN_MASK) >> USB_ENDPT_EPRXEN_SHIFT)
#define USB_BRD_ENDPT_EPRXEN(base, index) (BME_UBFX8(&USB_ENDPT_REG(base,
index), USB_ENDPT_EPRXEN_SHIFT, USB_ENDPT_EPRXEN_WIDTH))

/*! @brief Set the EPRXEN field to a new value. */
#define USB_WR_ENDPT_EPRXEN(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_EPRXEN_MASK, USB_ENDPT_EPRXEN(value)))
#define USB_BWR_ENDPT_EPRXEN(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_EPRXEN_SHIFT), USB_ENDPT_EPRXEN_SHIFT, USB_ENDPT_EPRXEN_WIDTH))

```

```

/*@}*/

/*
 * @name Register USB_ENDPT, field EPCTLDIS[4] (RW)
 *
 * This bit, when set, disables control (SETUP) transfers. When cleared,
control
 * transfers are enabled. This applies if and only if the EPRXEN and
EPTXEN bits
 * are also set.
*/
/*@{*/
/*! @brief Read current value of the USB_ENDPT_EPCTLDIS field. */
#define USB_RD_ENDPT_EPCTLDIS(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_EPCTLDIS_MASK) >> USB_ENDPT_EPCTLDIS_SHIFT)
#define USB_BRD_ENDPT_EPCTLDIS(base, index)
(BME_UBFX8(&USB_ENDPT_REG(base, index), USB_ENDPT_EPCTLDIS_SHIFT,
USB_ENDPT_EPCTLDIS_WIDTH))

/*! @brief Set the EPCTLDIS field to a new value. */
#define USB_WR_ENDPT_EPCTLDIS(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_EPCTLDIS_MASK, USB_ENDPT_EPCTLDIS(value)))
#define USB_BWR_ENDPT_EPCTLDIS(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_EPCTLDIS_SHIFT), USB_ENDPT_EPCTLDIS_SHIFT,
USB_ENDPT_EPCTLDIS_WIDTH))
/*@}*/

/*
 * @name Register USB_ENDPT, field RETRYDIS[6] (RW)
 *
 * This is a Host mode only bit and is present in the control register
for
 * endpoint 0 (ENDPT0) only. When set this bit causes the host to not
retry NAK'ed
 * (Negative Acknowledgement) transactions. When a transaction is NAKed,
the BDT PID
 * field is updated with the NAK PID, and the TOKEN_DNE interrupt is set.
When
 * this bit is cleared NAKed transactions is retried in hardware. This
bit must be
 * set when the host is attempting to poll an interrupt endpoint.
*/
/*@{*/
/*! @brief Read current value of the USB_ENDPT_RETRYDIS field. */
#define USB_RD_ENDPT_RETRYDIS(base, index) ((USB_ENDPT_REG(base, index) &
USB_ENDPT_RETRYDIS_MASK) >> USB_ENDPT_RETRYDIS_SHIFT)
#define USB_BRD_ENDPT_RETRYDIS(base, index)
(BME_UBFX8(&USB_ENDPT_REG(base, index), USB_ENDPT_RETRYDIS_SHIFT,
USB_ENDPT_RETRYDIS_WIDTH))

/*! @brief Set the RETRYDIS field to a new value. */
#define USB_WR_ENDPT_RETRYDIS(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_RETRYDIS_MASK, USB_ENDPT_RETRYDIS(value)))

```

```

#define USB_BWR_ENDPT_RETRYDIS(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_RETRYDIS_SHIFT), USB_ENDPT_RETRYDIS_SHIFT,
USB_ENDPT_RETRYDIS_WIDTH))
/*@}*/

/*!
 * @name Register USB_ENDPT, field HOSTWOHUB[7] (RW)
 *
 * This is a Host mode only field and is present in the control register
for
 * endpoint 0 (ENDPT0) only. When set this bit allows the host to
communicate to a
 * directly connected low speed device. When cleared, the host produces
the
 * PRE_PID. It then switches to low-speed signaling when sends a token to
a low speed
 * device as required to communicate with a low speed device through a
hub.
 */
/*@*/
/*! @brief Read current value of the USB_ENDPT_HOSTWOHUB field. */
#define USB_RD_ENDPT_HOSTWOHUB(base, index) ((USB_ENDPT_REG(base, index)
& USB_ENDPT_HOSTWOHUB_MASK) >> USB_ENDPT_HOSTWOHUB_SHIFT)
#define USB_BRD_ENDPT_HOSTWOHUB(base, index)
(BME_UBFX8(&USB_ENDPT_REG(base, index), USB_ENDPT_HOSTWOHUB_SHIFT,
USB_ENDPT_HOSTWOHUB_WIDTH))

/*! @brief Set the HOSTWOHUB field to a new value. */
#define USB_WR_ENDPT_HOSTWOHUB(base, index, value) (USB_RMW_ENDPT(base,
index, USB_ENDPT_HOSTWOHUB_MASK, USB_ENDPT_HOSTWOHUB(value)))
#define USB_BWR_ENDPT_HOSTWOHUB(base, index, value)
(BME_BFI8(&USB_ENDPT_REG(base, index), ((uint8_t)(value) <<
USB_ENDPT_HOSTWOHUB_SHIFT), USB_ENDPT_HOSTWOHUB_SHIFT,
USB_ENDPT_HOSTWOHUB_WIDTH))
/*@*/

/***********************
*****
 * USB_USBCTRL - USB Control register
*****
***********************/

/*!
 * @brief USB_USBCTRL - USB Control register (RW)
 *
 * Reset value: 0xC0U
 */
/*!
 * @name Constants and macros for entire USB_USBCTRL register
 */
/*@*/
#define USB_RD_USBCTRL(base) (USB_USBCTRL_REG(base))

```

```

#define USB_WR_USBCTRL(base, value)  (USB_USBCTRL_REG(base) = (value))
#define USB_RMW_USBCTRL(base, mask, value)  (USB_WR_USBCTRL(base,
(USB_RD_USBCTRL(base) & ~mask) | (value)))
#define USB_SET_USBCTRL(base, value)  (BME_OR8(&USB_USBCTRL_REG(base),
(uint8_t)(value)))
#define USB_CLR_USBCTRL(base, value)  (BME_AND8(&USB_USBCTRL_REG(base),
(uint8_t)(~(value))))
#define USB_TOG_USBCTRL(base, value)  (BME_XOR8(&USB_USBCTRL_REG(base),
(uint8_t)(value)))
/*@}*/

/*
 * Constants & macros for individual USB_USBCTRL bitfields
 */

/*!!
 * @name Register USB_USBCTRL, field PDE[6] (RW)
 *
 * Enables the weak pulldowns on the USB transceiver.
 *
 * Values:
 * - 0b0 - Weak pulldowns are disabled on D+ and D-.
 * - 0b1 - Weak pulldowns are enabled on D+ and D-.
 */
/*@*/
/**! @brief Read current value of the USB_USBCTRL_PDE field. */
#define USB_RD_USBCTRL_PDE(base) ((USB_USBCTRL_REG(base) &
USB_USBCTRL_PDE_MASK) >> USB_USBCTRL_PDE_SHIFT)
#define USB_BRD_USBCTRL_PDE(base) (BME_UBFX8(&USB_USBCTRL_REG(base),
USB_USBCTRL_PDE_SHIFT, USB_USBCTRL_PDE_WIDTH))

/**! @brief Set the PDE field to a new value. */
#define USB_WR_USBCTRL_PDE(base, value) (USB_RMW_USBCTRL(base,
USB_USBCTRL_PDE_MASK, USB_USBCTRL_PDE(value)))
#define USB_BWR_USBCTRL_PDE(base, value)
(BME_BFI8(&USB_USBCTRL_REG(base), ((uint8_t)(value) <<
USB_USBCTRL_PDE_SHIFT), USB_USBCTRL_PDE_SHIFT, USB_USBCTRL_PDE_WIDTH))
/*@*/

/*!!
 * @name Register USB_USBCTRL, field SUSP[7] (RW)
 *
 * Places the USB transceiver into the suspend state.
 *
 * Values:
 * - 0b0 - USB transceiver is not in suspend state.
 * - 0b1 - USB transceiver is in suspend state.
 */
/*@*/
/**! @brief Read current value of the USB_USBCTRL_SUSP field. */
#define USB_RD_USBCTRL_SUSP(base) ((USB_USBCTRL_REG(base) &
USB_USBCTRL_SUSP_MASK) >> USB_USBCTRL_SUSP_SHIFT)
#define USB_BRD_USBCTRL_SUSP(base) (BME_UBFX8(&USB_USBCTRL_REG(base),
USB_USBCTRL_SUSP_SHIFT, USB_USBCTRL_SUSP_WIDTH))

```

```

/*! @brief Set the SUSP field to a new value. */
#define USB_WR_USBCTRL_SUSP(base, value) (USB_RMW_USBCTRL(base,
USB_USBCTRL_SUSP_MASK, USB_USBCTRL_SUSP(value)))
#define USB_BWR_USBCTRL_SUSP(base, value)
(BME_BFI8(&USB_USBCTRL_REG(base), ((uint8_t)(value) <<
USB_USBCTRL_SUSP_SHIFT), USB_USBCTRL_SUSP_SHIFT, USB_USBCTRL_SUSP_WIDTH))
/*@}*/

/********************* USB_OBSERVE - USB OTG Observe register *****/
****

* USB_OBSERVE - USB OTG Observe register

*****/
****

/*!
* @brief USB_OBSERVE - USB OTG Observe register (RO)
*
* Reset value: 0x50U
*
* Provides visibility on the state of the pull-ups and pull-downs at the
* transceiver. Useful when interfacing to an external OTG control module
via a serial
* interface.
*/
/*!
* @name Constants and macros for entire USB_OBSERVE register
*/
/*@*/
#define USB_RD_OBSERVE(base) (USB_OBSERVE_REG(base))
/*@*/

/*
* Constants & macros for individual USB_OBSERVE bitfields
*/
/*!

* @name Register USB_OBSERVE, field DMPD[4] (RO)
*
* Provides observability of the D- Pulldown . signal output from the USB
OTG
* module
*
* Values:
* - 0b0 - D- pulldown disabled.
* - 0b1 - D- pulldown enabled.
*/
/*@*/
/*! @brief Read current value of the USB_OBSERVE_DMPD field. */
#define USB_RD_OBSERVE_DMPD(base) ((USB_OBSERVE_REG(base) &
USB_OBSERVE_DMPD_MASK) >> USB_OBSERVE_DMPD_SHIFT)
#define USB_BRD_OBSERVE_DMPD(base) (BME_UBFX8(&USB_OBSERVE_REG(base),
USB_OBSERVE_DMPD_SHIFT, USB_OBSERVE_DMPD_WIDTH))

```

```

/*@}*/

/*
 * @name Register USB_OBSERVE, field DPPD[6] (RO)
 *
 * Provides observability of the D+ Pulldown . signal output from the USB
OTG
 * module
 *
 * Values:
 * - 0b0 - D+ pulldown disabled.
 * - 0b1 - D+ pulldown enabled.
 */
/*@{*/
/*! @brief Read current value of the USB_OBSERVE_DPPD field. */
#define USB_RD_OBSERVE_DPPD(base) ((USB_OBSERVE_REG(base) &
USB_OBSERVE_DPPD_MASK) >> USB_OBSERVE_DPPD_SHIFT)
#define USB_BRD_OBSERVE_DPPD(base) (BME_UBFX8(&USB_OBSERVE_REG(base),
USB_OBSERVE_DPPD_SHIFT, USB_OBSERVE_DPPD_WIDTH))
/*@}*/

/*
 * @name Register USB_OBSERVE, field DPPU[7] (RO)
 *
 * Provides observability of the D+ Pullup . signal output from the USB
OTG
 * module
 *
 * Values:
 * - 0b0 - D+ pullup disabled.
 * - 0b1 - D+ pullup enabled.
 */
/*@{*/
/*! @brief Read current value of the USB_OBSERVE_DPPU field. */
#define USB_RD_OBSERVE_DPPU(base) ((USB_OBSERVE_REG(base) &
USB_OBSERVE_DPPU_MASK) >> USB_OBSERVE_DPPU_SHIFT)
#define USB_BRD_OBSERVE_DPPU(base) (BME_UBFX8(&USB_OBSERVE_REG(base),
USB_OBSERVE_DPPU_SHIFT, USB_OBSERVE_DPPU_WIDTH))
/*@}*/

*****
* USB_CONTROL - USB OTG Control register
*****
/*!
 * @brief USB_CONTROL - USB OTG Control register (RW)
 *
 * Reset value: 0x00U
 */
/*!
 * @name Constants and macros for entire USB_CONTROL register

```

```

/*
/*@{*/
#define USB_RD_CONTROL(base)      (USB_CONTROL_REG(base))
#define USB_WR_CONTROL(base, value) (USB_CONTROL_REG(base) = (value))
#define USB_RMW_CONTROL(base, mask, value) (USB_WR_CONTROL(base,
(USB_RD_CONTROL(base) & ~mask) | (value)))
#define USB_SET_CONTROL(base, value) (BME_OR8(&USB_CONTROL_REG(base),
(uint8_t)(value)))
#define USB_CLR_CONTROL(base, value) (BME_AND8(&USB_CONTROL_REG(base),
(uint8_t)(~value))))
#define USB_TOG_CONTROL(base, value) (BME_XOR8(&USB_CONTROL_REG(base),
(uint8_t)(value)))
/*@}*/
/*!
 * Constants & macros for individual USB_CONTROL bitfields
 */

/*!
 * @name Register USB_CONTROL, field DPPULLUPNONOTG[4] (RW)
 *
 * Provides control of the DP Pullup in the USB OTG module, if USB is
configured
 * in non-OTG device mode.
 *
 * Values:
 * - 0b0 - DP Pullup in non-OTG device mode is not enabled.
 * - 0b1 - DP Pullup in non-OTG device mode is enabled.
 */
/*@*/
/*! @brief Read current value of the USB_CONTROL_DPPULLUPNONOTG field. */
#define USB_RD_CONTROL_DPPULLUPNONOTG(base) ((USB_CONTROL_REG(base) &
USB_CONTROL_DPPULLUPNONOTG_MASK) >> USB_CONTROL_DPPULLUPNONOTG_SHIFT)
#define USB_BRD_CONTROL_DPPULLUPNONOTG(base)
(BME_UBFX8(&USB_CONTROL_REG(base), USB_CONTROL_DPPULLUPNONOTG_SHIFT,
USB_CONTROL_DPPULLUPNONOTG_WIDTH))

/*! @brief Set the DPPULLUPNONOTG field to a new value. */
#define USB_WR_CONTROL_DPPULLUPNONOTG(base, value) (USB_RMW_CONTROL(base,
USB_CONTROL_DPPULLUPNONOTG_MASK, USB_CONTROL_DPPULLUPNONOTG(value)))
#define USB_BWR_CONTROL_DPPULLUPNONOTG(base, value)
(BME_BFI8(&USB_CONTROL_REG(base), ((uint8_t)(value) <<
USB_CONTROL_DPPULLUPNONOTG_SHIFT), USB_CONTROL_DPPULLUPNONOTG_SHIFT,
USB_CONTROL_DPPULLUPNONOTG_WIDTH))
/*@}*/

*****
* USB_USBTRC0 - USB Transceiver Control Register 0
*****
*/

```

```

* @brief USB_USBTRC0 - USB Transceiver Control Register 0 (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire USB_USBTRC0 register
*/
/*@{ */
#define USB_RD_USBTRC0(base)      (USB_USBTRC0_REG(base))
#define USB_WR_USBTRC0(base, value) (USB_USBTRC0_REG(base) = (value))
#define USB_RMW_USBTRC0(base, mask, value) (USB_WR_USBTRC0(base,
(USB_RD_USBTRC0(base) & ~mask) | (value)))
#define USB_SET_USBTRC0(base, value) (BME_OR8(&USB_USBTRC0_REG(base),
(uint8_t)(value)))
#define USB_CLR_USBTRC0(base, value) (BME_AND8(&USB_USBTRC0_REG(base),
(uint8_t)(~value)))
#define USB_TOG_USBTRC0(base, value) (BME_XOR8(&USB_USBTRC0_REG(base),
(uint8_t)(value)))
/*@} */

/*
* Constants & macros for individual USB_USBTRC0 bitfields
*/
/*!
* @name Register USB_USBTRC0, field USB_RESUME_INT[0] (RO)
*
* Values:
* - 0b0 - No interrupt was generated.
* - 0b1 - Interrupt was generated because of the USB asynchronous
interrupt.
*/
/*@{ */
/*! @brief Read current value of the USB_USBTRC0_USB_RESUME_INT field. */
#define USB_RD_USBTRC0_USB_RESUME_INT(base) ((USB_USBTRC0_REG(base) &
USB_USBTRC0_USB_RESUME_INT_MASK) >> USB_USBTRC0_USB_RESUME_INT_SHIFT)
#define USB_BRD_USBTRC0_USB_RESUME_INT(base)
(BME_UBFX8(&USB_USBTRC0_REG(base), USB_USBTRC0_USB_RESUME_INT_SHIFT,
USB_USBTRC0_USB_RESUME_INT_WIDTH))
/*@} */

/*!
* @name Register USB_USBTRC0, field SYNC_DET[1] (RO)
*
* Values:
* - 0b0 - Synchronous interrupt has not been detected.
* - 0b1 - Synchronous interrupt has been detected.
*/
/*@{ */
/*! @brief Read current value of the USB_USBTRC0_SYNC_DET field. */
#define USB_RD_USBTRC0_SYNC_DET(base) ((USB_USBTRC0_REG(base) &
USB_USBTRC0_SYNC_DET_MASK) >> USB_USBTRC0_SYNC_DET_SHIFT)
#define USB_BRD_USBTRC0_SYNC_DET(base) (BME_UBFX8(&USB_USBTRC0_REG(base),
USB_USBTRC0_SYNC_DET_SHIFT, USB_USBTRC0_SYNC_DET_WIDTH))

```

```

/*@}*/

/*
 * @name Register USB_USBTRC0, field USBRESMEN[5] (RW)
 *
 * This bit, when set, allows the USB module to send an asynchronous
wakeup
 * event to the MCU upon detection of resume signaling on the USB bus.
The MCU then
 * re-enables clocks to the USB module. It is used for low-power suspend
mode when
 * USB module clocks are stopped or the USB transceiver is in Suspend
mode.
 * Async wakeup only works in device mode.
 *
 * Values:
 * - 0b0 - USB asynchronous wakeup from suspend mode disabled.
 * - 0b1 - USB asynchronous wakeup from suspend mode enabled. The
asynchronous
 * resume interrupt differs from the synchronous resume interrupt in
that it
 * asynchronously detects K-state using the unfiltered state of the
D+ and D-
 * pins. This interrupt should only be enabled when the Transceiver is
suspended.
*/
/*@*/
/*! @brief Read current value of the USB_USBTRC0_USBRESMEN field. */
#define USB_RD_USBTRC0_USBRESMEN(base) ((USB_USBTRC0_REG(base) &
USB_USBTRC0_USBRESMEN_MASK) >> USB_USBTRC0_USBRESMEN_SHIFT)
#define USB_BRD_USBTRC0_USBRESMEN(base)
(BME_UBXF8(&USB_USBTRC0_REG(base), USB_USBTRC0_USBRESMEN_SHIFT,
USB_USBTRC0_USBRESMEN_WIDTH))

/*! @brief Set the USBRESMEN field to a new value. */
#define USB_WR_USBTRC0_USBRESMEN(base, value) (USB_RMW_USBTRC0(base,
USB_USBTRC0_USBRESMEN_MASK, USB_USBTRC0_USBRESMEN(value)))
#define USB_BWR_USBTRC0_USBRESMEN(base, value)
(BME_BFI8(&USB_USBTRC0_REG(base), ((uint8_t)(value) <<
USB_USBTRC0_USBRESMEN_SHIFT), USB_USBTRC0_USBRESMEN_SHIFT,
USB_USBTRC0_USBRESMEN_WIDTH))
/*@*/

/*
 * @name Register USB_USBTRC0, field USBRESET[7] (WO)
 *
 * Generates a hard reset to the USB_OTG module. After this bit is set
and the
 * reset occurs, this bit is automatically cleared. This bit is always
read as
 * zero. Wait two USB clock cycles after setting this bit.
 *
 * Values:
 * - 0b0 - Normal USB module operation.

```

```

* - 0b1 - Returns the USB module to its reset state.
*/
/*@{ */
/*! @brief Set the USBRESET field to a new value. */
#define USB_WR_USBTRC0_USBRESET(base, value) (USB_RMW_USBTRC0(base,
USB_USBTRC0_USBRESET_MASK, USB_USBTRC0_USBRESET(value)))
#define USB_BWR_USBTRC0_USBRESET(base, value)
(USB_WR_USBTRC0_USBRESET(base, value))
/*@} */

***** ****
* USB_USBFRMADJUST - Frame Adjust Register

***** ****
**** */

/*!
* @brief USB_USBFRMADJUST - Frame Adjust Register (RW)
*
* Reset value: 0x00U
*/
/*!
* @name Constants and macros for entire USB_USBFRMADJUST register
*/
/*@{ */
#define USB_RD_USBFRMADJUST(base) (USB_USBFRMADJUST_REG(base))
#define USB_WR_USBFRMADJUST(base, value) (USB_USBFRMADJUST_REG(base) =
(value))
#define USB_RMW_USBFRMADJUST(base, mask, value)
(USB_WR_USBFRMADJUST(base, (USB_RD_USBFRMADJUST(base) & ~mask) | value))
#define USB_SET_USBFRMADJUST(base, value)
(BME_OR8(&USB_USBFRMADJUST_REG(base), (uint8_t)(value)))
#define USB_CLR_USBFRMADJUST(base, value)
(BME_AND8(&USB_USBFRMADJUST_REG(base), (uint8_t)(~value)))
#define USB_TOG_USBFRMADJUST(base, value)
(BME_XOR8(&USB_USBFRMADJUST_REG(base), (uint8_t)(value)))
/*@} */

/* Instance numbers for core modules */
#define JTAG_IDX (0) /*!< Instance number for JTAG. */
#define TPIU_IDX (0) /*!< Instance number for TPIU. */
#define SCB_IDX (0) /*!< Instance number for SCB. */
#define SWD_IDX (0) /*!< Instance number for SWD. */
#define CoreDebug_IDX (0) /*!< Instance number for CoreDebug. */

#if defined(__IAR_SYSTEMS_ICC__)
/* Restore checking of "Error[Pm008]: sections of code should not be
'commented out' (MISRA C 2004 rule 2.4)" */
#pragma diag_default=pm008
#endif

#endif /* __MKL25Z4_EXTENSION_H__ */

```

```
/* EOF */
/*
** ##### Version: rev. 1.0, 2014-05-15
** Build: b141209
**
** Abstract:
**     Common include file for CMSIS register access layer headers.
**
** Copyright (c) 2015 Freescale Semiconductor, Inc.
** All rights reserved.
**
** Redistribution and use in source and binary forms, with or without
** modification,
** are permitted provided that the following conditions are met:
**
**     o Redistributions of source code must retain the above copyright
** notice, this list
**         of conditions and the following disclaimer.
**
**     o Redistributions in binary form must reproduce the above
** copyright notice, this
**         list of conditions and the following disclaimer in the
** documentation and/or
**         other materials provided with the distribution.
**
**     o Neither the name of Freescale Semiconductor, Inc. nor the names
** of its
**         contributors may be used to endorse or promote products derived
** from this
**         software without specific prior written permission.
**
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
** CONTRIBUTORS "AS IS" AND
** ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
** THE IMPLIED
** WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
** ARE
** DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
** BE LIABLE FOR
** ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
** CONSEQUENTIAL DAMAGES
** (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
** SERVICES;
** LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
** CAUSED AND ON
** ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
** TORT
** (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
** USE OF THIS
** SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
** http:          www.freescale.com
** mail:          support@freescale.com
```

```

** 
**     Revisions:
**         - rev. 1.0 (2014-05-15)
**             Customer release.
**
** #####
*/
#ifndef __FSL_DEVICE_REGISTERS_H__
#define __FSL_DEVICE_REGISTERS_H__

/*
 * Include the cpu specific register header files.
 *
 * The CPU macro should be declared in the project or makefile.
 */
#if (defined(CPU_MK02FN128VFM10) || defined(CPU_MK02FN64VFM10) ||
defined(CPU_MK02FN128VLF10) || \
    defined(CPU_MK02FN64VLF10) || defined(CPU_MK02FN128VLH10) || \
defined(CPU_MK02FN64VLH10))

#define K02F12810_SERIES

/* CMSIS-style register definitions */
#include "MK02F12810/include/MK02F12810.h"
/* Extension register definitions */
#include "MK02F12810/include/MK02F12810_extension.h"
/* CPU specific feature definitions */
#include "MK02F12810/include/MK02F12810_features.h"

#elif (defined(CPU_MK10DN512VLK10) || defined(CPU_MK10DN512VLL10) || \
defined(CPU_MK10DX128VLQ10) || \
    defined(CPU_MK10DX256VLQ10) || defined(CPU_MK10DN512VLQ10) || \
defined(CPU_MK10DN512VMC10) || \
    defined(CPU_MK10DX128VMD10) || defined(CPU_MK10DX256VMD10) || \
defined(CPU_MK10DN512VMD10))

#define K10D10_SERIES

/* CMSIS-style register definitions */
#include "MK10D10/include/MK10D10.h"
/* Extension register definitions */
#include "MK10D10/include/MK10D10_extension.h"
/* CPU specific feature definitions */
#include "MK10D10/include/MK10D10_features.h"

#elif (defined(CPU_MK11DX128AVLK5) || defined(CPU_MK11DX256AVLK5) || \
defined(CPU_MK11DN512AVLK5) || \
    defined(CPU_MK11DX128AVMC5) || defined(CPU_MK11DX256AVMC5) || \
defined(CPU_MK11DN512AVMC5))

#define K11DA5_SERIES

/* CMSIS-style register definitions */

```

```

#include "MK11DA5/include/MK11DA5.h"
/* Extension register definitions */
#include "MK11DA5/include/MK11DA5_extension.h"
/* CPU specific feature definitions */
#include "MK11DA5/include/MK11DA5_features.h"

#elif (defined(CPU_MK20DN512VLK10) || defined(CPU_MK20DX256VLK10) ||
defined(CPU_MK20DN512VLL10) || \
defined(CPU_MK20DX256VLL10) || defined(CPU_MK20DX128VLQ10) || \
defined(CPU_MK20DX256VLQ10) || \
defined(CPU_MK20DN512VLQ10) || defined(CPU_MK20DX256VMC10) || \
defined(CPU_MK20DN512VMC10) || \
defined(CPU_MK20DX128VMD10) || defined(CPU_MK20DX256VMD10) || \
defined(CPU_MK20DN512VMD10))

#define K20D10_SERIES

/* CMSIS-style register definitions */
#include "MK20D10/include/MK20D10.h"
/* Extension register definitions */
#include "MK20D10/include/MK20D10_extension.h"
/* CPU specific feature definitions */
#include "MK20D10/include/MK20D10_features.h"

#elif (defined(CPU_MK20DX128VMP5) || defined(CPU_MK20DN128VMP5) ||
defined(CPU_MK20DX64VMP5) || \
defined(CPU_MK20DN64VMP5) || defined(CPU_MK20DX32VMP5) || \
defined(CPU_MK20DN32VMP5) || \
defined(CPU_MK20DX128VLH5) || defined(CPU_MK20DN128VLH5) || \
defined(CPU_MK20DX64VLH5) || \
defined(CPU_MK20DN64VLH5) || defined(CPU_MK20DX32VLH5) || \
defined(CPU_MK20DN32VLH5) || \
defined(CPU_MK20DX128VFM5) || defined(CPU_MK20DN128VFM5) || \
defined(CPU_MK20DX64VFM5) || \
defined(CPU_MK20DN64VFM5) || defined(CPU_MK20DX32VFM5) || \
defined(CPU_MK20DN32VFM5) || \
defined(CPU_MK20DX128VFT5) || defined(CPU_MK20DN128VFT5) || \
defined(CPU_MK20DX64VFT5) || \
defined(CPU_MK20DN64VFT5) || defined(CPU_MK20DX32VFT5) || \
defined(CPU_MK20DN32VFT5) || \
defined(CPU_MK20DX128VLF5) || defined(CPU_MK20DN128VLF5) || \
defined(CPU_MK20DX64VLF5) || \
defined(CPU_MK20DN64VLF5) || defined(CPU_MK20DX32VLF5) || \
defined(CPU_MK20DN32VLF5))

#define K20D5_SERIES

/* CMSIS-style register definitions */
#include "MK20D5/include/MK20D5.h"
/* Extension register definitions */
#include "MK20D5/include/MK20D5_extension.h"
/* CPU specific feature definitions */
#include "MK20D5/include/MK20D5_features.h"

```

```

#elif (defined(CPU_MK21DX128AVLK5) || defined(CPU_MK21DX256AVLK5) ||
defined(CPU_MK21DN512AVLK5) || \
    defined(CPU_MK21DX128AVMC5) || defined(CPU_MK21DX256AVMC5) || \
defined(CPU_MK21DN512AVMC5))

#define K21DA5_SERIES

/* CMSIS-style register definitions */
#include "MK21DA5/include/MK21DA5.h"
/* Extension register definitions */
#include "MK21DA5/include/MK21DA5_extension.h"
/* CPU specific feature definitions */
#include "MK21DA5/include/MK21DA5_features.h"

#elif (defined(CPU_MK21FX512AVLQ12) || defined(CPU_MK21FN1M0AVLQ12) ||
defined(CPU_MK21FX512AVMC12) || \
    defined(CPU_MK21FN1M0AVMC12) || defined(CPU_MK21FX512AVMD12) || \
defined(CPU_MK21FN1M0AVMD12))

#define K21FA12_SERIES

/* CMSIS-style register definitions */
#include "MK21FA12/include/MK21FA12.h"
/* Extension register definitions */
#include "MK21FA12/include/MK21FA12_extension.h"
/* CPU specific feature definitions */
#include "MK21FA12/include/MK21FA12_features.h"

#elif (defined(CPU_MK22FN128VDC10) || defined(CPU_MK22FN128VLH10) ||
defined(CPU_MK22FN128VLL10) || \
defined(CPU_MK22FN128VMP10))

#define K22F12810_SERIES

/* CMSIS-style register definitions */
#include "MK22F12810/include/MK22F12810.h"
/* Extension register definitions */
#include "MK22F12810/include/MK22F12810_extension.h"
/* CPU specific feature definitions */
#include "MK22F12810/include/MK22F12810_features.h"

#elif (defined(CPU_MK22FX512AVLH12) || defined(CPU_MK22FN1M0AVLH12) ||
defined(CPU_MK22FX512AVLK12) || \
defined(CPU_MK22FN1M0AVLK12) || defined(CPU_MK22FX512AVLL12) || \
defined(CPU_MK22FN1M0AVLL12) || \
defined(CPU_MK22FX512AVLQ12) || defined(CPU_MK22FN1M0AVLQ12) || \
defined(CPU_MK22FX512AVMC12) || \
defined(CPU_MK22FN1M0AVMC12) || defined(CPU_MK22FX512AVMD12) || \
defined(CPU_MK22FN1M0AVMD12))

#define K22FA12_SERIES

/* CMSIS-style register definitions */
#include "MK22FA12/include/MK22FA12.h"

```

```

/* Extension register definitions */
#include "MK22FA12/include/MK22FA12_extension.h"
/* CPU specific feature definitions */
#include "MK22FA12/include/MK22FA12_features.h"

#elif (defined(CPU_MK22FN256CAH12) || defined(CPU_MK22FN128CAH12) ||
defined(CPU_MK22FN256VDC12) || \
defined(CPU_MK22FN256VLH12) || defined(CPU_MK22FN256VLL12) || \
defined(CPU_MK22FN256VMP12))

#define K22F25612_SERIES

/* CMSIS-style register definitions */
#include "MK22F25612/include/MK22F25612.h"
/* Extension register definitions */
#include "MK22F25612/include/MK22F25612_extension.h"
/* CPU specific feature definitions */
#include "MK22F25612/include/MK22F25612_features.h"

#elif (defined(CPU_MK22FN512CAP12) || defined(CPU_MK22FN512VDC12) ||
defined(CPU_MK22FN512VLH12) || \
defined(CPU_MK22FN512VLL12) || defined(CPU_MK22FN512VMP12))

#define K22F51212_SERIES

/* CMSIS-style register definitions */
#include "MK22F51212/include/MK22F51212.h"
/* Extension register definitions */
#include "MK22F51212/include/MK22F51212_extension.h"
/* CPU specific feature definitions */
#include "MK22F51212/include/MK22F51212_features.h"

#elif (defined(CPU_MK24FN1M0VDC12) || defined(CPU_MK24FN1M0VLL12) ||
defined(CPU_MK24FN1M0VLQ12))

#define K24F12_SERIES

/* CMSIS-style register definitions */
#include "MK24F12/include/MK24F12.h"
/* Extension register definitions */
#include "MK24F12/include/MK24F12_extension.h"
/* CPU specific feature definitions */
#include "MK24F12/include/MK24F12_features.h"

#elif (defined(CPU_MK24FN256VDC12))

#define K24F25612_SERIES

/* CMSIS-style register definitions */
#include "MK24F25612/include/MK24F25612.h"
/* Extension register definitions */
#include "MK24F25612/include/MK24F25612_extension.h"
/* CPU specific feature definitions */
#include "MK24F25612/include/MK24F25612_features.h"

```

```

#elif (defined(CPU_MK26FN2M0CAC18) || defined(CPU_MK26FN2M0VLQ18) ||
defined(CPU_MK26FN2M0VMD18) || \
defined(CPU_MK26FN2M0VMI18))

#define K26F18_SERIES

/* CMSIS-style register definitions */
#include "MK26F18/include/MK26F18.h"
/* Extension register definitions */
#include "MK26F18/include/MK26F18_extension.h"
/* CPU specific feature definitions */
#include "MK26F18/include/MK26F18_features.h"

#elif (defined(CPU_MK30DN512VLK10) || defined(CPU_MK30DN512VLL10) ||
defined(CPU_MK30DX128VLQ10) || \
defined(CPU_MK30DX256VLQ10) || defined(CPU_MK30DN512VLQ10) || \
defined(CPU_MK30DN512VMC10) || \
defined(CPU_MK30DX128VMD10) || defined(CPU_MK30DX256VMD10) || \
defined(CPU_MK30DN512VMD10))

#define K30D10_SERIES

/* CMSIS-style register definitions */
#include "MK30D10/include/MK30D10.h"
/* Extension register definitions */
#include "MK30D10/include/MK30D10_extension.h"
/* CPU specific feature definitions */
#include "MK30D10/include/MK30D10_features.h"

#elif (defined(CPU_MK40DN512VLK10) || defined(CPU_MK40DN512VLL10) ||
defined(CPU_MK40DX128VLQ10) || \
defined(CPU_MK40DX256VLQ10) || defined(CPU_MK40DN512VLQ10) || \
defined(CPU_MK40DN512VMC10) || \
defined(CPU_MK40DX128VMD10) || defined(CPU_MK40DX256VMD10) || \
defined(CPU_MK40DN512VMD10))

#define K40D10_SERIES

/* CMSIS-style register definitions */
#include "MK40D10/include/MK40D10.h"
/* Extension register definitions */
#include "MK40D10/include/MK40D10_extension.h"
/* CPU specific feature definitions */
#include "MK40D10/include/MK40D10_features.h"

#elif (defined(CPU_MK50DX256CLL10) || defined(CPU_MK50DN512CLL10) ||
defined(CPU_MK50DN512CLQ10) || \
defined(CPU_MK50DX256CMC10) || defined(CPU_MK50DN512CMC10) || \
defined(CPU_MK50DN512CMD10) || \
defined(CPU_MK50DX256CMD10) || defined(CPU_MK50DX256CLK10))

#define K50D10_SERIES

```

```

/* CMSIS-style register definitions */
#include "MK50D10/include/MK50D10.h"
/* Extension register definitions */
#include "MK50D10/include/MK50D10_extension.h"
/* CPU specific feature definitions */
#include "MK50D10/include/MK50D10_features.h"

#elif (defined(CPU_MK51DX256CLL10) || defined(CPU_MK51DN512CLL10) ||
defined(CPU_MK51DN256CLQ10) || \
defined(CPU_MK51DN512CLQ10) || defined(CPU_MK51DX256CMC10) || \
defined(CPU_MK51DN512CMC10) || \
defined(CPU_MK51DN256CMD10) || defined(CPU_MK51DN512CMD10) || \
defined(CPU_MK51DX256CLK10))

#define K51D10_SERIES

/* CMSIS-style register definitions */
#include "MK51D10/include/MK51D10.h"
/* Extension register definitions */
#include "MK51D10/include/MK51D10_extension.h"
/* CPU specific feature definitions */
#include "MK51D10/include/MK51D10_features.h"

#elif (defined(CPU_MK52DN512CLQ10) || defined(CPU_MK52DN512CMD10))

#define K52D10_SERIES

/* CMSIS-style register definitions */
#include "MK52D10/include/MK52D10.h"
/* Extension register definitions */
#include "MK52D10/include/MK52D10_extension.h"
/* CPU specific feature definitions */
#include "MK52D10/include/MK52D10_features.h"

#elif (defined(CPU_MK53DN512CLQ10) || defined(CPU_MK53DX256CLQ10) ||
defined(CPU_MK53DN512CMD10) || \
defined(CPU_MK53DX256CMD10))

#define K53D10_SERIES

/* CMSIS-style register definitions */
#include "MK53D10/include/MK53D10.h"
/* Extension register definitions */
#include "MK53D10/include/MK53D10_extension.h"
/* CPU specific feature definitions */
#include "MK53D10/include/MK53D10_features.h"

#elif (defined(CPU_MK60DN256VLL10) || defined(CPU_MK60DX256VLL10) ||
defined(CPU_MK60DN512VLL10) || \
defined(CPU_MK60DN256VLQ10) || defined(CPU_MK60DX256VLQ10) || \
defined(CPU_MK60DN512VLQ10) || \
defined(CPU_MK60DN256VMC10) || defined(CPU_MK60DX256VMC10) || \
defined(CPU_MK60DN512VMC10) || \

```

```

    defined(CPU_MK60DN256VMD10) || defined(CPU_MK60DX256VMD10) ||
defined(CPU_MK60DN512VMD10))

#define K60D10_SERIES

/* CMSIS-style register definitions */
#include "MK60D10/include/MK60D10.h"
/* Extension register definitions */
#include "MK60D10/include/MK60D10_extension.h"
/* CPU specific feature definitions */
#include "MK60D10/include/MK60D10_features.h"

#elif (defined(CPU_MK63FN1M0VLQ12) || defined(CPU_MK63FN1M0VMD12))

#define K63F12_SERIES

/* CMSIS-style register definitions */
#include "MK63F12/include/MK63F12.h"
/* Extension register definitions */
#include "MK63F12/include/MK63F12_extension.h"
/* CPU specific feature definitions */
#include "MK63F12/include/MK63F12_features.h"

#elif (defined(CPU_MK64FX512VDC12) || defined(CPU_MK64FN1M0VDC12) ||
defined(CPU_MK64FX512VLL12) || \
defined(CPU_MK64FN1M0VLL12) || defined(CPU_MK64FX512VLQ12) || \
defined(CPU_MK64FN1M0VLQ12) || \
defined(CPU_MK64FX512VMD12) || defined(CPU_MK64FN1M0VMD12))

#define K64F12_SERIES
/* CMSIS-style register definitions */
#include "MK64F12/include/MK64F12.h"
/* Extension register definitions */
#include "MK64F12/include/MK64F12_extension.h"
/* CPU specific feature definitions */
#include "MK64F12/include/MK64F12_features.h"

#elif (defined(CPU_MK65FN2M0CAC18) || defined(CPU_MK65FX1M0CAC18) || \
defined(CPU_MK65FN2M0VMI18) || \
defined(CPU_MK65FX1M0VMI18))

#define K65F18_SERIES

/* CMSIS-style register definitions */
#include "MK65F18/include/MK65F18.h"
/* Extension register definitions */
#include "MK65F18/include/MK65F18_extension.h"
/* CPU specific feature definitions */
#include "MK65F18/include/MK65F18_features.h"

#elif (defined(CPU_MK66FN2M0VLQ18) || defined(CPU_MK66FX1M0VLQ18) || \
defined(CPU_MK66FN2M0VMD18) || \
defined(CPU_MK66FX1M0VMD18))

```

```

#define K66F18_SERIES

/* CMSIS-style register definitions */
#include "MK66F18/include/MK66F18.h"
/* Extension register definitions */
#include "MK66F18/include/MK66F18_extension.h"
/* CPU specific feature definitions */
#include "MK66F18/include/MK66F18_features.h"

#elif (defined(CPU_MK70FN1M0VMJ12) || defined(CPU_MK70FX512VMJ12))

#define K70F12_SERIES

/* CMSIS-style register definitions */
#include "MK70F12/include/MK70F12.h"
/* Extension register definitions */
#include "MK70F12/include/MK70F12_extension.h"
/* CPU specific feature definitions */
#include "MK70F12/include/MK70F12_features.h"

#elif (defined(CPU_MK70FN1M0VMJ15) || defined(CPU_MK70FX512VMJ15))

#define K70F15_SERIES

/* CMSIS-style register definitions */
#include "MK70F15/include/MK70F15.h"
/* Extension register definitions */
#include "MK70F15/include/MK70F15_extension.h"
/* CPU specific feature definitions */
#include "MK70F15/include/MK70F15_features.h"

#elif (defined(CPU_MK80FN256CAx15) || defined(CPU_MK80FN256VDC15) ||
      defined(CPU_MK80FN256VLL15) || \
      defined(CPU_MK80FN256VLQ15))

#define K80F25615_SERIES

/* CMSIS-style register definitions */
#include "MK80F25615/include/MK80F25615.h"
/* Extension register definitions */
#include "MK80F25615/include/MK80F25615_extension.h"
/* CPU specific feature definitions */
#include "MK80F25615/include/MK80F25615_features.h"

#elif (defined(CPU_MK81FN256CAx15) || defined(CPU_MK81FN256VDC15) ||
      defined(CPU_MK81FN256VLL15) || \
      defined(CPU_MK81FN256VLQ15))

#define K81F25615_SERIES

/* CMSIS-style register definitions */
#include "MK81F25615/include/MK81F25615.h"
/* Extension register definitions */

```

```

#include "MK81F25615/include/MK81F25615_extension.h"
/* CPU specific feature definitions */
#include "MK81F25615/include/MK81F25615_features.h"

#elif (defined(CPU_MK82FN256CAX15) || defined(CPU_MK82FN256VDC15) ||
defined(CPU_MK82FN256VLL15) || \
defined(CPU_MK82FN256VLQ15))

#define K82F25615_SERIES

/* CMSIS-style register definitions */
#include "MK82F25615/include/MK82F25615.h"
/* Extension register definitions */
#include "MK82F25615/include/MK82F25615_extension.h"
/* CPU specific feature definitions */
#include "MK82F25615/include/MK82F25615_features.h"

#elif (defined(CPU_MKE02Z64VLC2) || defined(CPU_MKE02Z32VLC2) ||
defined(CPU_MKE02Z16VLC2) || \
defined(CPU_MKE02Z64VLD2) || defined(CPU_MKE02Z32VLD2) || \
defined(CPU_MKE02Z16VLD2) || \
defined(CPU_MKE02Z64VLH2) || defined(CPU_MKE02Z64VQH2) || \
defined(CPU_MKE02Z32VLH2) || \
defined(CPU_MKE02Z32VQH2))

#define KE02Z2_SERIES

/* CMSIS-style register definitions */
#include "MKE02Z2/include/MKE02Z2.h"
/* Extension register definitions */
#include "MKE02Z2/include/MKE02Z2_extension.h"
/* CPU specific feature definitions */
#include "MKE02Z2/include/MKE02Z2_features.h"

#elif (defined(CPU_SKEAZN64MLC2) || defined(CPU_SKEAZN32MLC2) ||
defined(CPU_SKEAZN16MLC2) || \
defined(CPU_SKEAZN64MLD2) || defined(CPU_SKEAZN32MLD2) || \
defined(CPU_SKEAZN16MLD2) || \
defined(CPU_SKEAZN64MLH2) || defined(CPU_SKEAZN32MLH2))

#define SKEAZN642_SERIES

/* CMSIS-style register definitions */
#include "SKEAZN642/include/SKEAZN642.h"
/* Extension register definitions */
#include "SKEAZN642/include/SKEAZN642_extension.h"
/* CPU specific feature definitions */
#include "SKEAZN642/include/SKEAZN642_features.h"

#elif (defined(CPU_MKE02Z64VLC4) || defined(CPU_MKE02Z32VLC4) ||
defined(CPU_MKE02Z16VLC4) || \
defined(CPU_MKE02Z64VLD4) || defined(CPU_MKE02Z32VLD4) || \
defined(CPU_MKE02Z16VLD4) || \

```

```

    defined(CPU_MKE02Z64VLH4) || defined(CPU_MKE02Z64VQH4) ||
defined(CPU_MKE02Z32VLH4) || \
defined(CPU_MKE02Z32VQH4))

#define KE02Z4_SERIES

/* CMSIS-style register definitions */
#include "MKE02Z4/include/MKE02Z4.h"
/* Extension register definitions */
#include "MKE02Z4/include/MKE02Z4_extension.h"
/* CPU specific feature definitions */
#include "MKE02Z4/include/MKE02Z4_features.h"

#elif (defined(CPU_MKE04Z128VLD4) || defined(CPU_MKE04Z64VLD4) ||
defined(CPU_MKE04Z128VLK4) || \
defined(CPU_MKE04Z64VLK4) || defined(CPU_MKE04Z128VQH4) ||
defined(CPU_MKE04Z64VQH4) || \
defined(CPU_MKE04Z128VLH4) || defined(CPU_MKE04Z64VLH4))

#define KE04Z1284_SERIES

/* CMSIS-style register definitions */
#include "MKE04Z1284/include/MKE04Z1284.h"
/* Extension register definitions */
#include "MKE04Z1284/include/MKE04Z1284_extension.h"
/* CPU specific feature definitions */
#include "MKE04Z1284/include/MKE04Z1284_features.h"

#elif (defined(CPU_MKE04Z8VFK4) || defined(CPU_MKE04Z8VTG4) ||
defined(CPU_MKE04Z8VWJ4))

#define KE04Z4_SERIES

/* CMSIS-style register definitions */
#include "MKE04Z4/include/MKE04Z4.h"
/* Extension register definitions */
#include "MKE04Z4/include/MKE04Z4_extension.h"
/* CPU specific feature definitions */
#include "MKE04Z4/include/MKE04Z4_features.h"

#elif (defined(CPU_SKEAZN8MFK) || defined(CPU_SKEAZN8MTG))

#define SKEAZN84_SERIES

/* CMSIS-style register definitions */
#include "SKEAZN84/include/SKEAZN84.h"
/* Extension register definitions */
#include "SKEAZN84/include/SKEAZN84_extension.h"
/* CPU specific feature definitions */
#include "SKEAZN84/include/SKEAZN84_features.h"

#elif (defined(CPU_MKE06Z128VLD4) || defined(CPU_MKE06Z64VLD4) ||
defined(CPU_MKE06Z128VLK4) || \

```

```

    defined(CPU_MKE06Z64VLK4) || defined(CPU_MKE06Z128VQH4) ||
defined(CPU_MKE06Z64VQH4) || \
    defined(CPU_MKE06Z128VLH4) || defined(CPU_MKE06Z64VLH4))

#define KE06Z4_SERIES

/* CMSIS-style register definitions */
#include "MKE06Z4/include/MKE06Z4.h"
/* Extension register definitions */
#include "MKE06Z4/include/MKE06Z4_extension.h"
/* CPU specific feature definitions */
#include "MKE06Z4/include/MKE06Z4_features.h"

#elif (defined(CPU_MKL02Z32CAF4) || defined(CPU_MKL02Z8VFG4) ||
defined(CPU_MKL02Z16VFG4) || \
    defined(CPU_MKL02Z32VFG4) || defined(CPU_MKL02Z16VFK4) ||
defined(CPU_MKL02Z32VFK4) || \
    defined(CPU_MKL02Z16VFM4) || defined(CPU_MKL02Z32VFM4))

#define KL02Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL02Z4/include/MKL02Z4.h"
/* Extension register definitions */
#include "MKL02Z4/include/MKL02Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL02Z4/include/MKL02Z4_features.h"

#elif (defined(CPU_MKL03Z32CAF4) || defined(CPU_MKL03Z8VFG4) ||
defined(CPU_MKL03Z16VFG4) || \
    defined(CPU_MKL03Z32VFG4) || defined(CPU_MKL03Z8VFK4) ||
defined(CPU_MKL03Z16VFK4) || \
    defined(CPU_MKL03Z32VFK4))

#define KL03Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL03Z4/include/MKL03Z4.h"
/* Extension register definitions */
#include "MKL03Z4/include/MKL03Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL03Z4/include/MKL03Z4_features.h"

#elif (defined(CPU_MKL04Z8VFK4) || defined(CPU_MKL04Z16VFK4) ||
defined(CPU_MKL04Z32VFK4) || \
    defined(CPU_MKL04Z8VLC4) || defined(CPU_MKL04Z16VLC4) ||
defined(CPU_MKL04Z32VLC4) || \
    defined(CPU_MKL04Z8VFM4) || defined(CPU_MKL04Z16VFM4) ||
defined(CPU_MKL04Z32VFM4) || \
    defined(CPU_MKL04Z16VLF4) || defined(CPU_MKL04Z32VLF4))

#define KL04Z4_SERIES

/* CMSIS-style register definitions */

```

```

#include "MKL04Z4/include/MKL04Z4.h"
/* Extension register definitions */
#include "MKL04Z4/include/MKL04Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL04Z4/include/MKL04Z4_features.h"

#elif (defined(CPU_MKL05Z8VFK4) || defined(CPU_MKL05Z16VFK4) ||
defined(CPU_MKL05Z32VFK4) || \
defined(CPU_MKL05Z8VLC4) || defined(CPU_MKL05Z16VLC4) || \
defined(CPU_MKL05Z32VLC4) || \
defined(CPU_MKL05Z8VFM4) || defined(CPU_MKL05Z16VFM4) || \
defined(CPU_MKL05Z32VFM4) || \
defined(CPU_MKL05Z16VLF4) || defined(CPU_MKL05Z32VLF4))

#define KL05Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL05Z4/include/MKL05Z4.h"
/* Extension register definitions */
#include "MKL05Z4/include/MKL05Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL05Z4/include/MKL05Z4_features.h"

#elif (defined(CPU_MKL13Z32VFM4) || defined(CPU_MKL13Z64VFM4) ||
defined(CPU_MKL13Z32VFT4) || \
defined(CPU_MKL13Z64VFT4) || defined(CPU_MKL13Z32VLH4) || \
defined(CPU_MKL13Z64VLH4) || \
defined(CPU_MKL13Z32VLK4) || defined(CPU_MKL13Z64VLK4) || \
defined(CPU_MKL13Z32VMP4) || \
defined(CPU_MKL13Z64VMP4))

#define KL13Z644_SERIES

/* CMSIS-style register definitions */
#include "MKL13Z644/include/MKL13Z644.h"
/* Extension register definitions */
#include "MKL13Z644/include/MKL13Z644_extension.h"
/* CPU specific feature definitions */
#include "MKL13Z644/include/MKL13Z644_features.h"

#elif (defined(CPU_MKL14Z32VFM4) || defined(CPU_MKL14Z64VFM4) ||
defined(CPU_MKL14Z32VFT4) || \
defined(CPU_MKL14Z64VFT4) || defined(CPU_MKL14Z32VLH4) || \
defined(CPU_MKL14Z64VLH4) || \
defined(CPU_MKL14Z32VLK4) || defined(CPU_MKL14Z64VLK4))

#define KL14Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL14Z4/include/MKL14Z4.h"
/* Extension register definitions */
#include "MKL14Z4/include/MKL14Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL14Z4/include/MKL14Z4_features.h"

```

```

#elif (defined(CPU_MKL15Z128CAD4) || defined(CPU_MKL15Z32VFM4) ||
defined(CPU_MKL15Z64VFM4) || \
    defined(CPU_MKL15Z128VFM4) || defined(CPU_MKL15Z32VFT4) || \
defined(CPU_MKL15Z64VFT4) || \
    defined(CPU_MKL15Z128VFT4) || defined(CPU_MKL15Z32VLH4) || \
defined(CPU_MKL15Z64VLH4) || \
    defined(CPU_MKL15Z128VLH4) || defined(CPU_MKL15Z32VLK4) || \
defined(CPU_MKL15Z64VLK4) || \
    defined(CPU_MKL15Z128VLK4))

#define KL15Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL15Z4/include/MKL15Z4.h"
/* Extension register definitions */
#include "MKL15Z4/include/MKL15Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL15Z4/include/MKL15Z4_features.h"

#elif (defined(CPU_MKL16Z32VFM4) || defined(CPU_MKL16Z64VFM4) ||
defined(CPU_MKL16Z128VFM4) || \
    defined(CPU_MKL16Z32VFT4) || defined(CPU_MKL16Z64VFT4) || \
defined(CPU_MKL16Z128VFT4) || \
    defined(CPU_MKL16Z32VLH4) || defined(CPU_MKL16Z64VLH4) || \
defined(CPU_MKL16Z128VLH4) || \
    defined(CPU_MKL16Z256VLH4) || defined(CPU_MKL16Z256VMP4))

#define KL16Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL16Z4/include/MKL16Z4.h"
/* Extension register definitions */
#include "MKL16Z4/include/MKL16Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL16Z4/include/MKL16Z4_features.h"

#elif (defined(CPU_MKL17Z128VFM4) || defined(CPU_MKL17Z256VFM4) ||
defined(CPU_MKL17Z128VFT4) || \
    defined(CPU_MKL17Z256VFT4) || defined(CPU_MKL17Z128VLH4) || \
defined(CPU_MKL17Z256VLH4) || \
    defined(CPU_MKL17Z128VMP4) || defined(CPU_MKL17Z256VMP4))

#define KL17Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL17Z4/include/MKL17Z4.h"
/* Extension register definitions */
#include "MKL17Z4/include/MKL17Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL17Z4/include/MKL17Z4_features.h"

#elif (defined(CPU_MKL17Z32VDA4) || defined(CPU_MKL17Z64VDA4) ||
defined(CPU_MKL17Z32VFM4) || \

```

```

    defined(CPU_MKL17Z64VFM4) || defined(CPU_MKL17Z32VFT4) ||
defined(CPU_MKL17Z64VFT4) || \
    defined(CPU_MKL17Z32VLH4) || defined(CPU_MKL17Z64VLH4) ||
defined(CPU_MKL17Z32VMP4) || \
    defined(CPU_MKL17Z64VMP4))

#define KL17Z644_SERIES

/* CMSIS-style register definitions */
#include "MKL17Z644/include/MKL17Z644.h"
/* Extension register definitions */
#include "MKL17Z644/include/MKL17Z644_extension.h"
/* CPU specific feature definitions */
#include "MKL17Z644/include/MKL17Z644_features.h"

#elif (defined(CPU_MKL24Z32VFM4) || defined(CPU_MKL24Z64VFM4) ||
defined(CPU_MKL24Z32VFT4) || \
    defined(CPU_MKL24Z64VFT4) || defined(CPU_MKL24Z32VLH4) ||
defined(CPU_MKL24Z64VLH4) || \
    defined(CPU_MKL24Z32VLK4) || defined(CPU_MKL24Z64VLK4))

#define KL24Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL24Z4/include/MKL24Z4.h"
/* Extension register definitions */
#include "MKL24Z4/include/MKL24Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL24Z4/include/MKL24Z4_features.h"

#elif (defined(CPU_MKL25Z32VFM4) || defined(CPU_MKL25Z64VFM4) ||
defined(CPU_MKL25Z128VFM4) || \
    defined(CPU_MKL25Z32VFT4) || defined(CPU_MKL25Z64VFT4) ||
defined(CPU_MKL25Z128VFT4) || \
    defined(CPU_MKL25Z32VLH4) || defined(CPU_MKL25Z64VLH4) ||
defined(CPU_MKL25Z128VLH4) || \
    defined(CPU_MKL25Z32VLK4) || defined(CPU_MKL25Z64VLK4) ||
defined(CPU_MKL25Z128VLK4))

#define KL25Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL25Z4.h"
/* Extension register definitions */
#include "MKL25Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL25Z4_features.h"

#elif (defined(CPU_MKL26Z128CAL4) || defined(CPU_MKL26Z32VFM4) ||
defined(CPU_MKL26Z64VFM4) || \
    defined(CPU_MKL26Z128VFM4) || defined(CPU_MKL26Z32VFT4) ||
defined(CPU_MKL26Z64VFT4) || \

```

```

    defined(CPU_MKL26Z128VFT4) || defined(CPU_MKL26Z32VLH4) ||
defined(CPU_MKL26Z64VLH4) || \
    defined(CPU_MKL26Z128VLH4) || defined(CPU_MKL26Z256VLH4) ||
defined(CPU_MKL26Z128VLL4) || \
    defined(CPU_MKL26Z256VLL4) || defined(CPU_MKL26Z128VMC4) ||
defined(CPU_MKL26Z256VMC4) || \
    defined(CPU_MKL26Z256VMP4))

#define KL26Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL26Z4/include/MKL26Z4.h"
/* Extension register definitions */
#include "MKL26Z4/include/MKL26Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL26Z4/include/MKL26Z4_features.h"

#elif (defined(CPU_MKL27Z128VFM4) || defined(CPU_MKL27Z256VFM4) ||
defined(CPU_MKL27Z128VFT4) || \
    defined(CPU_MKL27Z256VFT4) || defined(CPU_MKL27Z128VLH4) ||
defined(CPU_MKL27Z256VLH4) || \
    defined(CPU_MKL27Z128VMP4) || defined(CPU_MKL27Z256VMP4))

#define KL27Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL27Z4/include/MKL27Z4.h"
/* Extension register definitions */
#include "MKL27Z4/include/MKL27Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL27Z4/include/MKL27Z4_features.h"

#elif (defined(CPU_MKL27Z32VDA4) || defined(CPU_MKL27Z64VDA4) ||
defined(CPU_MKL27Z32VFM4) || \
    defined(CPU_MKL27Z64VFM4) || defined(CPU_MKL27Z32VFT4) ||
defined(CPU_MKL27Z64VFT4) || \
    defined(CPU_MKL27Z32VLH4) || defined(CPU_MKL27Z64VLH4) ||
defined(CPU_MKL27Z32VMP4) || \
    defined(CPU_MKL27Z64VMP4))

#define KL27Z644_SERIES

/* CMSIS-style register definitions */
#include "MKL27Z644/include/MKL27Z644.h"
/* Extension register definitions */
#include "MKL27Z644/include/MKL27Z644_extension.h"
/* CPU specific feature definitions */
#include "MKL27Z644/include/MKL27Z644_features.h"

#elif (defined(CPU_MKL33Z128VLH4) || defined(CPU_MKL33Z256VLH4) ||
defined(CPU_MKL33Z128VMP4) || \
    defined(CPU_MKL33Z256VMP4))

#define KL33Z4_SERIES

```

```

/* CMSIS-style register definitions */
#include "MKL33Z4/include/MKL33Z4.h"
/* Extension register definitions */
#include "MKL33Z4/include/MKL33Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL33Z4/include/MKL33Z4_features.h"

#elif (defined(CPU_MKL33Z32VFT4) || defined(CPU_MKL33Z64VFT4) ||
defined(CPU_MKL33Z32VLH4) || \
defined(CPU_MKL33Z64VLH4) || defined(CPU_MKL33Z32VLK4) || \
defined(CPU_MKL33Z64VLK4) || \
defined(CPU_MKL33Z32VMP4) || defined(CPU_MKL33Z64VMP4))

#define KL33Z644_SERIES

/* CMSIS-style register definitions */
#include "MKL33Z644/include/MKL33Z644.h"
/* Extension register definitions */
#include "MKL33Z644/include/MKL33Z644_extension.h"
/* CPU specific feature definitions */
#include "MKL33Z644/include/MKL33Z644_features.h"

#elif (defined(CPU_MKL34Z64VLH4) || defined(CPU_MKL34Z64VLL4))

#define KL34Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL34Z4/include/MKL34Z4.h"
/* Extension register definitions */
#include "MKL34Z4/include/MKL34Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL34Z4/include/MKL34Z4_features.h"

#elif (defined(CPU_MKL36Z64VLH4) || defined(CPU_MKL36Z128VLH4) ||
defined(CPU_MKL36Z256VLH4) || \
defined(CPU_MKL36Z64VLL4) || defined(CPU_MKL36Z128VLL4) || \
defined(CPU_MKL36Z256VLL4) || \
defined(CPU_MKL36Z128VMC4) || defined(CPU_MKL36Z256VMC4) || \
defined(CPU_MKL36Z256VMP4))

#define KL36Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL36Z4/include/MKL36Z4.h"
/* Extension register definitions */
#include "MKL36Z4/include/MKL36Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL36Z4/include/MKL36Z4_features.h"

#elif (defined(CPU_MKL43Z128VLH4) || defined(CPU_MKL43Z256VLH4) ||
defined(CPU_MKL43Z128VMP4) || \
defined(CPU_MKL43Z256VMP4))

```

```

#define KL43Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL43Z4/include/MKL43Z4.h"
/* Extension register definitions */
#include "MKL43Z4/include/MKL43Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL43Z4/include/MKL43Z4_features.h"

#elif (defined(CPU_MKL46Z128VLH4) || defined(CPU_MKL46Z256VLH4) ||
defined(CPU_MKL46Z128VLL4) || \
defined(CPU_MKL46Z256VLL4) || defined(CPU_MKL46Z128VMC4) || \
defined(CPU_MKL46Z256VMC4) || \
defined(CPU_MKL46Z256VMP4))

#define KL46Z4_SERIES

/* CMSIS-style register definitions */
#include "MKL46Z4/include/MKL46Z4.h"
/* Extension register definitions */
#include "MKL46Z4/include/MKL46Z4_extension.h"
/* CPU specific feature definitions */
#include "MKL46Z4/include/MKL46Z4_features.h"

#elif (defined(CPU_MKM14Z128AHH5) || defined(CPU_MKM14Z64AHH5))

#define KM14ZA5_SERIES

/* CMSIS-style register definitions */
#include "MKM14ZA5/include/MKM14ZA5.h"
/* Extension register definitions */
#include "MKM14ZA5/include/MKM14ZA5_extension.h"
/* CPU specific feature definitions */
#include "MKM14ZA5/include/MKM14ZA5_features.h"

#elif (defined(CPU_MKM33Z128ALH5) || defined(CPU_MKM33Z64ALH5) ||
defined(CPU_MKM33Z128ALL5) || \
defined(CPU_MKM33Z64ALL5))

#define KM33ZA5_SERIES

/* CMSIS-style register definitions */
#include "MKM33ZA5/include/MKM33ZA5.h"
/* Extension register definitions */
#include "MKM33ZA5/include/MKM33ZA5_extension.h"
/* CPU specific feature definitions */
#include "MKM33ZA5/include/MKM33ZA5_features.h"

#elif (defined(CPU_MKM34Z128ALL5))

#define KM34ZA5_SERIES

/* CMSIS-style register definitions */
#include "MKM34ZA5/include/MKM34ZA5.h"

```

```

/* Extension register definitions */
#include "MKM34ZA5/include/MKM34ZA5_extension.h"
/* CPU specific feature definitions */
#include "MKM34ZA5/include/MKM34ZA5_features.h"

#elif (defined(CPU_MKM34Z256VLL7) || defined(CPU_MKM34Z256VLQ7))

#define KM34Z7_SERIES

/* CMSIS-style register definitions */
#include "MKM34Z7/include/MKM34Z7.h"
/* Extension register definitions */
#include "MKM34Z7/include/MKM34Z7_extension.h"
/* CPU specific feature definitions */
#include "MKM34Z7/include/MKM34Z7_features.h"

#elif (defined(CPU_MKV10Z16VFM7) || defined(CPU_MKV10Z16VLC7) ||
      defined(CPU_MKV10Z16VLF7) || \
      defined(CPU_MKV10Z32VFM7) || defined(CPU_MKV10Z32VLC7) || \
      defined(CPU_MKV10Z32VLF7))

#define KV10Z7_SERIES

/* CMSIS-style register definitions */
#include "MKV10Z7/include/MKV10Z7.h"
/* Extension register definitions */
#include "MKV10Z7/include/MKV10Z7_extension.h"
/* CPU specific feature definitions */
#include "MKV10Z7/include/MKV10Z7_features.h"

#elif (defined(CPU_MKV10Z128VFM7) || defined(CPU_MKV10Z128VLC7) ||
      defined(CPU_MKV10Z128VLF7) || \
      defined(CPU_MKV10Z128VLH7) || defined(CPU_MKV10Z64VFM7) || \
      defined(CPU_MKV10Z64VLC7) || \
      defined(CPU_MKV10Z64VLF7) || defined(CPU_MKV10Z64VLH7))

#define KV10Z1287_SERIES

/* CMSIS-style register definitions */
#include "MKV10Z1287/include/MKV10Z1287.h"
/* Extension register definitions */
#include "MKV10Z1287/include/MKV10Z1287_extension.h"
/* CPU specific feature definitions */
#include "MKV10Z1287/include/MKV10Z1287_features.h"

#elif (defined(CPU_MKV11Z128VFM7) || defined(CPU_MKV11Z128VLC7) ||
      defined(CPU_MKV11Z128VLF7) || \
      defined(CPU_MKV11Z128VLH7) || defined(CPU_MKV11Z64VFM7) || \
      defined(CPU_MKV11Z64VLC7) || \
      defined(CPU_MKV11Z64VLF7) || defined(CPU_MKV11Z64VLH7))

#define KV11Z7_SERIES

/* CMSIS-style register definitions */

```

```

#include "MKV11Z7/include/MKV11Z7.h"
/* Extension register definitions */
#include "MKV11Z7/include/MKV11Z7_extension.h"
/* CPU specific feature definitions */
#include "MKV11Z7/include/MKV11Z7_features.h"

#elif (defined(CPU_MKV30F128VFM10) || defined(CPU_MKV30F64VFM10) ||
defined(CPU_MKV30F128VLF10) || \
defined(CPU_MKV30F64VLF10) || defined(CPU_MKV30F128VLH10) || \
defined(CPU_MKV30F64VLH10))

#define KV30F12810_SERIES

/* CMSIS-style register definitions */
#include "MKV30F12810/include/MKV30F12810.h"
/* Extension register definitions */
#include "MKV30F12810/include/MKV30F12810_extension.h"
/* CPU specific feature definitions */
#include "MKV30F12810/include/MKV30F12810_features.h"

#elif (defined(CPU_MKV31F128VLH10) || defined(CPU_MKV31F128VLL10))

#define KV31F12810_SERIES

/* CMSIS-style register definitions */
#include "MKV31F12810/include/MKV31F12810.h"
/* Extension register definitions */
#include "MKV31F12810/include/MKV31F12810_extension.h"
/* CPU specific feature definitions */
#include "MKV31F12810/include/MKV31F12810_features.h"

#elif (defined(CPU_MKV31F256VLH12) || defined(CPU_MKV31F256VLL12))

#define KV31F25612_SERIES

/* CMSIS-style register definitions */
#include "MKV31F25612/include/MKV31F25612.h"
/* Extension register definitions */
#include "MKV31F25612/include/MKV31F25612_extension.h"
/* CPU specific feature definitions */
#include "MKV31F25612/include/MKV31F25612_features.h"

#elif (defined(CPU_MKV31F512VLH12) || defined(CPU_MKV31F512VLL12))

#define KV31F51212_SERIES

/* CMSIS-style register definitions */
#include "MKV31F51212/include/MKV31F51212.h"
/* Extension register definitions */
#include "MKV31F51212/include/MKV31F51212_extension.h"
/* CPU specific feature definitions */
#include "MKV31F51212/include/MKV31F51212_features.h"

```

```

#elif (defined(CPU_MKV40F128VLH15) || defined(CPU_MKV40F128VLL15) ||
defined(CPU_MKV40F256VLH15) || \
defined(CPU_MKV40F256VLL15) || defined(CPU_MKV40F64VLH15))

#define KV40F15_SERIES

/* CMSIS-style register definitions */
#include "MKV40F15/include/MKV40F15.h"
/* Extension register definitions */
#include "MKV40F15/include/MKV40F15_extension.h"
/* CPU specific feature definitions */
#include "MKV40F15/include/MKV40F15_features.h"

#elif (defined(CPU_MKV43F128VLH15) || defined(CPU_MKV43F128VLL15) ||
defined(CPU_MKV43F64VLH15))

#define KV43F15_SERIES

/* CMSIS-style register definitions */
#include "MKV43F15/include/MKV43F15.h"
/* Extension register definitions */
#include "MKV43F15/include/MKV43F15_extension.h"
/* CPU specific feature definitions */
#include "MKV43F15/include/MKV43F15_features.h"

#elif (defined(CPU_MKV44F128VLH15) || defined(CPU_MKV44F128VLL15) ||
defined(CPU_MKV44F64VLH15))

#define KV44F15_SERIES

/* CMSIS-style register definitions */
#include "MKV44F15/include/MKV44F15.h"
/* Extension register definitions */
#include "MKV44F15/include/MKV44F15_extension.h"
/* CPU specific feature definitions */
#include "MKV44F15/include/MKV44F15_features.h"

#elif (defined(CPU_MKV45F128VLH15) || defined(CPU_MKV45F128VLL15) ||
defined(CPU_MKV45F256VLH15) || \
defined(CPU_MKV45F256VLL15))

#define KV45F15_SERIES

/* CMSIS-style register definitions */
#include "MKV45F15/include/MKV45F15.h"
/* Extension register definitions */
#include "MKV45F15/include/MKV45F15_extension.h"
/* CPU specific feature definitions */
#include "MKV45F15/include/MKV45F15_features.h"

#elif (defined(CPU_MKV46F128VLH15) || defined(CPU_MKV46F128VLL15) ||
defined(CPU_MKV46F256VLH15) || \
defined(CPU_MKV46F256VLL15))

```

```

#define KV46F15_SERIES

/* CMSIS-style register definitions */
#include "MKV46F15/include/MKV46F15.h"
/* Extension register definitions */
#include "MKV46F15/include/MKV46F15_extension.h"
/* CPU specific feature definitions */
#include "MKV46F15/include/MKV46F15_features.h"

#elif (defined(CPU_MKW01Z128CHN4))

#define KW01Z4_SERIES

/* CMSIS-style register definitions */
#include "MKW01Z4/include/MKW01Z4.h"
/* Extension register definitions */
#include "MKW01Z4/include/MKW01Z4_extension.h"
/* CPU specific feature definitions */
#include "MKW01Z4/include/MKW01Z4_features.h"

#elif (defined(CPU_MKW20Z160VHT4))

#define KW20Z4_SERIES

/* CMSIS-style register definitions */
#include "MKW20Z4/include/MKW20Z4.h"
/* Extension register definitions */
#include "MKW20Z4/include/MKW20Z4_extension.h"
/* CPU specific feature definitions */
#include "MKW20Z4/include/MKW20Z4_features.h"

#elif (defined(CPU_MKW21D256VHA5) || defined(CPU_MKW21D512VHA5))

#define KW21D5_SERIES

/* CMSIS-style register definitions */
#include "MKW21D5/include/MKW21D5.h"
/* Extension register definitions */
#include "MKW21D5/include/MKW21D5_extension.h"
/* CPU specific feature definitions */
#include "MKW21D5/include/MKW21D5_features.h"

#elif (defined(CPU_MKW22D512VHA5))

#define KW22D5_SERIES

/* CMSIS-style register definitions */
#include "MKW22D5/include/MKW22D5.h"
/* Extension register definitions */
#include "MKW22D5/include/MKW22D5_extension.h"
/* CPU specific feature definitions */
#include "MKW22D5/include/MKW22D5_features.h"

#elif (defined(CPU_MKW24D512VHA5))

```

```

#define KW24D5_SERIES

/* CMSIS-style register definitions */
#include "MKW24D5/include/MKW24D5.h"
/* Extension register definitions */
#include "MKW24D5/include/MKW24D5_extension.h"
/* CPU specific feature definitions */
#include "MKW24D5/include/MKW24D5_features.h"

#elif (defined(CPU_MKW30Z160VHM4))

#define KW30Z4_SERIES

/* CMSIS-style register definitions */
#include "MKW30Z4/include/MKW30Z4.h"
/* Extension register definitions */
#include "MKW30Z4/include/MKW30Z4_extension.h"
/* CPU specific feature definitions */
#include "MKW30Z4/include/MKW30Z4_features.h"

#elif (defined(CPU_MKW40Z160VHT4))

#define KW40Z4_SERIES

/* CMSIS-style register definitions */
#include "MKW40Z4/include/MKW40Z4.h"
/* Extension register definitions */
#include "MKW40Z4/include/MKW40Z4_extension.h"
/* CPU specific feature definitions */
#include "MKW40Z4/include/MKW40Z4_features.h"

#elif (defined(CPU_SKEAZ128MLH) || defined(CPU_SKEAZ64MLH) ||
      defined(CPU_SKEAZ128MLK) || \
      defined(CPU_SKEAZ64MLK))

#define SKEAZ1284_SERIES

/* CMSIS-style register definitions */
#include "SKEAZ1284/include/SKEAZ1284.h"
/* Extension register definitions */
#include "SKEAZ1284/include/SKEAZ1284_extension.h"
/* CPU specific feature definitions */
#include "SKEAZ1284/include/SKEAZ1284_features.h"

#else
    #error "No valid CPU defined!"
#endif

#endif /* __FSL_DEVICE_REGISTERS_H__ */

*****  

* EOF

```

```
*****
*/
/*
 * Copyright (c) 2013 - 2014, Freescale Semiconductor, Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
modification,
 * are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright
notice, this list
 *   of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright
notice, this
 *   list of conditions and the following disclaimer in the documentation
and/or
 *   other materials provided with the distribution.
 *
 * o Neither the name of Freescale Semiconductor, Inc. nor the names of
its
 *   contributors may be used to endorse or promote products derived from
this
 *   software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#ifndef _STARTUP_H_
#define _STARTUP_H_

*****
*/
/*
 * API

```

```
*****
**** */

/*!
 * @brief Make necessary initializations for RAM.
 *
 * - Copy initialized data from ROM to RAM.
 * - Clear the zero-initialized data section.
 * - Copy the vector table from ROM to RAM. This could be an option.
 */
void init_data_bss(void);

#endif /* _STARTUP_H_ */
*****
**** */

***** /



#ifndef _MEMORY_H_
#define _MEMORY_H_

#include <stdint.h>
#include <stdlib.h>

typedef enum {
    NO_ERROR,
    ERROR,
    INVALID_POINTER,
    /*
    NO_OVERLAP,
    SRC_IN_DST_OVERLAP,
    DST_IN_SRC_OVERLAP,
    */
} mem_enum;

typedef enum
{
    ONE_BYTE = 1,
    TWO_BYTE = 2,
    FOUR_BYTE = 0,
} dma_block_size;

/*typedef struct {
    uint8_t *src;
    uint8_t *dst;
    size_t length;
    int8_t value;
}
```

```

        size_t size;
}mem_struct; */

//mem_enum memmove(uint8_t * src, uint8_t * dst, size_t length);
//mem_enum memset(uint8_t * dst, size_t length, uint8_t value);
//mem_enum memmove_overlap(uint8_t * src, uint8_t * dst, size_t length);
mem_enum memmove_dma(uint8_t * src, uint8_t * dst, size_t length,
dma_block_size size);
mem_enum memset_dma(uint8_t * dst, size_t length, uint8_t value,
dma_block_size size);

#endif
#ifndef __NORDIC_H__
#define __NORDIC_H__

#include "MKL25Z4.h"
#include "port.h"
#include "spi.h"

// Active low
#define nrf_chip_enable() GPIOD_PCOR |= (1 << 0); // Enable NRF chip
#define nrf_chip_disable() GPIOD_PSOR |= (1 << 0); // Disable NRF chip
// Active high
#define nrf_transmit_enable() GPIOD_PSOR |= (1 << 5); // Enable NRF
transmission
#define nrf_transmit_disable() GPIOD_PCOR |= (1 << 5); // Disable NRF
transmission

//register address map
#define nrf_CONFIG (0x00)
#define nrf_STATUS (0x07)
#define nrf_TX_ADDR (0x10)
#define nrf_RF_SETUP (0x06)
#define nrf_RF_chregister (0x05)
#define nrf_FIFO_STATUS (0x17)

//Commands
#define nrf_TXFIFO_FLUSH_CMD (0xE1)
#define nrf_RXFIFO_FLUSH_CMD (0xE2)
#define nrf_W_TXPAYLD_CMD (0xA0)
#define nrf_R_RXPAYLD_CMD (0x61)
#define nrf_ACTIVATE_CMD (0x50)
#define nrf_ACTIVATE_DATA (0x73)
#define nrf_RXPAYLD_W_CMD (0x60)
#define nrf_NOP (0xFF)

//masks
#define nrf_POWER_UP_MASK (0x02)
#define nrf_POWER_DOWN_MASK (0X00)
#define nrf_CHANNEL_FREQ_MASK (0x03)
#define nrf_SET_LAN_GAIN_MASK (0x01)

//read write register

```

```

#define READ_INST 0x00
#define WRITE_INST 0x20

/*Read the register and return the value*/
uint8_t nrf_read_register(uint8_t reg);

/*Write to the given register with the data.*/
void nrf_write_register(uint8_t reg, uint8_t value);

/*Reads the STATUS register*/
uint8_t nrf_read_status();

/*Write to CONFIG register*/
void nrf_write_config(uint8_t config);

/*Read the CONFIG register*/
uint8_t nrf_read_config();

/* Reads RF_SETUP register*/
uint8_t nrf_read_rf_setup();

/*Writes to the RF_SETUP register*/
void nrf_write_rf_setup(uint8_t config);

/* Reads RF_CH register*/
uint8_t nrf_read_rf_ch();

/*Writes to the RF_CH register*/
void nrf_write_rf_ch(uint8_t channel);

/*Reads the 5 bytes of the TX_ADDR register*/
void nrf_read_TX_ADDR(uint8_t *address);

/* Writes the 5 byte TX_ADDR register*/
void nrf_write_TX_ADDR(uint8_t *tx_addr);

/* Reads FIFO_STATUS register*/
uint8_t nrf_read_fifo_status();

/*Sends the command FLUSH_TX*/
void nrf_flush_tx_fifo();

/*Sends the command FLUSH_RX*/
void nrf_flush_rx_fifo();

/* Tests reads and writes to some registers over SPI */
void nordic_test();

#endif/*
** ##### Version: rev. 2.10, 2015-05-27
** Build: b150715
**
** Abstract:

```

```
**           Chip specific module features.  
**  
**           Copyright (c) 2015 Freescale Semiconductor, Inc.  
**           All rights reserved.  
**  
**           Redistribution and use in source and binary forms, with or without  
modification,  
**           are permitted provided that the following conditions are met:  
**  
**               o Redistributions of source code must retain the above copyright  
notice, this list  
**                   of conditions and the following disclaimer.  
**  
**               o Redistributions in binary form must reproduce the above  
copyright notice, this  
**                   list of conditions and the following disclaimer in the  
documentation and/or  
**                   other materials provided with the distribution.  
**  
**               o Neither the name of Freescale Semiconductor, Inc. nor the names  
of its  
**                   contributors may be used to endorse or promote products derived  
from this  
**                   software without specific prior written permission.  
**  
**           THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND  
CONTRIBUTORS "AS IS" AND  
**           ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
THE IMPLIED  
**           WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  
ARE  
**           DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS  
BE LIABLE FOR  
**           ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
CONSEQUENTIAL DAMAGES  
**           (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
**           LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON  
**           ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR  
TORT  
**           (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  
USE OF THIS  
**           SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
**  
**           http:                      www.freescale.com  
**           mail:                     support@freescale.com  
**  
**           Revisions:  
**               - rev. 1.0 (2012-06-13)  
**                   Initial version.  
**               - rev. 1.1 (2012-06-21)  
**                   Update according to reference manual rev. 1.  
**               - rev. 1.2 (2012-08-01)
```

```

**           Device type UARTLP changed to UART0.
** - rev. 1.3 (2012-10-04)
**           Update according to reference manual rev. 3.
** - rev. 1.4 (2012-11-22)
**           MCG module - bit LOLS in MCG_S register renamed to LOLSO.
**           NV registers - bit EZPORT_DIS in NV_FOPT register removed.
** - rev. 2.0 (2013-10-29)
**           Register accessor macros added to the memory map.
**           Symbols for Processor Expert memory map compatibility added to
the memory map.
**           Startup file for gcc has been updated according to CMSIS 3.2.
**           System initialization updated.
** - rev. 2.1 (2014-01-30)
**           Added single maximum value generation and a constrain to
varying feature values that only numbers can have maximum.
** - rev. 2.2 (2014-07-16)
**           Module access macro module_BASES replaced by module_BASE_PTRS.
**           System initialization and startup updated.
** - rev. 2.3 (2014-08-22)
**           System initialization updated - default clock config changed.
** - rev. 2.4 (2014-08-28)
**           Update of startup files - possibility to override DefaultISR
added.
** - rev. 2.5 (2014-10-14)
**           Interrupt INT_LPTimer renamed to INT_LPTMR0.
** - rev. 2.6 (2015-01-21)
**           Added FSL_FEATURE_SOC_peripheral_COUNT with number of
peripheral instances
** - rev. 2.7 (2015-02-19)
**           Renamed interrupt vector LLW to LLWU.
** - rev. 2.8 (2015-05-19)
**           FSL_FEATURE_SOC_CAU_COUNT renamed to
FSL_FEATURE_SOC_MMCAU_COUNT.
**           Added FSL_FEATURE_SOC_peripheral_COUNT for TRNG and HSADC.
**           Added features for PORT.
** - rev. 2.9 (2015-05-25)
**           Added FSL_FEATURE_FLASH_PFLASH_START_ADDRESS
** - rev. 2.10 (2015-05-27)
**           Several USB features added.
**
** ##### #####
*/

```

```

#if !defined(__FSL_MKL25Z4_FEATURES_H__)
#define __FSL_MKL25Z4_FEATURES_H__

/* ADC16 module features */

/* @brief Has Programmable Gain Amplifier (PGA) in ADC (register PGA). */
#define FSL_FEATURE_ADC16_HAS_PGA (0)
/* @brief Has PGA chopping control in ADC (bit PGA[PGACHPb] or
PGA[PGACHP]). */
#define FSL_FEATURE_ADC16_HAS_PGA_CHOPPING (0)
/* @brief Has PGA offset measurement mode in ADC (bit PGA[PGAOFSM]). */

```

```

#define FSL_FEATURE_ADC16_HAS_PGA_OFFSET_MEASUREMENT (0)
/* @brief Has DMA support (bit SC2[DMAEN] or SC4[DMAEN]). */
#define FSL_FEATURE_ADC16_HAS_DMA (1)
/* @brief Has differential mode (bitfield SC1x[DIFF]). */
#define FSL_FEATURE_ADC16_HAS_DIFF_MODE (1)
/* @brief Has FIFO (bit SC4[AFDEP]). */
#define FSL_FEATURE_ADC16_HAS_FIFO (0)
/* @brief FIFO size if available (bitfield SC4[AFDEP]). */
#define FSL_FEATURE_ADC16_FIFO_SIZE (0)
/* @brief Has channel set a/b multiplexor (bitfield CFG2[MUXSEL]). */
#define FSL_FEATURE_ADC16_HAS_MUX_SELECT (1)
/* @brief Has HW trigger masking (bitfield SC5[HTRGMASKE]). */
#define FSL_FEATURE_ADC16_HAS_HW_TRIGGER_MASK (0)
/* @brief Has calibration feature (bit SC3[CAL] and registers CLPx,
CLMx). */
#define FSL_FEATURE_ADC16_HAS_CALIBRATION (1)
/* @brief Has HW averaging (bit SC3[AVGE]). */
#define FSL_FEATURE_ADC16_HAS_HW_AVERAGE (1)
/* @brief Has offset correction (register OFS). */
#define FSL_FEATURE_ADC16_HAS_OFFSET_CORRECTION (1)
/* @brief Maximum ADC resolution. */
#define FSL_FEATURE_ADC16_MAX_RESOLUTION (16)
/* @brief Number of SC1x and Rx register pairs (conversion control and
result registers). */
#define FSL_FEATURE_ADC16_CONVERSION_CONTROL_COUNT (2)

/* CMP module features */

/* @brief Has Trigger mode in CMP (register bit field CR1[TRIGM]). */
#define FSL_FEATURE_CMP_HAS_TRIGGER_MODE (1)
/* @brief Has Window mode in CMP (register bit field CR1[WE]). */
#define FSL_FEATURE_CMP_HAS_WINDOW_MODE (0)
/* @brief Has External sample supported in CMP (register bit field
CR1[SE]). */
#define FSL_FEATURE_CMP_HAS_EXTERNAL_SAMPLE_SUPPORT (0)
/* @brief Has DMA support in CMP (register bit field SCR[DMAEN]). */
#define FSL_FEATURE_CMP_HAS_DMA (1)
/* @brief Has Pass Through mode in CMP (register bit field MUXCR[PSTM]). */
*/
#define FSL_FEATURE_CMP_HAS_PASS_THROUGH_MODE (0)
/* @brief Has DAC Test function in CMP (register DACTEST). */
#define FSL_FEATURE_CMP_HAS_DAC_TEST (0)

/* COP module features */

/* @brief Has the COP Debug Enable bit (COPC[COPDBGGEN]) */
#define FSL_FEATURE_COP_HAS_DEBUG_ENABLE (0)
/* @brief Has the COP Stop mode Enable bit (COPC[COPSTPEN]) */
#define FSL_FEATURE_COP_HAS_STOP_ENABLE (0)
/* @brief Has more clock sources like MCGIRC */
#define FSL_FEATURE_COP_HAS_MORE_CLKSRC (0)
/* @brief Has the timeout long and short mode bit (COPC[COPCLKSEL] and
COPC[COPCLKS]) */
#define FSL_FEATURE_COP_HAS_LONGTIME_MODE (0)

```

```
/* SOC module features */

/* @brief ACMP availability on the SoC. */
#define FSL FEATURE SOC_ACMP_COUNT (0)
/* @brief ADC16 availability on the SoC. */
#define FSL FEATURE SOC_ADC16_COUNT (1)
/* @brief AFE availability on the SoC. */
#define FSL FEATURE SOC_AFE_COUNT (0)
/* @brief AIPS availability on the SoC. */
#define FSL FEATURE SOC_AIPS_COUNT (0)
/* @brief AOI availability on the SoC. */
#define FSL FEATURE SOC_AOI_COUNT (0)
/* @brief AXBS availability on the SoC. */
#define FSL FEATURE SOC_AXBS_COUNT (0)
/* @brief CADC availability on the SoC. */
#define FSL FEATURE SOC_CADC_COUNT (0)
/* @brief FLEXCAN availability on the SoC. */
#define FSL FEATURE SOC_FLEXCAN_COUNT (0)
/* @brief MMCAU availability on the SoC. */
#define FSL FEATURE SOC_MMCAU_COUNT (0)
/* @brief CMP availability on the SoC. */
#define FSL FEATURE SOC_CMP_COUNT (1)
/* @brief CMT availability on the SoC. */
#define FSL FEATURE SOC_CMT_COUNT (0)
/* @brief CNC availability on the SoC. */
#define FSL FEATURE SOC_CNC_COUNT (0)
/* @brief CRC availability on the SoC. */
#define FSL FEATURE SOC_CRC_COUNT (0)
/* @brief DAC availability on the SoC. */
#define FSL FEATURE SOC_DAC_COUNT (1)
/* @brief DCDC availability on the SoC. */
#define FSL FEATURE SOC_DCDC_COUNT (0)
/* @brief DDR availability on the SoC. */
#define FSL FEATURE SOC_DDR_COUNT (0)
/* @brief DMA availability on the SoC. */
#define FSL FEATURE SOC_DMA_COUNT (1)
/* @brief DMAMUX availability on the SoC. */
#define FSL FEATURE SOC_DMAMUX_COUNT (1)
/* @brief DRY availability on the SoC. */
#define FSL FEATURE SOC_DRY_COUNT (0)
/* @brief DSPI availability on the SoC. */
#define FSL FEATURE SOC_DSPI_COUNT (0)
/* @brief EDMA availability on the SoC. */
#define FSL FEATURE SOC_EDMA_COUNT (0)
/* @brief EMVSIM availability on the SoC. */
#define FSL FEATURE SOC_EMVSIM_COUNT (0)
/* @brief ENC availability on the SoC. */
#define FSL FEATURE SOC_ENC_COUNT (0)
/* @brief ENET availability on the SoC. */
#define FSL FEATURE SOC_ENET_COUNT (0)
/* @brief EWM availability on the SoC. */
#define FSL FEATURE SOC_EWM_COUNT (0)
/* @brief FB availability on the SoC. */
```

```
#define FSL_FEATURE_SOC_FB_COUNT (0)
/* @brief GPIO availability on the SoC. */
#define FSL_FEATURE_SOC_FGPIO_COUNT (0)
/* @brief FLEXIO availability on the SoC. */
#define FSL_FEATURE_SOC_FLEXIO_COUNT (0)
/* @brief FMC availability on the SoC. */
#define FSL_FEATURE_SOC_FMC_COUNT (0)
/* @brief FSKDT availability on the SoC. */
#define FSL_FEATURE_SOC_FSKDT_COUNT (0)
/* @brief FTFA availability on the SoC. */
#define FSL_FEATURE_SOC_FTFA_COUNT (1)
/* @brief FTFE availability on the SoC. */
#define FSL_FEATURE_SOC_FTFE_COUNT (0)
/* @brief FTFL availability on the SoC. */
#define FSL_FEATURE_SOC_FTFL_COUNT (0)
/* @brief FTM availability on the SoC. */
#define FSL_FEATURE_SOC_FTM_COUNT (0)
/* @brief FTMRA availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRA_COUNT (0)
/* @brief FTMRE availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRE_COUNT (0)
/* @brief FTMRH availability on the SoC. */
#define FSL_FEATURE_SOC_FTMRH_COUNT (0)
/* @brief GPIO availability on the SoC. */
#define FSL_FEATURE_SOC_GPIO_COUNT (5)
/* @brief HSADC availability on the SoC. */
#define FSL_FEATURE_SOC_HSADC_COUNT (0)
/* @brief I2C availability on the SoC. */
#define FSL_FEATURE_SOC_I2C_COUNT (2)
/* @brief I2S availability on the SoC. */
#define FSL_FEATURE_SOC_I2S_COUNT (0)
/* @brief ICS availability on the SoC. */
#define FSL_FEATURE_SOC_ICS_COUNT (0)
/* @brief IRQ availability on the SoC. */
#define FSL_FEATURE_SOC_IRQ_COUNT (0)
/* @brief KBI availability on the SoC. */
#define FSL_FEATURE_SOC_KBI_COUNT (0)
/* @brief SLCD availability on the SoC. */
#define FSL_FEATURE_SOC_SLCD_COUNT (0)
/* @brief LCDC availability on the SoC. */
#define FSL_FEATURE_SOC_LCDC_COUNT (0)
/* @brief LDO availability on the SoC. */
#define FSL_FEATURE_SOC_LDO_COUNT (0)
/* @brief LLWU availability on the SoC. */
#define FSL_FEATURE_SOC_LLWU_COUNT (1)
/* @brief LMEM availability on the SoC. */
#define FSL_FEATURE_SOC_LMEM_COUNT (0)
/* @brief LPSCI availability on the SoC. */
#define FSL_FEATURE_SOC_LPSCI_COUNT (1)
/* @brief LPTMR availability on the SoC. */
#define FSL_FEATURE_SOC_LPTMR_COUNT (1)
/* @brief LPTPM availability on the SoC. */
#define FSL_FEATURE_SOC_LPTPM_COUNT (0)
/* @brief LPUART availability on the SoC. */
```

```
#define FSL_FEATURE_SOC_LPUART_COUNT (0)
/* @brief LTC availability on the SoC. */
#define FSL_FEATURE_SOC_LTC_COUNT (0)
/* @brief MC availability on the SoC. */
#define FSL_FEATURE_SOC_MC_COUNT (0)
/* @brief MCG availability on the SoC. */
#define FSL_FEATURE_SOC_MCG_COUNT (1)
/* @brief MCGLITE availability on the SoC. */
#define FSL_FEATURE_SOC_MCGLITE_COUNT (0)
/* @brief MCM availability on the SoC. */
#define FSL_FEATURE_SOC_MCM_COUNT (1)
/* @brief MMAU availability on the SoC. */
#define FSL_FEATURE_SOC_MMAU_COUNT (0)
/* @brief MMDVSQ availability on the SoC. */
#define FSL_FEATURE_SOC_MMDVSQ_COUNT (0)
/* @brief MPU availability on the SoC. */
#define FSL_FEATURE_SOC_MPUS_COUNT (0)
/* @brief MSCAN availability on the SoC. */
#define FSL_FEATURE_SOC_MSCAN_COUNT (0)
/* @brief MTB availability on the SoC. */
#define FSL_FEATURE_SOC_MTB_COUNT (1)
/* @brief MTBDWT availability on the SoC. */
#define FSL_FEATURE_SOC_MTBDWT_COUNT (1)
/* @brief NFC availability on the SoC. */
#define FSL_FEATURE_SOC_NFC_COUNT (0)
/* @brief OPAMP availability on the SoC. */
#define FSL_FEATURE_SOC_OPAMP_COUNT (0)
/* @brief OSC availability on the SoC. */
#define FSL_FEATURE_SOC_OSC_COUNT (1)
/* @brief OTFAD availability on the SoC. */
#define FSL_FEATURE_SOC_OTFAD_COUNT (0)
/* @brief PDB availability on the SoC. */
#define FSL_FEATURE_SOC_PDB_COUNT (0)
/* @brief PGA availability on the SoC. */
#define FSL_FEATURE_SOC_PGA_COUNT (0)
/* @brief PIT availability on the SoC. */
#define FSL_FEATURE_SOC_PIT_COUNT (1)
/* @brief PMC availability on the SoC. */
#define FSL_FEATURE_SOC_PMC_COUNT (1)
/* @brief PORT availability on the SoC. */
#define FSL_FEATURE_SOC_PORT_COUNT (5)
/* @brief PWM availability on the SoC. */
#define FSL_FEATURE_SOC_PWM_COUNT (0)
/* @brief PWT availability on the SoC. */
#define FSL_FEATURE_SOC_PWT_COUNT (0)
/* @brief QuadSPIO availability on the SoC. */
#define FSL_FEATURE_SOC_QuadSPIO_COUNT (0)
/* @brief RCM availability on the SoC. */
#define FSL_FEATURE_SOC_RCM_COUNT (1)
/* @brief RFSYS availability on the SoC. */
#define FSL_FEATURE_SOC_RFSYS_COUNT (0)
/* @brief RFVBAT availability on the SoC. */
#define FSL_FEATURE_SOC_RFVBAT_COUNT (0)
/* @brief RNG availability on the SoC. */
```

```
#define FSL_FEATURE_SOC_RNG_COUNT (0)
/* @brief RNGB availability on the SoC. */
#define FSL_FEATURE_SOC_RNGB_COUNT (0)
/* @brief ROM availability on the SoC. */
#define FSL_FEATURE_SOC_ROM_COUNT (1)
/* @brief RSIM availability on the SoC. */
#define FSL_FEATURE_SOC_RSIM_COUNT (0)
/* @brief RTC availability on the SoC. */
#define FSL_FEATURE_SOC_RTC_COUNT (1)
/* @brief SCI availability on the SoC. */
#define FSL_FEATURE_SOC_SCI_COUNT (0)
/* @brief SDHC availability on the SoC. */
#define FSL_FEATURE_SOC_SDHC_COUNT (0)
/* @brief SDRAM availability on the SoC. */
#define FSL_FEATURE_SOC_SDRAM_COUNT (0)
/* @brief SIM availability on the SoC. */
#define FSL_FEATURE_SOC_SIM_COUNT (1)
/* @brief SMC availability on the SoC. */
#define FSL_FEATURE_SOC_SMC_COUNT (1)
/* @brief SPI availability on the SoC. */
#define FSL_FEATURE_SOC_SPI_COUNT (2)
/* @brief TMR availability on the SoC. */
#define FSL_FEATURE_SOC_TMR_COUNT (0)
/* @brief TPM availability on the SoC. */
#define FSL_FEATURE_SOC TPM_COUNT (3)
/* @brief TRIAMP availability on the SoC. */
#define FSL_FEATURE_SOC_TRIAMP_COUNT (0)
/* @brief TRNG availability on the SoC. */
#define FSL_FEATURE_SOC_TRNG_COUNT (0)
/* @brief TSI availability on the SoC. */
#define FSL_FEATURE_SOC_TSI_COUNT (1)
/* @brief UART availability on the SoC. */
#define FSL_FEATURE_SOC_UART_COUNT (2)
/* @brief USB availability on the SoC. */
#define FSL_FEATURE_SOC_USB_COUNT (1)
/* @brief USBDCD availability on the SoC. */
#define FSL_FEATURE_SOC_USBDCD_COUNT (0)
/* @brief USBHSDCD availability on the SoC. */
#define FSL_FEATURE_SOC_USBHSDCD_COUNT (0)
/* @brief USBPHY availability on the SoC. */
#define FSL_FEATURE_SOC_USBPHY_COUNT (0)
/* @brief VREF availability on the SoC. */
#define FSL_FEATURE_SOC_VREF_COUNT (0)
/* @brief WDOG availability on the SoC. */
#define FSL_FEATURE_SOC_WDOG_COUNT (0)
/* @brief XBAR availability on the SoC. */
#define FSL_FEATURE_SOC_XBAR_COUNT (0)
/* @brief XCVR availability on the SoC. */
#define FSL_FEATURE_SOC_XCVR_COUNT (0)
/* @brief ZLL availability on the SoC. */
#define FSL_FEATURE_SOC_ZLL_COUNT (0)

/* DAC module features */
```

```

/* @brief Define the size of hardware buffer */
#define FSL_FEATURE_DAC_BUFFER_SIZE (2)
/* @brief Define whether the buffer supports watermark event detection or
not. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_DETECTION (0)
/* @brief Define whether the buffer supports watermark selection
detection or not. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_SELECTION (0)
/* @brief Define whether the buffer supports watermark event 1 word
before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_1_WORD (0)
/* @brief Define whether the buffer supports watermark event 2 words
before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_2_WORDS (0)
/* @brief Define whether the buffer supports watermark event 3 words
before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_3_WORDS (0)
/* @brief Define whether the buffer supports watermark event 4 words
before buffer upper limit. */
#define FSL_FEATURE_DAC_HAS_WATERMARK_4_WORDS (0)
/* @brief Define whether FIFO buffer mode is available or not. */
#define FSL_FEATURE_DAC_HAS_BUFFER_FIFO_MODE (0)
/* @brief Define whether swing buffer mode is available or not.. */
#define FSL_FEATURE_DAC_HAS_BUFFER_SWING_MODE (0)

/* DMA module features */

/* @brief Total number of DMA channels on all modules. */
#define FSL_FEATURE_DMA_DMAMUX_CHANNELS (DMA_INSTANCE_COUNT * 4)

/* DMAMUX module features */

/* @brief Number of DMA channels (related to number of register CHCFGn).
*/
#define FSL_FEATURE_DMAMUX_MODULE_CHANNEL (4)
/* @brief Total number of DMA channels on all modules. */
#define FSL_FEATURE_DMAMUX_DMAMUX_CHANNELS (DMAMUX_INSTANCE_COUNT * 4)
/* @brief Has the periodic trigger capability for the triggered DMA
channel 0 (register bit CHCFG0[TRIG]). */
#define FSL_FEATURE_DMAMUX_HAS_TRIG (1)

/* FLASH module features */

#if defined(CPU_MKL25Z32VFM4) || defined(CPU_MKL25Z32VFT4) ||
defined(CPU_MKL25Z32VLH4) || defined(CPU_MKL25Z32VLK4)
    /* @brief Is of type FTFA. */
    #define FSL_FEATURE_FLASH_IS_FTFA (1)
    /* @brief Is of type FTFE. */
    #define FSL_FEATURE_FLASH_IS_FTFE (0)
    /* @brief Is of type FTFL. */
    #define FSL_FEATURE_FLASH_IS_FTFL (0)
    /* @brief Has flags indicating the status of the FlexRAM (register
bits FCNFG[EEERDY], FCNFG[RAMRDY] and FCNFG[PFLSH]). */
    #define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)

```

```

/* @brief Has program flash swapping status flag (register bit
FCNFG[SWAP]). */
#define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
/* @brief Has EEPROM region protection (register FEPROT). */
#define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
/* @brief Has data flash region protection (register FDPROT). */
#define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
/* @brief Has flash access control (registers XACChn, SACChn, where n
is a number, FACSS and FACSN). */
#define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
/* @brief Has flash cache control in FMC module. */
#define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
/* @brief Has flash cache control in MCM module. */
#define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
/* @brief P-Flash start address. */
#define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
/* @brief P-Flash block count. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
/* @brief P-Flash block size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (32768)
/* @brief P-Flash sector size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
/* @brief P-Flash write unit size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
/* @brief P-Flash data path width. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
/* @brief P-Flash block swap feature. */
#define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
/* @brief Has FlexNVM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
/* @brief FlexNVM start address. (Valid only if FlexNVM is
available.) */
#define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
/* @brief FlexNVM block count. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
/* @brief FlexNVM block size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
/* @brief FlexNVM sector size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)
/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is
available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)

```

```

/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (0)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (0)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Read resource command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT
(0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Read resource command address alignment. */

```

```

#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1011
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */

```

```

#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1111
(0xFFFFFFFF)
/* @brief Emulated eeprom size code 0000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0000
(0xFFFF)
/* @brief Emulated eeprom size code 0001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0001
(0xFFFF)
/* @brief Emulated eeprom size code 0010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0010
(0xFFFF)
/* @brief Emulated eeprom size code 0011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0011
(0xFFFF)
/* @brief Emulated eeprom size code 0100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0100
(0xFFFF)
/* @brief Emulated eeprom size code 0101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0101
(0xFFFF)
/* @brief Emulated eeprom size code 0110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0110
(0xFFFF)
/* @brief Emulated eeprom size code 0111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_0111
(0xFFFF)
/* @brief Emulated eeprom size code 1000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1000
(0xFFFF)
/* @brief Emulated eeprom size code 1001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1001
(0xFFFF)

```

```

/* @brief Emulated eeprom size code 1010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1010
(0xFFFF)
/* @brief Emulated eeprom size code 1011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1011
(0xFFFF)
/* @brief Emulated eeprom size code 1100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1100
(0xFFFF)
/* @brief Emulated eeprom size code 1101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1101
(0xFFFF)
/* @brief Emulated eeprom size code 1110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1110
(0xFFFF)
/* @brief Emulated eeprom size code 1111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1111
(0xFFFF)

#elif defined(CPU_MKL25Z64VFM4) || defined(CPU_MKL25Z64VFT4) ||
defined(CPU_MKL25Z64VLH4) || defined(CPU_MKL25Z64VLK4)
/* @brief Is of type FTFA. */
#define FSL_FEATURE_FLASH_IS_FTFA (1)
/* @brief Is of type FTFE. */
#define FSL_FEATURE_FLASH_IS_FTFE (0)
/* @brief Is of type FTFL. */
#define FSL_FEATURE_FLASH_IS_FTFL (0)
/* @brief Has flags indicating the status of the FlexRAM (register
bits FCNFG[EEERDY], FCNFG[RAMRDY] and FCNFG[PFLSH]). */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)
/* @brief Has program flash swapping status flag (register bit
FCNFG[SWAP]). */
#define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
/* @brief Has EEPROM region protection (register FEPROT). */
#define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
/* @brief Has data flash region protection (register FDPROT). */
#define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
/* @brief Has flash access control (registers XACCHn, SACCHn, where n
is a number, FACSS and FACSN). */
#define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
/* @brief Has flash cache control in FMC module. */
#define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
/* @brief Has flash cache control in MCM module. */
#define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
/* @brief P-Flash start address. */
#define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
/* @brief P-Flash block count. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
/* @brief P-Flash block size. */

```

```

#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (65536)
/* @brief P-Flash sector size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
/* @brief P-Flash write unit size. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
/* @brief P-Flash data path width. */
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
/* @brief P-Flash block swap feature. */
#define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
/* @brief Has FlexNVM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
/* @brief FlexNVM start address. (Valid only if FlexNVM is
available.) */
#define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
/* @brief FlexNVM block count. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
/* @brief FlexNVM block size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
/* @brief FlexNVM sector size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)
/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is
available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)
/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (0)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */

```

```

#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (0)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Read resource command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGMENT
(0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Read resource command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011
(0xFFFFFFFF)

```

```

/* @brief FlexNVM partition code 0100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_0100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_0101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_0110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_0111
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1000
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1011
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1111
(0xFFFFFFFF)
/* @brief Emulated eeprom size code 0000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0000
(0xFFFF)
/* @brief Emulated eeprom size code 0001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */

```

```

#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0001
(0xFFFF)
/* @brief Emulated eeprom size code 0010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0010
(0xFFFF)
/* @brief Emulated eeprom size code 0011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0011
(0xFFFF)
/* @brief Emulated eeprom size code 0100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0100
(0xFFFF)
/* @brief Emulated eeprom size code 0101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0101
(0xFFFF)
/* @brief Emulated eeprom size code 0110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0110
(0xFFFF)
/* @brief Emulated eeprom size code 0111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0111
(0xFFFF)
/* @brief Emulated eeprom size code 1000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1000
(0xFFFF)
/* @brief Emulated eeprom size code 1001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1001
(0xFFFF)
/* @brief Emulated eeprom size code 1010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1010
(0xFFFF)
/* @brief Emulated eeprom size code 1011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1011
(0xFFFF)
/* @brief Emulated eeprom size code 1100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1100
(0xFFFF)
/* @brief Emulated eeprom size code 1101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1101
(0xFFFF)
/* @brief Emulated eeprom size code 1110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1110
(0xFFFF)

```

```

/* @brief Emulated eeprom size code 1111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_EEPROM_SIZE_FOR_EEESIZE_1111
(0xFFFF)
#elif defined(CPU_MKL25Z128VFM4) || defined(CPU_MKL25Z128VFT4) ||
defined(CPU_MKL25Z128VLH4) || defined(CPU_MKL25Z128VLK4)
    /* @brief Is of type FTFA. */
    #define FSL_FEATURE_FLASH_IS_FTFA (1)
    /* @brief Is of type FTFE. */
    #define FSL_FEATURE_FLASH_IS_FTFE (0)
    /* @brief Is of type FTFL. */
    #define FSL_FEATURE_FLASH_IS_FTFL (0)
    /* @brief Has flags indicating the status of the FlexRAM (register
bits FCNFG[EEERDY], FCNFG[RAMRDY] and FCNFG[PFLSH]). */
    #define FSL_FEATURE_FLASH_HAS_FLEX_RAM_FLAGS (0)
    /* @brief Has program flash swapping status flag (register bit
FCNFG[SWAP]). */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_SWAPPING_STATUS_FLAG (0)
    /* @brief Has EEPROM region protection (register FEPROT). */
    #define FSL_FEATURE_FLASH_HAS_EEROM_REGION_PROTECTION (0)
    /* @brief Has data flash region protection (register FDPROT). */
    #define FSL_FEATURE_FLASH_HAS_DATA_FLASH_REGION_PROTECTION (0)
    /* @brief Has flash access control (registers XACChn, SACChn, where n
is a number, FACSS and FACSN). */
    #define FSL_FEATURE_FLASH_HAS_ACCESS_CONTROL (0)
    /* @brief Has flash cache control in FMC module. */
    #define FSL_FEATURE_FLASH_HAS_FMC_FLASH_CACHE_CONTROLS (0)
    /* @brief Has flash cache control in MCM module. */
    #define FSL_FEATURE_FLASH_HAS_MCM_FLASH_CACHE_CONTROLS (1)
    /* @brief P-Flash start address. */
    #define FSL_FEATURE_FLASH_PFLASH_START_ADDRESS (0x00000000)
    /* @brief P-Flash block count. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_COUNT (1)
    /* @brief P-Flash block size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_SIZE (131072)
    /* @brief P-Flash sector size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_SECTOR_SIZE (1024)
    /* @brief P-Flash write unit size. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_WRITE_UNIT_SIZE (4)
    /* @brief P-Flash data path width. */
    #define FSL_FEATURE_FLASH_PFLASH_BLOCK_DATA_PATH_WIDTH (4)
    /* @brief P-Flash block swap feature. */
    #define FSL_FEATURE_FLASH_HAS_PFLASH_BLOCK_SWAP (0)
    /* @brief Has FlexNVM memory. */
    #define FSL_FEATURE_FLASH_HAS_FLEX_NVM (0)
    /* @brief FlexNVM start address. (Valid only if FlexNVM is
available.) */
    #define FSL_FEATURE_FLASH_FLEX_NVM_START_ADDRESS (0x00000000)
    /* @brief FlexNVM block count. */
    #define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_COUNT (0)
    /* @brief FlexNVM block size. */
    #define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SIZE (0)
    /* @brief FlexNVM sector size. */
    #define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_SECTOR_SIZE (0)

```

```

/* @brief FlexNVM write unit size. */
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_WRITE_UNIT_SIZE (0)
/* @brief FlexNVM data path width. */
#define FSL_FEATURE_FLASH_FLEX_BLOCK_DATA_PATH_WIDTH (0)
/* @brief Has FlexRAM memory. */
#define FSL_FEATURE_FLASH_HAS_FLEX_RAM (0)
/* @brief FlexRAM start address. (Valid only if FlexRAM is
available.) */
#define FSL_FEATURE_FLASH_FLEX_RAM_START_ADDRESS (0x00000000)
/* @brief FlexRAM size. */
#define FSL_FEATURE_FLASH_FLEX_RAM_SIZE (0)
/* @brief Has 0x00 Read 1s Block command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_BLOCK_CMD (0)
/* @brief Has 0x01 Read 1s Section command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_SECTION_CMD (1)
/* @brief Has 0x02 Program Check command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_CHECK_CMD (1)
/* @brief Has 0x03 Read Resource command. */
#define FSL_FEATURE_FLASH_HAS_READ_RESOURCE_CMD (1)
/* @brief Has 0x06 Program Longword command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_LONGWORD_CMD (1)
/* @brief Has 0x07 Program Phrase command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PHRASE_CMD (0)
/* @brief Has 0x08 Erase Flash Block command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_BLOCK_CMD (0)
/* @brief Has 0x09 Erase Flash Sector command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_FLASH_SECTOR_CMD (1)
/* @brief Has 0x0B Program Section command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_SECTION_CMD (0)
/* @brief Has 0x40 Read 1s All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_READ_1S_ALL_BLOCKS_CMD (0)
/* @brief Has 0x41 Read Once command. */
#define FSL_FEATURE_FLASH_HAS_READ_ONCE_CMD (1)
/* @brief Has 0x43 Program Once command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_ONCE_CMD (1)
/* @brief Has 0x44 Erase All Blocks command. */
#define FSL_FEATURE_FLASH_HAS_ERASE_ALL_BLOCKS_CMD (0)
/* @brief Has 0x45 Verify Backdoor Access Key command. */
#define FSL_FEATURE_FLASH_HAS_VERIFY_BACKDOOR_ACCESS_KEY_CMD (1)
/* @brief Has 0x46 Swap Control command. */
#define FSL_FEATURE_FLASH_HAS_SWAP_CONTROL_CMD (0)
/* @brief Has 0x80 Program Partition command. */
#define FSL_FEATURE_FLASH_HAS_PROGRAM_PARTITION_CMD (0)
/* @brief Has 0x81 Set FlexRAM Function command. */
#define FSL_FEATURE_FLASH_HAS_SET_FLEXRAM_FUNCTION_CMD (0)
/* @brief P-Flash Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_BLOCK_CMD_ADDRESS_ALIGNMENT (4)
/* @brief P-Flash Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SECTOR_CMD_ADDRESS_ALIGNMENT (4)
/* @brief P-Flash Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_PFLASH_SECTION_CMD_ADDRESS_ALIGNMENT (4)
/* @brief P-Flash Read resource command address alignment. */

```

```

#define FSL_FEATURE_FLASH_PFLASH_RESOURCE_CMD_ADDRESS_ALIGNMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_CHECK_CMD_ADDRESS_ALIGNMENT (4)
/* @brief P-Flash Program check command address alignment. */
#define FSL_FEATURE_FLASH_PFLASH_SWAP_CONTROL_CMD_ADDRESS_ALIGNMENT
(0)
/* @brief FlexNVM Erase/Read 1st all block command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_BLOCK_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM Erase sector command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTOR_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM Rrogram/Verify section command address alignment.
*/
#define FSL_FEATURE_FLASH_FLEX_NVM_SECTION_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM Read resource command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_RESOURCE_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM Program check command address alignment. */
#define FSL_FEATURE_FLASH_FLEX_NVM_CHECK_CMD_ADDRESS_ALIGNMENT (0)
/* @brief FlexNVM partition code 0000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0000
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0011
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 0111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_0111
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1000 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL_FEATURE_FLASH_FLEX_NVM_DFLASH_SIZE_FOR_DEPART_1000
(0xFFFFFFFF)

```

```

/* @brief FlexNVM partition code 1001 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1001
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1010 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1010
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1011 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1011
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1100 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1100
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1101 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1101
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1110 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1110
(0xFFFFFFFF)
/* @brief FlexNVM partition code 1111 mapping to data flash size in
bytes (0xFFFFFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM DFLASH SIZE FOR DEPART_1111
(0xFFFFFFFF)
/* @brief Emulated eeprom size code 0000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0000
(0xFFFF)
/* @brief Emulated eeprom size code 0001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0001
(0xFFFF)
/* @brief Emulated eeprom size code 0010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0010
(0xFFFF)
/* @brief Emulated eeprom size code 0011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0011
(0xFFFF)
/* @brief Emulated eeprom size code 0100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0100
(0xFFFF)
/* @brief Emulated eeprom size code 0101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL FEATURE FLASH FLEX NVM EEPROM SIZE FOR EEESIZE_0101
(0xFFFF)
/* @brief Emulated eeprom size code 0110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */

```

```

#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0110
(0xFFFF)
/* @brief Emulated eeprom size code 0111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_0111
(0xFFFF)
/* @brief Emulated eeprom size code 1000 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1000
(0xFFFF)
/* @brief Emulated eeprom size code 1001 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1001
(0xFFFF)
/* @brief Emulated eeprom size code 1010 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1010
(0xFFFF)
/* @brief Emulated eeprom size code 1011 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1011
(0xFFFF)
/* @brief Emulated eeprom size code 1100 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1100
(0xFFFF)
/* @brief Emulated eeprom size code 1101 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1101
(0xFFFF)
/* @brief Emulated eeprom size code 1110 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1110
(0xFFFF)
/* @brief Emulated eeprom size code 1111 mapping to emulated eeprom
size in bytes (0xFFFF = reserved). */
#define FSL_FEATURE_FLASH_NVM_EEPROM_SIZE_FOR_EEESIZE_1111
(0xFFFF)
#endif

/* GPIO module features */

/* @brief Has fast (single cycle) access capability via a dedicated
memory region. */
#define FSL_FEATURE_GPIO_HAS_FAST_GPIO (1)
/* @brief Has port input disable register (PIDR). */
#define FSL_FEATURE_GPIO_HAS_INPUT_DISABLE (0)
/* @brief Has dedicated interrupt vector. */
#define FSL_FEATURE_GPIO_HAS_INTERRUPT_VECTOR (1)

/* I2C module features */

/* @brief Has System Management Bus support (registers SMB, A2, SLTL and
SLTH). */

```

```

#define FSL_FEATURE_I2C_HAS_SMBUS (1)
/* @brief Maximum supported baud rate in kilobit per second. */
#define FSL_FEATURE_I2C_MAX_BAUD_KBPS (400)
/* @brief Is affected by errata with ID 6070 (repeat start cannot be
generated if the F[MULT] bit field is set to a non-zero value). */
#define FSL_FEATURE_I2C_HAS_ERRATA_6070 (1)
/* @brief Has DMA support (register bit C1[DMAEN]). */
#define FSL_FEATURE_I2C_HAS_DMA_SUPPORT (1)
/* @brief Has I2C bus start and stop detection (register bits FLT[SSIE],
FLT[STARTF] and FLT[STOPF]). */
#define FSL_FEATURE_I2C_HAS_START_STOP_DETECT (0)
/* @brief Has I2C bus stop detection (register bits FLT[STOPIE] and
FLT[STOPF]). */
#define FSL_FEATURE_I2C_HAS_STOP_DETECT (1)
/* @brief Has I2C bus stop hold off (register bit FLT[SHEN]). */
#define FSL_FEATURE_I2C_HAS_STOP_HOLD_OFF (1)
/* @brief Maximum width of the glitch filter in number of bus clocks. */
#define FSL_FEATURE_I2C_MAX_GLITCH_FILTER_WIDTH (31)
/* @brief Has control of the drive capability of the I2C pins. */
#define FSL_FEATURE_I2C_HAS_HIGH_DRIVE_SELECTION (1)
/* @brief Has double buffering support (register S2). */
#define FSL_FEATURE_I2C_HAS_DOUBLE_BUFFERING (0)

/* LLWU module features */

/* @brief Maximum number of pins (maximal index plus one) connected to
LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN (16)
/* @brief Has pins 8-15 connected to LLWU device. */
#define FSL_FEATURE_LLWU_EXTERNAL_PIN_GROUP2 (1)
/* @brief Maximum number of internal modules connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE (8)
/* @brief Number of digital filters. */
#define FSL_FEATURE_LLWU_HAS_PIN_FILTER (2)
/* @brief Has MF5 register. */
#define FSL_FEATURE_LLWU_HAS_MF (0)
/* @brief Has possibility to enable reset in low leakage power mode and
enable digital filter for RESET pin (register LLWU_RST). */
#define FSL_FEATURE_LLWU_HAS_RESET_ENABLE (0)
/* @brief Has external pin 0 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN0 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN0_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN0_GPIO_PIN (0)
/* @brief Has external pin 1 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN1 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN1_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN1_GPIO_PIN (0)
/* @brief Has external pin 2 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN2 (0)
/* @brief Index of port of external pin. */

```

```

#define FSL_FEATURE_LLWU_PIN2_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN2_GPIO_PIN (0)
/* @brief Has external pin 3 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN3 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN3_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN3_GPIO_PIN (0)
/* @brief Has external pin 4 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN4 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN4_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN4_GPIO_PIN (0)
/* @brief Has external pin 5 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN5 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN5_GPIO_IDX (GPIOB_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN5_GPIO_PIN (0)
/* @brief Has external pin 6 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN6 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN6_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN6_GPIO_PIN (1)
/* @brief Has external pin 7 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN7 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN7_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN7_GPIO_PIN (3)
/* @brief Has external pin 8 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN8 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN8_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN8_GPIO_PIN (4)
/* @brief Has external pin 9 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN9 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN9_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN9_GPIO_PIN (5)
/* @brief Has external pin 10 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN10 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN10_GPIO_IDX (GPIOC_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN10_GPIO_PIN (6)
/* @brief Has external pin 11 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN11 (0)
/* @brief Index of port of external pin. */

```

```

#define FSL_FEATURE_LLWU_PIN11_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN11_GPIO_PIN (0)
/* @brief Has external pin 11 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN12 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN12_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN12_GPIO_PIN (0)
/* @brief Has external pin 12 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN13 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN13_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN13_GPIO_PIN (0)
/* @brief Has external pin 13 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN14 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN14_GPIO_IDX (GPIOD_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN14_GPIO_PIN (4)
/* @brief Has external pin 14 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN15 (1)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN15_GPIO_IDX (GPIOE_IDX)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN15_GPIO_PIN (6)
/* @brief Has external pin 15 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN16 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN16_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN16_GPIO_PIN (0)
/* @brief Has external pin 16 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN17 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN17_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN17_GPIO_PIN (0)
/* @brief Has external pin 17 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN18 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN18_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN18_GPIO_PIN (0)
/* @brief Has external pin 18 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN19 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN19_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN19_GPIO_PIN (0)
/* @brief Has external pin 19 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN20 (0)
/* @brief Index of port of external pin. */

```

```

#define FSL_FEATURE_LLWU_PIN20_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN20_GPIO_PIN (0)
/* @brief Has external pin 21 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN21 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN21_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN21_GPIO_PIN (0)
/* @brief Has external pin 22 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN22 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN22_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN22_GPIO_PIN (0)
/* @brief Has external pin 23 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN23 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN23_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN23_GPIO_PIN (0)
/* @brief Has external pin 24 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN24 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN24_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN24_GPIO_PIN (0)
/* @brief Has external pin 25 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN25 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN25_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN25_GPIO_PIN (0)
/* @brief Has external pin 26 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN26 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN26_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN26_GPIO_PIN (0)
/* @brief Has external pin 27 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN27 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN27_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN27_GPIO_PIN (0)
/* @brief Has external pin 28 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN28 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN28_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN28_GPIO_PIN (0)
/* @brief Has external pin 29 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN29 (0)
/* @brief Index of port of external pin. */

```

```

#define FSL_FEATURE_LLWU_PIN29_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN29_GPIO_PIN (0)
/* @brief Has external pin 30 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN30 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN30_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN30_GPIO_PIN (0)
/* @brief Has external pin 31 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_EXTERNAL_PIN31 (0)
/* @brief Index of port of external pin. */
#define FSL_FEATURE_LLWU_PIN31_GPIO_IDX (0)
/* @brief Number of external pin port on specified port. */
#define FSL_FEATURE_LLWU_PIN31_GPIO_PIN (0)
/* @brief Has internal module 0 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE0 (1)
/* @brief Has internal module 1 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE1 (1)
/* @brief Has internal module 2 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE2 (0)
/* @brief Has internal module 3 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE3 (0)
/* @brief Has internal module 4 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE4 (1)
/* @brief Has internal module 5 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE5 (1)
/* @brief Has internal module 6 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE6 (0)
/* @brief Has internal module 7 connected to LLWU device. */
#define FSL_FEATURE_LLWU_HAS_INTERNAL_MODULE7 (1)

/* LPTMR module features */

/* @brief Has shared interrupt handler with another LPTMR module. */
#define FSL_FEATURE_LPTMR_HAS_SHARED_IRQ_HANDLER (0)

/* MCG module features */

/* @brief PRDIV base value (divider of register bit field [PRDIV] zero
value). */
#define FSL_FEATURE_MCG_PLL_PRDIV_BASE (1)
/* @brief Maximum PLL external reference divider value (max. value of
register bit field C5[PRVDIV]). */
#define FSL_FEATURE_MCG_PLL_PRDIV_MAX (24)
/* @brief VCO divider base value (multiply factor of register bit field
C6[VDIV] zero value). */
#define FSL_FEATURE_MCG_PLL_VDIV_BASE (24)
/* @brief PLL reference clock low range. OSCCLK/PLL_R. */
#define FSL_FEATURE_MCG_PLL_REF_MIN (2000000)
/* @brief PLL reference clock high range. OSCCLK/PLL_R. */
#define FSL_FEATURE_MCG_PLL_REF_MAX (4000000)
/* @brief The PLL clock is divided by 2 before VCO divider. */
#define FSL_FEATURE_MCG_HAS_PLL_INTERNAL_DIV (0)

```

```

/* @brief FRDIV supports 1280. */
#define FSL FEATURE_MCG_FRDIV_SUPPORT_1280 (1)
/* @brief FRDIV supports 1536. */
#define FSL FEATURE_MCG_FRDIV_SUPPORT_1536 (1)
/* @brief Is PLL clock divided by 2 before MCG PLL/FLL clock selection in
the SIM module. */
#define FSL FEATURE_MCG_HAS_PLL_EXTRA_DIV (1)
/* @brief Has 32kHz RTC external reference clock (register bits
C8[LOCS1], C8[CME1], C8[LOCRE1] and RTC module are present). */
#define FSL FEATURE_MCG_HAS_RTC_32K (0)
/* @brief Has PLL1 external reference clock (registers C10, C11, C12,
S2). */
#define FSL FEATURE_MCG_HAS_PLL1 (0)
/* @brief Has 48MHz internal oscillator. */
#define FSL FEATURE_MCG_HAS_IRC_48M (0)
/* @brief Has OSC1 external oscillator (registers C10, C11, C12, S2). */
#define FSL FEATURE_MCG_HAS_OSC1 (0)
/* @brief Has fast internal reference clock fine trim (register bit
C2[FCFTRIM]). */
#define FSL FEATURE_MCG_HAS_FCFTRIM (0)
/* @brief Has PLL loss of lock reset (register bit C8[LOLRE]). */
#define FSL FEATURE_MCG_HAS_LOLRE (1)
/* @brief Has MCG OSC clock selection (register bit C7[OSCSEL]). */
#define FSL FEATURE_MCG_USE_OSCSEL (0)
/* @brief Has PLL external reference selection (register bits
C5[PLLREFSEL0] and C11[PLLREFSEL1]). */
#define FSL FEATURE_MCG_USE_PLLREFSEL (0)
/* @brief TBD */
#define FSL FEATURE_MCG_USE_SYSTEM_CLOCK (0)
/* @brief Has phase-locked loop (PLL) (register C5 and bits C6[VDIV],
C6[PLLS], C6[LOLIE0], S[PLLST], S[LOCK0], S[LOLS]). */
#define FSL FEATURE_MCG_HAS_PLL (1)
/* @brief Has phase-locked loop (PLL) PRDIV (register C5[PRDIV]. */
#define FSL FEATURE_MCG_HAS_PLL_PRDIV (1)
/* @brief Has phase-locked loop (PLL) VDIV (register C6[VDIV]. */
#define FSL FEATURE_MCG_HAS_PLL_VDIV (1)
/* @brief PLL/OSC related register bit fields have PLL/OSC index in their
name. */
#define FSL FEATURE_MCG_HAS_PLL_OSC_INDEX (1)
/* @brief Has frequency-locked loop (FLL) (register ATCVH, ATCVL and bits
C1[IREFS], C1[FRDIV]). */
#define FSL FEATURE_MCG_HAS_FLL (1)
/* @brief Has PLL external to MCG (C9[PLL_CME], C9[PLL_LOCRE],
C9[EXT_PLL_LOCS]). */
#define FSL FEATURE_MCG_HAS_EXTERNAL_PLL (0)
/* @brief Has crystal oscillator or external reference clock low power
controls (register bits C2[HGO], C2[RANGE]). */
#define FSL FEATURE_MCG_HAS_EXT_REF_LOW_POWER_CONTROL (1)
/* @brief Has PLL/FLL selection as MCG output (register bit C6[PLLS]). */
#define FSL FEATURE_MCG_HAS_PLL_FLL_SELECTION (1)
/* @brief Has PLL output selection (PLL0/PLL1, PLL/external PLL)
(register bit C11[PLLCS]). */
#define FSL FEATURE_MCG_HAS_PLL_OUTPUT_SELECTION (0)

```

```

/* @brief Has automatic trim machine (registers ATCVH, ATCVL and bits
SC[ATMF], SC[ATMS], SC[ATME]). */
#define FSL_FEATURE_MCG_HAS_AUTO_TRIM_MACHINE (1)
/* @brief Has external clock monitor (register bit C6[CME]). */
#define FSL_FEATURE_MCG_HAS_EXTERNAL_CLOCK_MONITOR (1)
/* @brief Has low frequency internal reference clock (IRC) (registers
LTRIMRNG, LFRIM, LSTRIM and bit MC[LIRC_DIV2]). */
#define FSL_FEATURE_MCG_HAS_LOW_FREQ_IRC (0)
/* @brief Has high frequency internal reference clock (IRC) (registers
HCTRIM, HTTRIM, HFTRIM and bit MC[HIRCEN]). */
#define FSL_FEATURE_MCG_HAS_HIGH_FREQ_IRC (0)
/* @brief Has PEI mode or PBI mode. */
#define FSL_FEATURE_MCG_HAS_PLL_INTERNAL_MODE (0)
/* @brief Reset clock mode is BLPI. */
#define FSL_FEATURE_MCG_RESET_IS_BLPI (0)

/* interrupt module features */

/* @brief Lowest interrupt request number. */
#define FSL_FEATURE_INTERRUPT_IRQ_MIN (-14)
/* @brief Highest interrupt request number. */
#define FSL_FEATURE_INTERRUPT_IRQ_MAX (31)

/* OSC module features */

/* @brief Has OSC1 external oscillator. */
#define FSL_FEATURE_OSC_HAS_OSC1 (0)
/* @brief Has OSC0 external oscillator. */
#define FSL_FEATURE_OSC_HAS_OSC0 (1)
/* @brief Has OSC external oscillator (without index). */
#define FSL_FEATURE_OSC_HAS_OSC (0)
/* @brief Number of OSC external oscillators. */
#define FSL_FEATURE_OSC_OSC_COUNT (1)
/* @brief Has external reference clock divider (register bit field
DIV[ERPS]). */
#define FSL_FEATURE_OSC_HAS_EXT_REF_CLOCK_DIVIDER (0)

/* PIT module features */

/* @brief Number of channels (related to number of registers LDVALn,
CVALn, TCTRLn, TFLGn). */
#define FSL_FEATURE_PIT_TIMER_COUNT (2)
/* @brief Has lifetime timer (related to existence of registers LTMR64L
and LTMR64H). */
#define FSL_FEATURE_PIT_HAS_LIFETIME_TIMER (1)
/* @brief Has chain mode (related to existence of register bit field
TCTRLn[CHN]). */
#define FSL_FEATURE_PIT_HAS_CHAIN_MODE (1)
/* @brief Has shared interrupt handler (has not individual interrupt
handler for each channel). */
#define FSL_FEATURE_PIT_HAS_SHARED_IRQ_HANDLER (1)

/* PMC module features */

```

```

/* @brief Has Bandgap Enable In VLPx Operation support. */
#define FSL_FEATURE_PMC_HAS_BGEN (1)
/* @brief Has Bandgap Buffer Drive Select. */
#define FSL_FEATURE_PMC_HAS_BGBDS (0)

/* PORT module features */

/* @brief Has control lock (register bit PCR[LK]). */
#define FSL_FEATURE_PORT_HAS_PIN_CONTROL_LOCK (0)
/* @brief Has open drain control (register bit PCR[ODE]). */
#define FSL_FEATURE_PORT_HAS_OPEN_DRAIN (0)
/* @brief Has digital filter (registers DFER, DFCR and DFWR). */
#define FSL_FEATURE_PORT_HAS_DIGITAL_FILTER (0)
/* @brief Has DMA request (register bit field PCR[IRQC] values). */
#define FSL_FEATURE_PORT_HAS_DMA_REQUEST (1)
/* @brief Has pull resistor selection available. */
#define FSL_FEATURE_PORT_HAS_PULL_SELECTION (0)
/* @brief Has pull resistor enable (register bit PCR[PE]). */
#define FSL_FEATURE_PORT_HAS_PULL_ENABLE (1)
/* @brief Has slew rate control (register bit PCR[SRE]). */
#define FSL_FEATURE_PORT_HAS_SLEW_RATE (1)
/* @brief Has passive filter (register bit field PCR[PFE]). */
#define FSL_FEATURE_PORT_HAS_PASSIVE_FILTER (1)
/* @brief Has drive strength control (register bit PCR[DSE]). */
#define FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH (1)
/* @brief Has separate drive strength register (HDRVE). */
#define FSL_FEATURE_PORT_HAS_DRIVE_STRENGTH_REGISTER (0)
/* @brief Has glitch filter (register IOFLT). */
#define FSL_FEATURE_PORT_HAS_GLITCH_FILTER (0)
/* @brief Defines width of PCR[MUX] field. */
#define FSL_FEATURE_PORT_PCR_MUX_WIDTH (3)
/* @brief Defines whether PCR[IRQC] bit-field has flag states. */
#define FSL_FEATURE_PORT_HAS_IRQC_FLAG (0)
/* @brief Defines whether PCR[IRQC] bit-field has trigger states. */
#define FSL_FEATURE_PORT_HAS_IRQC_TRIGGER (0)

/* RCM module features */

/* @brief Has Loss-of-Lock Reset support. */
#define FSL_FEATURE_RCM_HAS_LOL (1)
/* @brief Has Loss-of-Clock Reset support. */
#define FSL_FEATURE_RCM_HAS_LOC (1)
/* @brief Has JTAG generated Reset support. */
#define FSL_FEATURE_RCM_HAS_JTAG (0)
/* @brief Has EzPort generated Reset support. */
#define FSL_FEATURE_RCM_HAS_EZPORT (0)
/* @brief Has bit-field indicating EZP_MS_B pin state during last reset. */
#define FSL_FEATURE_RCM_HAS_EZPMS (0)
/* @brief Has boot ROM configuration, MR[BOOTROM], FM[FORCEROM] */
#define FSL_FEATURE_RCM_HAS_BOOTROM (0)
/* @brief Has sticky system reset status register RCM_SSRS0 and
RCM_SSRS1. */
#define FSL_FEATURE_RCM_HAS_SSRS (0)

```

```

/* RTC module features */

/* @brief Has wakeup pin (bit field CR[WPS]). */
#define FSL FEATURE RTC HAS WAKEUP PIN (1)
/* @brief Has low power features (registers MER, MCLR and MCHR). */
#define FSL FEATURE RTC HAS MONOTONIC (0)
/* @brief Has read/write access control (registers WAR and RAR). */
#define FSL FEATURE RTC HAS ACCESS CONTROL (0)
/* @brief Has security features (registers TTSR, MER, MCLR and MCHR). */
#define FSL FEATURE RTC HAS SECURITY (0)

/* SIM module features */

/* @brief Has USB FS divider. */
#define FSL FEATURE SIM USBFS USE SPECIAL DIVIDER (0)
/* @brief Is PLL clock divided by 2 before MCG PLL/FLL clock selection.
*/
#define FSL FEATURE SIM PLLCLK USE SPECIAL DIVIDER (1)
/* @brief Has RAM size specification (register bit field SOPT1[RAMSIZE]). */
#define FSL FEATURE SIM OPT HAS RAMSIZE (0)
/* @brief Has 32k oscillator clock output (register bit
SOPT1[OSC32KOUT]). */
#define FSL FEATURE SIM OPT HAS OSC32K_OUT (0)
/* @brief Has 32k oscillator clock selection (register bit field
SOPT1[OSC32KSEL]). */
#define FSL FEATURE SIM OPT HAS OSC32K_SELECTION (1)
/* @brief 32k oscillator clock selection width (width of register bit
field SOPT1[OSC32KSEL]). */
#define FSL FEATURE SIM OPT OSC32K_SELECTION_WIDTH (2)
/* @brief Has RTC clock output selection (register bit
SOPT2[RTCCLKOUTSEL]). */
#define FSL FEATURE SIM OPT HAS RTC_CLOCK_OUT_SELECTION (1)
/* @brief Has USB voltage regulator (register bits SOPT1[USBVSTBY],
SOPT1[USBSSTBY], SOPT1[USBREGEN], SOPT1CFG[URWE], SOPT1CFG[UVSWE],
SOPT1CFG[USSWE]). */
#define FSL FEATURE SIM OPT HAS USB_VOLTAGE_REGULATOR (1)
/* @brief USB has integrated PHY (register bits USBPHYCTL[USBVREGSEL],
USBPHYCTL[USBVREGPD], USBPHYCTL[USB3VOUTTRG], USBPHYCTL[USBDISILIM],
SOPT2[USBSLSRC], SOPT2[USBREGEN]). */
#define FSL FEATURE SIM OPT HAS USB_PHY (0)
/* @brief Has PTD7 pad drive strength control (register bit
SOPT2[PTD7PAD]). */
#define FSL FEATURE SIM OPT HAS PTD7PAD (0)
/* @brief Has FlexBus security level selection (register bit
SOPT2[FBSL]). */
#define FSL FEATURE SIM OPT HAS FBSL (0)
/* @brief Has number of FlexBus hold cycle before FlexBus can release bus
(register bit SOPT6[PCR]). */
#define FSL FEATURE SIM OPT HAS PCR (0)
/* @brief Has number of NFC hold cycle in case of FlexBus request
(register bit SOPT6[MCC]). */
#define FSL FEATURE SIM OPT HAS MCC (0)

```

```

/* @brief Has UART open drain enable (register bits UARTnODE, where n is
a number, in register SOPT5). */
#define FSL_FEATURE_SIM_OPT_HAS_ODE (1)
/* @brief Number of LPUART modules (number of register bits LPUARTn,
where n is a number, in register SCGC5). */
#define FSL_FEATURE_SIM_OPT_LPUART_COUNT (0)
/* @brief Number of UART modules (number of register bits UARTn, where n
is a number, in register SCGC4). */
#define FSL_FEATURE_SIM_OPT_UART_COUNT (3)
/* @brief Has UART0 open drain enable (register bit SOPT5[UART0ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_ODE (1)
/* @brief Has UART1 open drain enable (register bit SOPT5[UART1ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_ODE (1)
/* @brief Has UART2 open drain enable (register bit SOPT5[UART2ODE]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART2_ODE (1)
/* @brief Has LPUART0 open drain enable (register bit SOPT5[LPUART0ODE]). */
*/
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_ODE (0)
/* @brief Has LPUART1 open drain enable (register bit SOPT5[LPUART1ODE]). */
*/
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_ODE (0)
/* @brief Has CMT/UART pad drive strength control (register bit
SOPT2[CMTUARTPAD]). */
#define FSL_FEATURE_SIM_OPT_HAS_CMTUARTPAD (0)
/* @brief Has LPUART0 transmit data source selection (register bit
SOPT5[LPUART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_TX_SRC (0)
/* @brief Has LPUART0 receive data source selection (register bit
SOPT5[LPUART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0_RX_SRC (0)
/* @brief Has LPUART1 transmit data source selection (register bit
SOPT5[LPUART1TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_TX_SRC (0)
/* @brief Has LPUART1 receive data source selection (register bit
SOPT5[LPUART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1_RX_SRC (0)
/* @brief Has UART0 transmit data source selection (register bit
SOPT5[UART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_TX_SRC (1)
/* @brief UART0 transmit data source selection width (width of register
bit SOPT5[UART0TXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART0_TX_SRC_WIDTH (2)
/* @brief Has UART0 receive data source selection (register bit
SOPT5[UART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0_RX_SRC (1)
/* @brief UART0 receive data source selection width (width of register
bit SOPT5[UART0RXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART0_RX_SRC_WIDTH (1)
/* @brief Has UART1 transmit data source selection (register bit
SOPT5[UART1TXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_TX_SRC (1)
/* @brief Has UART1 receive data source selection (register bit
SOPT5[UART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART1_RX_SRC (1)

```

```

/* @brief UART1 receive data source selection width (width of register
bit SOPT5[UART1RXSRC]). */
#define FSL_FEATURE_SIM_OPT_UART1_RX_SRC_WIDTH (1)
/* @brief Has FTM module(s) configuration. */
#define FSL_FEATURE_SIM_OPT_HAS_FTM (0)
/* @brief Number of FTM modules. */
#define FSL_FEATURE_SIM_OPT_FTM_COUNT (0)
/* @brief Number of FTM triggers with selectable source. */
#define FSL_FEATURE_SIM_OPT_FTM_TRIGGER_COUNT (0)
/* @brief Has FTM0 triggers source selection (register bits
SOPT4[FTM0TRGnSRC], where n is a number). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM0_TRIGGER (0)
/* @brief Has FTM3 triggers source selection (register bits
SOPT4[FTM3TRGnSRC], where n is a number). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM3_TRIGGER (0)
/* @brief Has FTM1 channel 0 input capture source selection (register bit
SOPT4[FTM1CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM1_CHANNELS (0)
/* @brief Has FTM2 channel 0 input capture source selection (register bit
SOPT4[FTM2CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM2_CHANNELS (0)
/* @brief Has FTM3 channel 0 input capture source selection (register bit
SOPT4[FTM3CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM3_CHANNELS (0)
/* @brief Has FTM2 channel 1 input capture source selection (register bit
SOPT4[FTM2CH1SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM2_CHANNEL1 (0)
/* @brief Number of configurable FTM0 fault detection input (number of
register bits SOPT4[FTM0FLTn], where n is a number starting from zero).
*/
#define FSL_FEATURE_SIM_OPT_FTM0_FAULT_COUNT (0)
/* @brief Number of configurable FTM1 fault detection input (number of
register bits SOPT4[FTM1FLTn], where n is a number starting from zero).
*/
#define FSL_FEATURE_SIM_OPT_FTM1_FAULT_COUNT (0)
/* @brief Number of configurable FTM2 fault detection input (number of
register bits SOPT4[FTM2FLTn], where n is a number starting from zero).
*/
#define FSL_FEATURE_SIM_OPT_FTM2_FAULT_COUNT (0)
/* @brief Number of configurable FTM3 fault detection input (number of
register bits SOPT4[FTM3FLTn], where n is a number starting from zero).
*/
#define FSL_FEATURE_SIM_OPT_FTM3_FAULT_COUNT (0)
/* @brief Has FTM hardware trigger 0 software synchronization (register
bit SOPT8[FTMnSYNCBIT], where n is a module instance index). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM_TRIGGER_SYNC (0)
/* @brief Has FTM channels output source selection (register bit
SOPT8[FTMxOCHnSRC], where x is a module instance index and n is a channel
index). */
#define FSL_FEATURE_SIM_OPT_HAS_FTM_CHANNELS_OUTPUT_SRC (0)
/* @brief Has TPM module(s) configuration. */
#define FSL_FEATURE_SIM_OPT_HAS TPM (1)
/* @brief The highest TPM module index. */
#define FSL_FEATURE_SIM_OPT_MAX TPM INDEX (2)

```

```

/* @brief Has TPM module with index 0. */
#define FSL_FEATURE_SIM_OPT_HAS TPM0 (1)
/* @brief Has TPM0 clock selection (register bit field
SOPT4[TPM0CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS TPM0_CLK_SEL (1)
/* @brief Is TPM channels configuration in the SOPT4 (not SOPT9) register
(register bits TPMnCH0SRC, TPMnCLKSEL, where n is a module instance
index). */
#define FSL_FEATURE_SIM_OPT_HAS TPM_CHANNELS_CONFIG_IN_SOPT4_REG (1)
/* @brief Has TPM1 channel 0 input capture source selection (register bit
field SOPT4[TPM1CH0SRC] or SOPT9[TPM1CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS TPM1_CH0_SRC_SELECTION (1)
/* @brief Has TPM1 clock selection (register bit field
SOPT4[TPM1CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS TPM1_CLK_SEL (1)
/* @brief TPM1 channel 0 input capture source selection width (width of
register bit field SOPT4[TPM1CH0SRC] or SOPT9[TPM1CH0SRC]). */
#define FSL_FEATURE_SIM_OPT TPM1_CH0_SRC_SELECTION_WIDTH (1)
/* @brief Has TPM2 channel 0 input capture source selection (register bit
field SOPT4[TPM2CH0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS TPM2_CH0_SRC_SELECTION (1)
/* @brief Has TPM2 clock selection (register bit field
SOPT4[TPM2CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS TPM2_CLK_SEL (1)
/* @brief Has PLL/FLL clock selection (register bit field
SOPT2[PLLFLLSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_PLL_FLL_SELECTION (1)
/* @brief PLL/FLL clock selection width (width of register bit field
SOPT2[PLLFLLSEL]). */
#define FSL_FEATURE_SIM_OPT_PLL_FLL_SELECTION_WIDTH (1)
/* @brief Has NFC clock source selection (register bit SOPT2[NFCSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_NFCSRC (0)
/* @brief Has eSDHC clock source selection (register bit
SOPT2[ESDHCSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_ESDHCSRC (0)
/* @brief Has SDHC clock source selection (register bit SOPT2[SDHCSRC]). */
*/
#define FSL_FEATURE_SIM_OPT_HAS_SDHCSRC (0)
/* @brief Has LCDC clock source selection (register bits SOPT2[LCDCSRC],
SOPT2[LCDC_CLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_LCDCSRC (0)
/* @brief Has ENET timestamp clock source selection (register bit
SOPT2[TIMESRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_TIMESRC (0)
/* @brief Has ENET RMII clock source selection (register bit
SOPT2[RMIISRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_RMIISRC (0)
/* @brief Has USB clock source selection (register bit SOPT2[USBSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_USBSRC (1)
/* @brief Has USB FS clock source selection (register bit
SOPT2[USBFSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_USBFSRC (0)
/* @brief Has USB HS clock source selection (register bit
SOPT2[USBHSRC]). */

```

```

#define FSL_FEATURE_SIM_OPT_HAS_USBHSRC (0)
/* @brief Has LPUART clock source selection (register bit
SOPT2[LPUARTSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUARTSRC (0)
/* @brief Has LPUART0 clock source selection (register bit
SOPT2[LPUART0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART0SRC (0)
/* @brief Has LPUART1 clock source selection (register bit
SOPT2[LPUART1SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_LPUART1SRC (0)
/* @brief Has FLEXIOSRC clock source selection (register bit
SOPT2[FLEXIOSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_FLEXIOSRC (0)
/* @brief Has UART0 clock source selection (register bit
SOPT2[UART0SRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_UART0SRC (1)
/* @brief Has TPM clock source selection (register bit SOPT2[TPMSRC]). */
#define FSL_FEATURE_SIM_OPT_HAS_TPMSRC (1)
/* @brief Has debug trace clock selection (register bit
SOPT2[TRACECLKSEL]). */
#define FSL_FEATURE_SIM_OPT_HAS_TRACE_CLKSEL (0)
/* @brief Number of ADC modules (register bits SOPT7[ADCnTRGSEL],
SOPT7[ADCnPRETRGSEL], SOPT7[ADCnALTTRGSEL], where n is a module instance
index). */
#define FSL_FEATURE_SIM_OPT_ADC_COUNT (1)
/* @brief Has clock 2 output divider (register bit field
CLKDIV1[OUTDIV2]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV2 (0)
/* @brief Has clock 3 output divider (register bit field
CLKDIV1[OUTDIV3]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV3 (0)
/* @brief Has clock 4 output divider (register bit field
CLKDIV1[OUTDIV4]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV4 (1)
/* @brief Clock 4 output divider width (width of register bit field
CLKDIV1[OUTDIV4]). */
#define FSL_FEATURE_SIM_DIVIDER_OUTDIV4_WIDTH (3)
/* @brief Has clock 5 output divider (register bit field
CLKDIV1[OUTDIV5]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_OUTDIV5 (0)
/* @brief Has USB clock divider (register bit field CLKDIV2[USBDIV] and
CLKDIV2[USBFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBDIV (0)
/* @brief Has USB FS clock divider (register bit field CLKDIV2[USBFSDIV]
and CLKDIV2[USBFSFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBFSDIV (0)
/* @brief Has USB HS clock divider (register bit field CLKDIV2[USBHSDIV]
and CLKDIV2[USBHSFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_USBHSDIV (0)
/* @brief Has PLL/FLL clock divider (register bit field
CLKDIV3[PLLFLLDIV] and CLKDIV3[PLLFLLFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_PLLFLLDIV (0)
/* @brief Has LCDC clock divider (register bit field CLKDIV3[LCDCDIV] and
CLKDIV3[LCDCFRAC]). */

```

```

#define FSL_FEATURE_SIM_DIVIDER_HAS_LCDCDIV (0)
/* @brief Has trace clock divider (register bit field CLKDIV4[TRACEDIV]
and CLKDIV4[TRACEFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_TRACEDIV (0)
/* @brief Has NFC clock divider (register bit field CLKDIV4[NFCDIV] and
CLKDIV4[NFCFRAC]). */
#define FSL_FEATURE_SIM_DIVIDER_HAS_NFCDIV (0)
/* @brief Has Kinetis family ID (register bit field SDID[FAMILYID]). */
#define FSL_FEATURE_SIM_SDID_HAS_FAMILYID (0)
/* @brief Has Kinetis family ID (register bit field SDID[FAMID]). */
#define FSL_FEATURE_SIM_SDID_HAS_FAMID (1)
/* @brief Has Kinetis sub-family ID (register bit field SDID[SUBFAMID]). */
*/
#define FSL_FEATURE_SIM_SDID_HAS_SUBFAMID (1)
/* @brief Has Kinetis series ID (register bit field SDID[SERIESID]). */
#define FSL_FEATURE_SIM_SDID_HAS_SERIESID (1)
/* @brief Has device die ID (register bit field SDID[DIEID]). */
#define FSL_FEATURE_SIM_SDID_HAS_DIEID (1)
/* @brief Has system SRAM size specifier (register bit field
SDID[SRAMSIZE]). */
#define FSL_FEATURE_SIM_SDID_HAS_SRAMSIZE (1)
/* @brief Has flash mode (register bit FCFG1[FLASHDOZE]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FLASHDOZE (1)
/* @brief Has flash disable (register bit FCFG1[FLASHDIS]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FLASHDIS (1)
/* @brief Has FTFE disable (register bit FCFG1[FTFDIS]). */
#define FSL_FEATURE_SIM_FCFG_HAS_FTFDIS (0)
/* @brief Has FlexNVM size specifier (register bit field FCFG1[NVMSIZE]). */
*/
#define FSL_FEATURE_SIM_FCFG_HAS_NVMSIZE (0)
/* @brief Has EEPROM size specifier (register bit field FCFG1[EESIZE]). */
*/
#define FSL_FEATURE_SIM_FCFG_HAS_EESIZE (0)
/* @brief Has FlexNVM partition (register bit field FCFG1[DEPART]). */
#define FSL_FEATURE_SIM_FCFG_HAS_DEPART (0)
/* @brief Maximum flash address block 0 address specifier (register bit
field FCFG2[MAXADDR0]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR0 (1)
/* @brief Maximum flash address block 1 address specifier (register bit
field FCFG2[MAXADDR1]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR1 (0)
/* @brief Maximum flash address block 0 or 1 address specifier (register
bit field FCFG2[MAXADDR01]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR01 (0)
/* @brief Maximum flash address block 2 or 3 address specifier (register
bit field FCFG2[MAXADDR23]). */
#define FSL_FEATURE_SIM_FCFG_HAS_MAXADDR23 (0)
/* @brief Has program flash availability specifier (register bit
FCFG2[PFLSH]). */
#define FSL_FEATURE_SIM_FCFG_HAS_PFLSH (0)
/* @brief Has program flash swapping (register bit FCFG2[SWAPPFLSH]). */
#define FSL_FEATURE_SIM_FCFG_HAS_PFLSH_SWAP (0)
/* @brief Has miscellaneous control register (register MCR). */
#define FSL_FEATURE_SIM_HAS_MISC_CONTROLS (0)

```

```

/* @brief Has COP watchdog (registers COPC and SRVCOP). */
#define FSL_FEATURE_SIM_HAS_COP_WATCHDOG (1)
/* @brief Has COP watchdog stop (register bits COPC[COPSTPEN], COPC[COPDBGEN] and COPC[COPCLKSEL]). */
#define FSL_FEATURE_SIM_HAS_COP_STOP (0)
/* @brief Has LLWU clock gate bit (e.g SIM_SCGC4). */
#define FSL_FEATURE_SIM_HAS_SCGC_LLWU (0)

/* SMC module features */

/* @brief Has partial stop option (register bit STOPCTRL[PSTOPO]). */
#define FSL_FEATURE_SMC_HAS_PSTOPO (1)
/* @brief Has LPO power option (register bit STOPCTRL[LPOPO]). */
#define FSL_FEATURE_SMC_HAS_LPOPO (0)
/* @brief Has POR power option (register bit STOPCTRL[PORPO] or VLLSCTRL[PORPO]). */
#define FSL_FEATURE_SMC_HAS_PORPO (1)
/* @brief Has low power wakeup on interrupt (register bit PMCTRL[LPWUI]). */
*/
#define FSL_FEATURE_SMC_HAS_LPWUI (0)
/* @brief Has LLS or VLLS mode control (register bit STOPCTRL[LLSM]). */
#define FSL_FEATURE_SMC_HAS_LLS_SUBMODE (0)
/* @brief Has VLLS mode control (register bit VLLSCTRL[VLLSM]). */
#define FSL_FEATURE_SMC_USE_VLLSCTRL_REG (0)
/* @brief Has VLLS mode control (register bit STOPCTRL[VLLSM]). */
#define FSL_FEATURE_SMC_USE_STOPCTRL_VLLSM (1)
/* @brief Has RAM partition 2 power option (register bit STOPCTRL[RAM2PO]). */
#define FSL_FEATURE_SMC_HAS_RAM2_POWER_OPTION (0)
/* @brief Has high speed run mode (register bit PMPROT[AHSRUN]). */
#define FSL_FEATURE_SMC_HAS_HIGH_SPEED_RUN_MODE (0)
/* @brief Has low leakage stop mode (register bit PMPROT[ALLS]). */
#define FSL_FEATURE_SMC_HAS_LOW_LEAKAGE_STOP_MODE (1)
/* @brief Has stop submode 0(VLLS0). */
#define FSL_FEATURE_SMC_HAS_STOP_SUBMODE0 (1)
/* @brief Has stop submode 2(VLLS2). */
#define FSL_FEATURE_SMC_HAS_STOP_SUBMODE2 (0)

/* SPI module features */

/* @brief Has DMA support (register bit fields C2[RXDMAE] and C2[TXDMAE]). */
#define FSL_FEATURE_SPI_HAS_DMA_SUPPORT (1)
/* @brief Receive/transmit FIFO size in number of 16-bit communication items. */
#define FSL_FEATURE_SPI_FIFO_SIZE (0)
#define FSL_FEATURE_SPI_FIFO_SIZEEx { 0, 0 }
/* @brief Maximum transfer data width in bits. */
#define FSL_FEATURE_SPI_MAX_DATA_WIDTH (8)
/* @brief The data register name has postfix (L as low and H as high). */
#define FSL_FEATURE_SPI_DATA_REGISTER_HAS_POSTFIX (0)
/* @brief Has separated TXDATA and CMD FIFOs (register SREX). */
#define FSL_FEATURE_SPI_HAS_SEPARATE_TXDATA_CMD_FIFO (0)
/* @brief Has 16-bit data transfer support. */

```

```

#define FSL_FEATURE_SPI_16BIT_TRANSFERS (0)

/* SysTick module features */

/* @brief Systick has external reference clock. */
#define FSL_FEATURE_SYSTICK_HAS_EXT_REF (1)
/* @brief Systick external reference clock is core clock divided by this
value. */
#define FSL_FEATURE_SYSTICK_EXT_REF_CORE_DIV (16)

/* TPM module features */

/* @brief Bus clock is the source clock for the module. */
#define FSL_FEATURE_TPM_BUS_CLOCK (0)
/* @brief Number of channels. */
#define FSL_FEATURE_TPM_CHANNEL_COUNT (6)
#define FSL_FEATURE_TPM_CHANNEL_COUNTx { 6, 2, 2 }
/* @brief Has counter reset by the selected input capture event (register
bits C0SC[ICRST], C1SC[ICRST], ...). */
#define FSL_FEATURE_TPM_HAS_COUNTER_RESET_BY_CAPTURE_EVENT (0)

/* TSI module features */

/* @brief TSI module version. */
#define FSL_FEATURE_TSI_VERSION (4)
/* @brief Has end-of-scan DMA transfer request enable (register bit
GENCS[EOSDMEO]). */
#define FSL_FEATURE_TSI_HAS_END_OF_SCAN_DMA_ENABLE (0)
/* @brief Number of TSI channels. */
#define FSL_FEATURE_TSI_CHANNEL_COUNT (16)

/* LPSCI module features */

/* @brief Has receive FIFO overflow detection (bit field CFIFO[RXOFE]). */
*/
#define FSL_FEATURE_LPSCI_HAS_IRQ_EXTENDED_FUNCTIONS (1)
/* @brief Has low power features (can be enabled in wait mode via
register bit C1[DOZEEN] or CTRL[DOZEEN] if the registers are 32-bit
wide). */
#define FSL_FEATURE_LPSCI_HAS_LOW_POWER_UART_SUPPORT (1)
/* @brief Has extended data register ED (or extra flags in the DATA
register if the registers are 32-bit wide). */
#define FSL_FEATURE_LPSCI_HAS_EXTENDED_DATA_REGISTER_FLAGS (0)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or
zero if no FIFO is available). */
#define FSL_FEATURE_LPSCI_HAS_FIFO (0)
/* @brief Hardware flow control (RTS, CTS) is supported. */
#define FSL_FEATURE_LPSCI_HAS_MODEM_SUPPORT (0)
/* @brief Infrared (modulation) is supported. */
#define FSL_FEATURE_LPSCI_HAS_IR_SUPPORT (0)
/* @brief 2 bits long stop bit is available. */
#define FSL_FEATURE_LPSCI_HAS_STOP_BIT_CONFIG_SUPPORT (1)
/* @brief Maximal data width without parity bit. */
#define FSL_FEATURE_LPSCI_HAS_10BIT_DATA_SUPPORT (1)

```

```

/* @brief Baud rate fine adjustment is available. */
#define FSL FEATURE LPSCI HAS BAUD RATE FINE ADJUST SUPPORT (0)
/* @brief Baud rate oversampling is available (has bit fields C4[OSR], C5[BOTHEDGE], C5[RESYNCDIS] or BAUD[OSR], BAUD[BOTHEDGE], BAUD[RESYNCDIS] if the registers are 32-bit wide). */
#define FSL FEATURE LPSCI HAS BAUD RATE OVER SAMPLING SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL FEATURE LPSCI HAS RX RESYNC SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL FEATURE LPSCI HAS BOTH EDGE SAMPLING SUPPORT (1)
/* @brief Peripheral type. */
#define FSL FEATURE LPSCI IS SCI (1)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or zero if no FIFO is available). */
#define FSL FEATURE LPSCI FIFO SIZE (0)
/* @brief Maximal data width without parity bit. */
#define FSL FEATURE LPSCI MAX DATA WIDTH WITH NO PARITY (10)
/* @brief Maximal data width with parity bit. */
#define FSL FEATURE LPSCI MAX DATA WIDTH WITH PARITY (9)
/* @brief Supports two match addresses to filter incoming frames. */
#define FSL FEATURE LPSCI HAS ADDRESS MATCHING (1)
/* @brief Has transmitter/receiver DMA enable bits C5[TDMAE]/C5[RDMAE] (or BAUD[TDMAE]/BAUD[RDMAE] if the registers are 32-bit wide). */
#define FSL FEATURE LPSCI HAS DMA ENABLE (1)
/* @brief Has transmitter/receiver DMA select bits C4[TDMAS]/C4[RDMAS], resp. C5[TDMAS]/C5[RDMAS] if IS SCI = 0. */
#define FSL FEATURE LPSCI HAS DMA SELECT (0)
/* @brief Data character bit order selection is supported (bit field S2[MSBF] or STAT[MSBF] if the registers are 32-bit wide). */
#define FSL FEATURE LPSCI HAS BIT ORDER SELECT (1)
/* @brief Has smart card (ISO7816 protocol) support and no improved smart card support. */
#define FSL FEATURE LPSCI HAS SMART CARD SUPPORT (0)
/* @brief Has improved smart card (ISO7816 protocol) support. */
#define FSL FEATURE LPSCI HAS IMPROVED SMART CARD SUPPORT (0)
/* @brief Has local operation network (CEA709.1-B protocol) support. */
#define FSL FEATURE LPSCI HAS LOCAL OPERATION NETWORK SUPPORT (0)
/* @brief Has 32-bit registers (BAUD, STAT, CTRL, DATA, MATCH, MODIR) instead of 8-bit (BDH, BDL, C1, S1, D, etc.). */
#define FSL FEATURE LPSCI HAS 32BIT REGISTERS (0)
/* @brief Lin break detect available (has bit BDH[LBKDI]). */
#define FSL FEATURE LPSCI HAS LIN BREAK DETECT (1)
/* @brief UART stops in Wait mode available (has bit C1[UARTSWAI]). */
#define FSL FEATURE LPSCI HAS WAIT MODE OPERATION (0)
/* @brief Has separate DMA RX and TX requests. */
#define FSL FEATURE LPSCI HAS SEPARATE DMA RX TX REQn(x) \
    ((x) == 0 ? (1) : (-1))

/* UART module features */

/* @brief Has receive FIFO overflow detection (bit field CFIFO[RXOFE]). */
#define FSL FEATURE UART HAS IRQ EXTENDED FUNCTIONS (1)

```

```

/* @brief Has low power features (can be enabled in wait mode via
register bit C1[DOZEEN] or CTRL[DOZEEN] if the registers are 32-bit
wide). */
#define FSL_FEATURE_UART_HAS_LOW_POWER_UART_SUPPORT (0)
/* @brief Has extended data register ED (or extra flags in the DATA
register if the registers are 32-bit wide). */
#define FSL_FEATURE_UART_HAS_EXTENDED_DATA_REGISTER_FLAGS (0)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or
zero if no FIFO is available). */
#define FSL_FEATURE_UART_HAS_FIFO (0)
/* @brief Hardware flow control (RTS, CTS) is supported. */
#define FSL_FEATURE_UART_HAS_MODEM_SUPPORT (0)
/* @brief Infrared (modulation) is supported. */
#define FSL_FEATURE_UART_HAS_IR_SUPPORT (0)
/* @brief 2 bits long stop bit is available. */
#define FSL_FEATURE_UART_HAS_STOP_BIT_CONFIG_SUPPORT (1)
/* @brief Maximal data width without parity bit. */
#define FSL_FEATURE_UART_HAS_10BIT_DATA_SUPPORT (0)
/* @brief Baud rate fine adjustment is available. */
#define FSL_FEATURE_UART_HAS_BAUD_RATE_FINE_ADJUST_SUPPORT (0)
/* @brief Baud rate oversampling is available (has bit fields C4[OSR],
C5[BOTHEDGE], C5[RESYNCDIS] or BAUD[OSR], BAUD[BOTHEDGE], BAUD[RESYNCDIS]
if the registers are 32-bit wide). */
#define FSL_FEATURE_UART_HAS_BAUD_RATE_OVER_SAMPLING_SUPPORT (0)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_UART_HAS_RX_RESYNC_SUPPORT (1)
/* @brief Baud rate oversampling is available. */
#define FSL_FEATURE_UART_HAS_BOTH_EDGE_SAMPLING_SUPPORT (1)
/* @brief Peripheral type. */
#define FSL_FEATURE_UART_IS_SCI (1)
/* @brief Capacity (number of entries) of the transmit/receive FIFO (or
zero if no FIFO is available). */
#define FSL_FEATURE_UART_FIFO_SIZE (0)
/* @brief Maximal data width without parity bit. */
#define FSL_FEATURE_UART_MAX_DATA_WIDTH_WITH_NO_PARITY (9)
/* @brief Maximal data width with parity bit. */
#define FSL_FEATURE_UART_MAX_DATA_WIDTH_WITH_PARITY (8)
/* @brief Supports two match addresses to filter incoming frames. */
#define FSL_FEATURE_UART_HAS_ADDRESS_MATCHING (0)
/* @brief Has transmitter/receiver DMA enable bits C5[TDMAE]/C5[RDMAE]
(or BAUD[TDMAE]/BAUD[RDMAE] if the registers are 32-bit wide). */
#define FSL_FEATURE_UART_HAS_DMA_ENABLE (0)
/* @brief Has transmitter/receiver DMA select bits C4[TDMAS]/C4[RDMAS],
resp. C5[TDMAS]/C5[RDMAS] if IS_SCI = 0. */
#define FSL_FEATURE_UART_HAS_DMA_SELECT (1)
/* @brief Data character bit order selection is supported (bit field
S2[MSBF] or STAT[MSBF] if the registers are 32-bit wide). */
#define FSL_FEATURE_UART_HAS_BIT_ORDER_SELECT (0)
/* @brief Has smart card (ISO7816 protocol) support and no improved smart
card support. */
#define FSL_FEATURE_UART_HAS_SMART_CARD_SUPPORT (0)
/* @brief Has improved smart card (ISO7816 protocol) support. */
#define FSL_FEATURE_UART_HAS_IMPROVED_SMART_CARD_SUPPORT (0)
/* @brief Has local operation network (CEA709.1-B protocol) support. */

```

```

#define FSL_FEATURE_UART_HAS_LOCAL_OPERATION_NETWORK_SUPPORT (0)
/* @brief Has 32-bit registers (BAUD, STAT, CTRL, DATA, MATCH, MODIR)
instead of 8-bit (BDH, BDL, C1, S1, D, etc.). */
#define FSL_FEATURE_UART_HAS_32BIT_REGISTERS (0)
/* @brief Lin break detect available (has bit BDH[LBKDI]). */
#define FSL_FEATURE_UART_HAS_LIN_BREAK_DETECT (1)
/* @brief UART stops in Wait mode available (has bit C1[UARTSWAI]). */
#define FSL_FEATURE_UART_HAS_WAIT_MODE_OPERATION (1)
/* @brief Has separate DMA RX and TX requests. */
#define FSL_FEATURE_UART_HAS_SEPARATE_DMA_RX_TX_REQn(x) \
    ((x) == 1 ? (1) : \
     ((x) == 2 ? (1) : (-1)))

/* USB module features */

/* @brief HOST mode enabled */
#define FSL_FEATURE_USB_KHCI_HOST_ENABLED (1)
/* @brief OTG mode enabled */
#define FSL_FEATURE_USB_KHCI_OTG_ENABLED (1)
/* @brief Size of the USB dedicated RAM */
#define FSL_FEATURE_USB_KHCI_USB_RAM (0)
/* @brief Has KEEP_ALIVE_CTRL register */
#define FSL_FEATURE_USB_KHCI_KEEP_ALIVE_ENABLED (0)
/* @brief Has the Dynamic SOF threshold compare support */
#define FSL_FEATURE_USB_KHCI_DYNAMIC_SOF_THRESHOLD_COMPARE_ENABLED (0)
/* @brief Has the VBUS detect support */
#define FSL_FEATURE_USB_KHCI_VBUS_DETECT_ENABLED (0)
/* @brief Has the IRC48M module clock support */
#define FSL_FEATURE_USB_KHCI_IRC48M_MODULE_CLOCK_ENABLED (0)

#endif /* __FSL_MKL25Z4_FEATURES_H__ */

*****
*** EOF

*****
** 
 * @file arch_arm32.h
 *
 * @brief source file for arch_arm.c
 *
 * header files for arch_arm32.h for the following function
 * - ARM32_AIRCR_get_endianness_setting
 *
 * @author Miles Frain
 * @author Shreya Chakraborty
 * @version 1
 * @date 2018-02-10
 */
#include <stdint.h>

```

```

#define __SCB_ADDRESS 0xE000ED00
#define __AIRCR_ADDRESS_OFFSET 0xC
#define __AIRCR *(uint32_t *)
*) (__SCB_ADDRESS+__AIRCR_ADDRESS_OFFSET)
#define __AIRCR_ENDIANNESS_OFFSET (15)
#define __AIRCR_ENDIANNESS_MASK (1<<__AIRCR_ENDIANNESS_OFFSET)
#define __CPUID_ADDRESS_OFFSET (0x00)
#define __CPUID (__SCB_ADDRESS + __CPUID_ADDRESS_OFFSET )
#define __CPUID_PART_NO_OFFSET (4)
#define __CPUID_PART_NO_MASK (0xFFFF)
#define CPUID ((uint32_t*)__CPUID)
#define __CCR_ADDRESS_OFFSET (0x14)
#define __CCR (__SCB_ADDRESS + __CCR_ADDRESS_OFFSET )
#define CCR ((uint32_t*)__CCR)
#define __CCR_STK_ALIGNMENT_OFFSET (9)
#define __CCR_STK_ALIGNMENT_MASK (0x200)
#define __CCR_UNALIGNED_ACCESS_TRAP_OFFSET (3)
#define __CCR_UNALIGNED_ACCESS_TRAP_MASK (0x8)
#define __CCR_DIVIDE_BY_ZERO_TRAP_OFFSET (4)
#define __CCR_DIVIDE_BY_ZERO_TRAP_MASK (0x10)

__attribute__((always_inline)) uint32_t ARM32_CCR_get_stack_alignment()
{
    return
(uint32_t) (((*CCR) & __CCR_STK_ALIGNMENT_MASK )>>(__CCR_STK_ALIGNMENT_OFFSET));
}

__attribute__((always_inline)) uint32_t ARM32_CPUID_get_part_number()
{
    return
(uint32_t) (((*CPUID) & __CPUID_PART_NO_MASK )>>(__CPUID_PART_NO_OFFSET));
}

__attribute__((always_inline)) uint32_t
ARM32_CCR_enable_divide_by_zero_trap()
{
    CCR |= (1)<<__CCR_DIVIDE_BY_ZERO_TRAP_OFFSET;
}

__attribute__((always_inline)) uint32_t
ARM32_CCR_enable_unaligned_access_trap()
{
    CCR |= (1)<<__CCR_STK_ALIGNMENT_OFFSET;
}

static inline void ARM32_create_unaligned_access_trap()
{
    /* Perform an unaligned read/access */
    uint32_t data[2];
    uint32_t element = *((uint32_t*)((uint8_t*)data) + 1))
}

```

```

static inline void ARM32_create_divide_by_zero_trap()
{
    uint8_t div = 1;
    div = div/0;
}
__attribute__((always_inline)) uint32_t
ARM32_AIRCR_get_endianness_setting()
{
    return ((__AIRCR &
    __AIRCR_ENDIANNESS_MASK)>>__AIRCR_ENDIANNESS_OFFSET);
}

// __attribute__((always_inline)) uint32_t
ARM32_AIRCR_get_endianness_setting();
//****************************************************************************
**//*
 * @file      core_cmInstr.h
 * @brief     CMSIS Cortex-M Core Instruction Access Header File
 * @version   V4.10
 * @date      18. March 2015
 *
 * @note
 *

*****
/* Copyright (c) 2009 - 2014 ARM LIMITED

   All rights reserved.
   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
met:
   - Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
   - Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
   - Neither the name of ARM nor the names of its contributors may be
used
   to endorse or promote products derived from this software without
   specific prior written permission.
*
   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS
BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS

```

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER  
IN  
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR  
OTHERWISE)  
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF  
THE  
POSSIBILITY OF SUCH DAMAGE.

-----  
----\*/

```
#ifndef __CORE_CMINSTR_H
#define __CORE_CMINSTR_H

/* ##### Core Instruction Access
/** \defgroup CMSIS_Core_InstructionInterface CMSIS Core Instruction
Interface
Access to dedicated instructions
@{
*/
#if defined ( __CC_ARM ) /*-----RealView Compiler -----
*/
/* ARM armcc specific functions */

#if (__ARMCC_VERSION < 400677)
#error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif

/** \brief No Operation

No Operation does nothing. This instruction can be used for code
alignment purposes.
*/
#define __NOP __nop

/** \brief Wait For Interrupt

Wait For Interrupt is a hint instruction that suspends execution
until one of a number of events occurs.
*/
#define __WFI __wfi

/** \brief Wait For Event

Wait For Event is a hint instruction that permits the processor to
enter
a low-power state until one of a number of events occurs.
*/
```

```

#define __WFE           __wfe
/** \brief Send Event

    Send Event is a hint instruction. It causes an event to be signaled
to the CPU.
 */
#define __SEV           __sev

/** \brief Instruction Synchronization Barrier

    Instruction Synchronization Barrier flushes the pipeline in the
processor,
so that all instructions following the ISB are fetched from cache or
memory, after the instruction has been completed.
*/
#define __ISB() do { \
    __schedule_barrier(); \
    __isb(0xF); \
    __schedule_barrier(); \
} while (0)

/** \brief Data Synchronization Barrier

    This function acts as a special kind of Data Memory Barrier.
    It completes when all explicit memory accesses before this
instruction complete.
*/
#define __DSB() do { \
    __schedule_barrier(); \
    __dsb(0xF); \
    __schedule_barrier(); \
} while (0)

/** \brief Data Memory Barrier

    This function ensures the apparent order of the explicit memory
operations before
and after the instruction, without ensuring their completion.
*/
#define __DMB() do { \
    __schedule_barrier(); \
    __dmb(0xF); \
    __schedule_barrier(); \
} while (0)

/** \brief Reverse byte order (32 bit)

    This function reverses the byte order in integer value.

\param [in]      value  Value to reverse
\return          Reversed value

```

```

*/
#define __REV           __rev

/** \brief Reverse byte order (16 bit)

    This function reverses the byte order in two unsigned short values.

    \param [in]      value  Value to reverse
    \return          Reversed value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".rev16_text"))) __STATIC_INLINE __ASM uint32_t
__REV16(uint32_t value)
{
    rev16 r0, r0
    bx lr
}
#endif

/** \brief Reverse byte order in signed short value

    This function reverses the byte order in a signed short value with
    sign extension to integer.

    \param [in]      value  Value to reverse
    \return          Reversed value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".revsh_text"))) __STATIC_INLINE __ASM int32_t
__REVSH(int32_t value)
{
    revsh r0, r0
    bx lr
}
#endif

/** \brief Rotate Right in unsigned value (32 bit)

    This function Rotate Right (immediate) provides the value of the
    contents of a register rotated by a variable number of bits.

    \param [in]      value  Value to rotate
    \param [in]      value  Number of Bits to rotate
    \return          Rotated value
*/
#define __ROR           __ror

/** \brief Breakpoint

    This function causes the processor to enter Debug state.

```

Debug tools can use this to investigate system state when the instruction at a particular address is reached.

```
\param [in]      value  is ignored by the processor.  
                  If required, a debugger can use it to store additional  
information about the breakpoint.  
*/  
#define __BKPT(value)           __breakpoint(value)  
  
/** \brief Reverse bit order of value  
  
This function reverses the bit order of the given value.  
  
\param [in]      value  Value to reverse  
\return          Reversed value  
*/  
#if      (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)  
#define __RBIT             __rbit  
#else  
__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t  
value)  
{  
    uint32_t result;  
    int32_t s = 4 /*sizeof(v) */ * 8 - 1; // extra shift needed at end  
  
    result = value;                      // r will be reversed bits of v;  
    first get LSB of v  
    for (value >= 1; value; value >= 1)  
    {  
        result <= 1;  
        result |= value & 1;  
        s--;  
    }  
    result <= s;                         // shift when v's highest bits are  
zero  
    return(result);  
}  
#endif  
  
/** \brief Count leading zeros  
  
This function counts the number of leading zeros of a data value.  
  
\param [in]      value  Value to count the leading zeros  
\return          number of leading zeros in value  
*/  
#define __CLZ               __clz  
  
#if      (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)  
/** \brief LDR Exclusive (8 bit)
```

This function executes a exclusive LDR instruction for 8 bit value.

```
\param [in]      ptr  Pointer to data
\return          value of type uint8_t at (*ptr)
*/
#define __LDREXB(ptr)          ((uint8_t) __ldrex(ptr))
```

/\*\* \brief LDR Exclusive (16 bit)

This function executes a exclusive LDR instruction for 16 bit values.

```
\param [in]      ptr  Pointer to data
\return          value of type uint16_t at (*ptr)
*/
#define __LDREXBH(ptr)         ((uint16_t) __ldrex(ptr))
```

/\*\* \brief LDR Exclusive (32 bit)

This function executes a exclusive LDR instruction for 32 bit values.

```
\param [in]      ptr  Pointer to data
\return          value of type uint32_t at (*ptr)
*/
#define __LDREXBW(ptr)         ((uint32_t) __ldrex(ptr))
```

/\*\* \brief STR Exclusive (8 bit)

This function executes a exclusive STR instruction for 8 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return     0  Function succeeded
\return     1  Function failed
*/
#define __STREXB(value, ptr)    __strex(value, ptr)
```

/\*\* \brief STR Exclusive (16 bit)

This function executes a exclusive STR instruction for 16 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return     0  Function succeeded
\return     1  Function failed
*/
#define __STREXBH(value, ptr)   __strex(value, ptr)
```

/\*\* \brief STR Exclusive (32 bit)

This function executes a exclusive STR instruction for 32 bit values.

```
\param [in]    value  Value to store
\param [in]    ptr    Pointer to location
\return        0  Function succeeded
\return        1  Function failed
*/
#define __STREXW(value, ptr)           __strex(value, ptr)
```

/\*\* \brief Remove the exclusive lock

This function removes the exclusive lock which is created by LDREX.

```
/*
#define __CLREX           __clrex
```

/\*\* \brief Signed Saturate

This function saturates a signed value.

```
\param [in]    value  Value to be saturated
\param [in]    sat    Bit position to saturate to (1..32)
\return        Saturated value
*/
#define __SSAT           __ssat
```

/\*\* \brief Unsigned Saturate

This function saturates an unsigned value.

```
\param [in]    value  Value to be saturated
\param [in]    sat    Bit position to saturate to (0..31)
\return        Saturated value
*/
#define __USAT           __usat
```

/\*\* \brief Rotate Right with Extend (32 bit)

This function moves each bit of a bitstring right by one bit.  
The carry input is shifted in at the left end of the bitstring.

```
\param [in]    value  Value to rotate
\return        Rotated value
*/
#ifndef __NO_EMBEDDED_ASM
__attribute__((section(".rrx_text"))) __STATIC_INLINE __ASM uint32_t
__RRX(uint32_t value)
{
    rrx r0, r0
```

```

        bx lr
    }
#endif

/** \brief LDRT Unprivileged (8 bit)

    This function executes a Unprivileged LDRT instruction for 8 bit
values.

    \param [in]      ptr  Pointer to data
    \return         value of type uint8_t at (*ptr)
*/
#define __LDRBT(ptr)          ((uint8_t) __ldrt(ptr))

/** \brief LDRT Unprivileged (16 bit)

    This function executes a Unprivileged LDRT instruction for 16 bit
values.

    \param [in]      ptr  Pointer to data
    \return         value of type uint16_t at (*ptr)
*/
#define __LDRHT(ptr)          ((uint16_t) __ldrt(ptr))

/** \brief LDRT Unprivileged (32 bit)

    This function executes a Unprivileged LDRT instruction for 32 bit
values.

    \param [in]      ptr  Pointer to data
    \return         value of type uint32_t at (*ptr)
*/
#define __LDRT(ptr)           ((uint32_t) __ldrt(ptr))

/** \brief STRT Unprivileged (8 bit)

    This function executes a Unprivileged STRT instruction for 8 bit
values.

    \param [in]      value  Value to store
    \param [in]      ptr    Pointer to location
*/
#define __STRB7(value, ptr)     __strt(value, ptr)

/** \brief STRT Unprivileged (16 bit)

    This function executes a Unprivileged STRT instruction for 16 bit
values.

```

```

    \param [in]  value  Value to store
    \param [in]  ptr    Pointer to location
 */
#define __STRHT(value, ptr)           __strt(value, ptr)

/** \brief STRT Unprivileged (32 bit)

    This function executes a Unprivileged STRT instruction for 32 bit
values.

    \param [in]  value  Value to store
    \param [in]  ptr    Pointer to location
 */
#define __STRT(value, ptr)           __strt(value, ptr)

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#ifndef __GNUC__
/*----- GNU Compiler -----
-----*/
/* GNU gcc specific functions */

/* Define macros for porting to both thumb1 and thumb2.
 * For thumb1, use low register (r0-r7), specified by constraint "l"
 * Otherwise, use general registers, specified by constraint "r" */
#if defined(__thumb__) && !defined(__thumb2__)
#define __CMSIS_GCC_OUT_REG(r) "=l" (r)
#define __CMSIS_GCC_USE_REG(r) "l" (r)
#else
#define __CMSIS_GCC_OUT_REG(r) "=r" (r)
#define __CMSIS_GCC_USE_REG(r) "r" (r)
#endif

/** \brief No Operation

    No Operation does nothing. This instruction can be used for code
alignment purposes.
*/
__attribute__((always_inline)) __STATIC_INLINE void __NOP(void)
{
    __ASM volatile ("nop");
}

/** \brief Wait For Interrupt

    Wait For Interrupt is a hint instruction that suspends execution
until one of a number of events occurs.
*/
__attribute__((always_inline)) __STATIC_INLINE void __WFI(void)
{
    __ASM volatile ("wfi");
}

```

```

/** \brief Wait For Event

    Wait For Event is a hint instruction that permits the processor to
enter
    a low-power state until one of a number of events occurs.
 */
attribute__((always_inline)) __STATIC_INLINE void __WFE(void)
{
    __ASM volatile ("wfe");
}

/** \brief Send Event

    Send Event is a hint instruction. It causes an event to be signaled
to the CPU.
 */
attribute__((always_inline)) __STATIC_INLINE void __SEV(void)
{
    __ASM volatile ("sev");
}

/** \brief Instruction Synchronization Barrier

    Instruction Synchronization Barrier flushes the pipeline in the
processor,
    so that all instructions following the ISB are fetched from cache or
memory, after the instruction has been completed.
 */
attribute__((always_inline)) __STATIC_INLINE void __ISB(void)
{
    __ASM volatile ("isb 0xF:::memory");
}

/** \brief Data Synchronization Barrier

    This function acts as a special kind of Data Memory Barrier.
    It completes when all explicit memory accesses before this
instruction complete.
 */
attribute__((always_inline)) __STATIC_INLINE void __DSB(void)
{
    __ASM volatile ("dsb 0xF:::memory");
}

/** \brief Data Memory Barrier

    This function ensures the apparent order of the explicit memory
operations before

```

```

        and after the instruction, without ensuring their completion.
*/
__attribute__((always_inline)) __STATIC_INLINE void __DMB(void)
{
    __ASM volatile ("dmb 0xF:::memory");
}

/** \brief Reverse byte order (32 bit)

This function reverses the byte order in integer value.

\param [in]      value  Value to reverse
\return          Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV(uint32_t
value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 5)
    return __builtin_bswap32(value);
#else
    uint32_t result;

    __ASM volatile ("rev %0, %1" : __CMSIS_GCC_OUT_REG (result) :
    __CMSIS_GCC_USE_REG (value) );
    return(result);
#endif
}

/** \brief Reverse byte order (16 bit)

This function reverses the byte order in two unsigned short values.

\param [in]      value  Value to reverse
\return          Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __REV16(uint32_t
value)
{
    uint32_t result;

    __ASM volatile ("rev16 %0, %1" : __CMSIS_GCC_OUT_REG (result) :
    __CMSIS_GCC_USE_REG (value) );
    return(result);
}

/** \brief Reverse byte order in signed short value

This function reverses the byte order in a signed short value with
sign extension to integer.

\param [in]      value  Value to reverse

```

```

        \return          Reversed value
*/
__attribute__((always_inline)) __STATIC_INLINE int32_t __REVSH(int32_t
value)
{
#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    return (short)__builtin_bswap16(value);
#else
    uint32_t result;

    __ASM volatile ("revsh %0, %1" : __CMSIS_GCC_OUT_REG (result) :
__CMSIS_GCC_USE_REG (value) );
    return(result);
#endif
}

```

/\*\* \brief Rotate Right in unsigned value (32 bit)

This function Rotate Right (immediate) provides the value of the contents of a register rotated by a variable number of bits.

```

\param [in]      value  Value to rotate
\param [in]      value  Number of Bits to rotate
\return          Rotated value
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __ROR(uint32_t
op1, uint32_t op2)
{
    return (op1 >> op2) | (op1 << (32 - op2));
}

```

/\*\* \brief Breakpoint

This function causes the processor to enter Debug state.  
Debug tools can use this to investigate system state when the instruction at a particular address is reached.

```

\param [in]      value  is ignored by the processor.
                  If required, a debugger can use it to store additional
information about the breakpoint.
*/
#define __BKPT(value)           __ASM volatile ("bkpt
"#value)

```

/\*\* \brief Reverse bit order of value

This function reverses the bit order of the given value.

```

\param [in]      value  Value to reverse
\return          Reversed value
*/

```

```

__attribute__((always_inline)) __STATIC_INLINE uint32_t __RBIT(uint32_t
value)
{
    uint32_t result;

#if      (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)
    __ASM volatile ("rbit %0, %1" : "=r" (result) : "r" (value) );
#else
    int32_t s = 4 /*sizeof(v)*/ * 8 - 1; // extra shift needed at end

    result = value;                                // r will be reversed bits of v;
first get LSB of v
    for (value >>= 1; value; value >>= 1)
    {
        result <<= 1;
        result |= value & 1;
        s--;
    }
    result <<= s;                                // shift when v's highest bits are
zero
#endif
    return(result);
}

```

/\*\* \brief Count leading zeros

This function counts the number of leading zeros of a data value.

```

\param [in]  value  Value to count the leading zeros
\return      number of leading zeros in value
*/
#define __CLZ          __builtin_clz

#if      (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)

/** \brief LDR Exclusive (8 bit)

This function executes a exclusive LDR instruction for 8 bit value.

\param [in]  ptr  Pointer to data
\return      value of type uint8_t at (*ptr)
*/
__attribute__((always_inline)) __STATIC_INLINE uint8_t __LDREXB(volatile
uint8_t *addr)
{
    uint32_t result;

#if  (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexb %0, %1" : "=r" (result) : "Q" (*addr) );
#else
    /* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not

```

```

        accepted by assembler. So has to use following less efficient
pattern.
 */
__ASM volatile ("ldrexb %0, [%1]" : "=r" (result) : "r" (addr) :
"memory" );
#endif
    return ((uint8_t) result);      /* Add explicit type cast here */
}

/** \brief LDR Exclusive (16 bit)

This function executes a exclusive LDR instruction for 16 bit values.

\param [in]     ptr  Pointer to data
\return         value of type uint16_t at (*ptr)
*/
_attribute_((always_inline)) __STATIC_INLINE uint16_t __LDREXH(volatile
uint16_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrexh %0, %1" : "=r" (result) : "Q" (*addr) );
#else
    /* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
       accepted by assembler. So has to use following less efficient
pattern.
 */
    __ASM volatile ("ldrexh %0, [%1]" : "=r" (result) : "r" (addr) :
"memory" );
#endif
    return ((uint16_t) result);      /* Add explicit type cast here */
}

/** \brief LDR Exclusive (32 bit)

This function executes a exclusive LDR instruction for 32 bit values.

\param [in]     ptr  Pointer to data
\return         value of type uint32_t at (*ptr)
*/
_attribute_((always_inline)) __STATIC_INLINE uint32_t __LDREXW(volatile
uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrex %0, %1" : "=r" (result) : "Q" (*addr) );
    return(result);
}

/** \brief STR Exclusive (8 bit)

```

This function executes a exclusive STR instruction for 8 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return      0  Function succeeded
\return      1  Function failed
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXB(uint8_t
value, volatile uint8_t *addr)
{
    uint32_t result;

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) :
"r" ((uint32_t)value) );
    return(result);
}
```

/\*\* \brief STR Exclusive (16 bit)

This function executes a exclusive STR instruction for 16 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return      0  Function succeeded
\return      1  Function failed
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXH(uint16_t
value, volatile uint16_t *addr)
{
    uint32_t result;

    __ASM volatile ("stre %0, %2, %1" : "=r" (result), "=Q" (*addr) :
"r" ((uint32_t)value) );
    return(result);
}
```

/\*\* \brief STR Exclusive (32 bit)

This function executes a exclusive STR instruction for 32 bit values.

```
\param [in]  value  Value to store
\param [in]  ptr    Pointer to location
\return      0  Function succeeded
\return      1  Function failed
*/
__attribute__((always_inline)) __STATIC_INLINE uint32_t __STREXW(uint32_t
value, volatile uint32_t *addr)
{
    uint32_t result;
```

```

    __ASM volatile ("strex %0, %2, %1" : "=r" (result), "=Q" (*addr) :
    "r" (value) );
        return(result);
}

/** \brief Remove the exclusive lock

    This function removes the exclusive lock which is created by LDREX.

*/
__attribute__((always_inline)) __STATIC_INLINE void __CLREX(void)
{
    __ASM volatile ("clrex" ::: "memory");
}

/** \brief Signed Saturate

    This function saturates a signed value.

    \param [in]    value  Value to be saturated
    \param [in]    sat    Bit position to saturate to (1..32)
    \return         Saturated value
*/
#define __SSAT(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

/** \brief Unsigned Saturate

    This function saturates an unsigned value.

    \param [in]    value  Value to be saturated
    \param [in]    sat    Bit position to saturate to (0..31)
    \return         Saturated value
*/
#define __USAT(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)); \
    __RES; \
})

/** \brief Rotate Right with Extend (32 bit)

    This function moves each bit of a bitstring right by one bit.

```

The carry input is shifted in at the left end of the bitstring.

```
\param [in]      value  Value to rotate
\return          Rotated value
*/
_attribute__((always_inline)) __STATIC_INLINE uint32_t __RRX(uint32_t
value)
{
    uint32_t result;

    __ASM volatile ("rrx %0, %1" : __CMSIS_GCC_OUT_REG (result) :
    __CMSIS_GCC_USE_REG (value) );
    return(result);
}
```

```
/** \brief LDRT Unprivileged (8 bit)
```

This function executes a Unprivileged LDRT instruction for 8 bit value.

```
\param [in]      ptr  Pointer to data
\return         value of type uint8_t at (*ptr)
*/
_attribute__((always_inline)) __STATIC_INLINE uint8_t __LDRBT(volatile
uint8_t *addr)
{
    uint32_t result;

#if (__GNUC__ > 4) || (__GNUC__ == 4 && __GNUC_MINOR__ >= 8)
    __ASM volatile ("ldrbt %0, %1" : "=r" (result) : "Q" (*addr) );
#else
    /* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
       accepted by assembler. So has to use following less efficient
       pattern.
    */
    __ASM volatile ("ldrbt %0, [%1]" : "=r" (result) : "r" (addr) :
    "memory" );
#endif
    return ((uint8_t) result);    /* Add explicit type cast here */
}
```

```
/** \brief LDRT Unprivileged (16 bit)
```

This function executes a Unprivileged LDRT instruction for 16 bit values.

```
\param [in]      ptr  Pointer to data
\return         value of type uint16_t at (*ptr)
*/
_attribute__((always_inline)) __STATIC_INLINE uint16_t __LDRHT(volatile
uint16_t *addr)
{
```

```

    uint32_t result;

#if (_GNUC__ > 4) || (_GNUC__ == 4 && _GNUC_MINOR_ >= 8)
    __ASM volatile ("ldrht %0, %1" : "=r" (result) : "Q" (*addr) );
#else
    /* Prior to GCC 4.8, "Q" will be expanded to [rx, #0] which is not
       accepted by assembler. So has to use following less efficient
       pattern.
    */
    __ASM volatile ("ldrht %0, [%1]" : "=r" (result) : "r" (addr) :
"memory");
#endif
    return ((uint16_t) result);      /* Add explicit type cast here */
}

```

/\*\* \brief LDRT Unprivileged (32 bit)

This function executes a Unprivileged LDRT instruction for 32 bit values.

```

\param [in]     ptr  Pointer to data
\return        value of type uint32_t at (*ptr)
*/
_attribute_((always_inline)) __STATIC_INLINE uint32_t __LDRT(volatile
uint32_t *addr)
{
    uint32_t result;

    __ASM volatile ("ldrt %0, %1" : "=r" (result) : "Q" (*addr) );
    return(result);
}

```

/\*\* \brief STRT Unprivileged (8 bit)

This function executes a Unprivileged STRT instruction for 8 bit values.

```

\param [in]   value  Value to store
\param [in]   ptr    Pointer to location
*/
_attribute_((always_inline)) __STATIC_INLINE void __STRBT(uint8_t
value, volatile uint8_t *addr)
{
    __ASM volatile ("strbt %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value)
);
}

```

/\*\* \brief STRT Unprivileged (16 bit)

This function executes a Unprivileged STRT instruction for 16 bit values.

```

    \param [in]  value  Value to store
    \param [in]  ptr    Pointer to location
*/
__attribute__((always_inline)) __STATIC_INLINE void __STRHT(uint16_t
value, volatile uint16_t *addr)
{
    __ASM volatile ("strht %1, %0" : "=Q" (*addr) : "r" ((uint32_t)value)
);
}

/** \brief  STRT Unprivileged (32 bit)

    This function executes a Unprivileged STRT instruction for 32 bit
values.

    \param [in]  value  Value to store
    \param [in]  ptr    Pointer to location
*/
__attribute__((always_inline)) __STATIC_INLINE void __STRT(uint32_t
value, volatile uint32_t *addr)
{
    __ASM volatile ("strt %1, %0" : "=Q" (*addr) : "r" (value) );
}

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#ifndef __ICCARM__
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#ifndef __TMS470__
/* TI CCS specific functions */
#include <cmsis_ccs.h>

```

```

#ifndef __TASKING__
/* TASKING arm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the
compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
*/

```

```

#ifndef __CSMC__
/* COSMIC Compiler -----*/

```

```

/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@}*/ /* end of group CMSIS_Core_InstructionInterface */

#endif /* __CORE_CMINSTR_H */
***** */
** */
* @file      core_cmFunc.h
* @brief     CMSIS Cortex-M Core Function Access Header File
* @version   V4.10
* @date      18. March 2015
*
* @note
*
***** */
****/
/* Copyright (c) 2009 - 2015 ARM LIMITED

```

All rights reserved.  
 Redistribution and use in source and binary forms, with or without  
 modification, are permitted provided that the following conditions are  
 met:  

- Redistributions of source code must retain the above copyright  
 notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright  
 notice, this list of conditions and the following disclaimer in the  
 documentation and/or other materials provided with the distribution.
- Neither the name of ARM nor the names of its contributors may be  
 used  
 to endorse or promote products derived from this software without  
 specific prior written permission.

 \*
 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS  
 "AS IS"  
 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
 THE  
 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR  
 PURPOSE  
 ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS  
 BE  
 LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF  
 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR  
 BUSINESS  
 INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER  
 IN  
 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR  
 OTHERWISE)  
 ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF  
 THE

```

POSSIBILITY OF SUCH DAMAGE.
-----
-----*/



#ifndef __CORE_CMFUNC_H
#define __CORE_CMFUNC_H


/* ##### Core Function Access
#####
/** \ingroup CMSIS_Core_FunctionInterface
    \defgroup CMSIS_Core_RegAccFunctions CMSIS Core Register Access
Functions
{
    @{
}

#if defined (__CC_ARM) /*-----RealView Compiler -----
-----*/
/* ARM armcc specific functions */

#if (_ARMCC_VERSION < 400677)
    #error "Please use ARM Compiler Toolchain V4.0.677 or later!"
#endif

/* intrinsic void __enable_irq();      */
/* intrinsic void __disable_irq();     */

/** \brief Get Control Register

    This function returns the content of the Control Register.

    \return          Control Register value
*/
STATIC_INLINE uint32_t __get_CONTROL(void)
{
    register uint32_t __regControl           __ASM("control");
    return(__regControl);
}

/** \brief Set Control Register

    This function writes the given value to the Control Register.

    \param [in]      control  Control Register value to set
*/
STATIC_INLINE void __set_CONTROL(uint32_t control)
{
    register uint32_t __regControl           __ASM("control");
    __regControl = control;
}

```

```

/** \brief Get IPSR Register

This function returns the content of the IPSR Register.

\return IPSR Register value
*/
STATIC_INLINE uint32_t __get_IPSR(void)
{
    register uint32_t __regIPSR           __ASM("ipsr");
    return(__regIPSR);
}

/** \brief Get APSR Register

This function returns the content of the APSR Register.

\return APSR Register value
*/
STATIC_INLINE uint32_t __get_APSR(void)
{
    register uint32_t __regAPSR           __ASM("apsr");
    return(__regAPSR);
}

/** \brief Get xPSR Register

This function returns the content of the xPSR Register.

\return xPSR Register value
*/
STATIC_INLINE uint32_t __get_xPSR(void)
{
    register uint32_t __regXPSR           __ASM("xpsr");
    return(__regXPSR);
}

/** \brief Get Process Stack Pointer

This function returns the current value of the Process Stack Pointer
(PSP).

\return PSP Register value
*/
STATIC_INLINE uint32_t __get_PSP(void)
{
    register uint32_t __regProcessStackPointer __ASM("psp");
    return(__regProcessStackPointer);
}

/** \brief Set Process Stack Pointer

```

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in]      topOfProcStack  Process Stack Pointer value to set
*/
STATIC_INLINE void __set_PSP(uint32_t topOfProcStack)
{
    register uint32_t __regProcessStackPointer  __ASM("psp");
    __regProcessStackPointer = topOfProcStack;
}
```

```
/** \brief Get Main Stack Pointer
```

This function returns the current value of the Main Stack Pointer (MSP).

```
\return          MSP Register value
*/
STATIC_INLINE uint32_t __get_MSP(void)
{
    register uint32_t __regMainStackPointer  __ASM("msp");
    return(__regMainStackPointer);
}
```

```
/** \brief Set Main Stack Pointer
```

This function assigns the given value to the Main Stack Pointer (MSP).

```
\param [in]      topOfMainStack  Main Stack Pointer value to set
*/
STATIC_INLINE void __set_MSP(uint32_t topOfMainStack)
{
    register uint32_t __regMainStackPointer  __ASM("msp");
    __regMainStackPointer = topOfMainStack;
}
```

```
/** \brief Get Priority Mask
```

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return          Priority Mask value
*/
STATIC_INLINE uint32_t __get_PRIMASK(void)
{
    register uint32_t __regPriMask  __ASM("primask");
    return(__regPriMask);
}
```

```

/** \brief Set Priority Mask

    This function assigns the given value to the Priority Mask Register.

        \param [in]     priMask  Priority Mask
*/
STATIC_INLINE void __set_PRIMASK(uint32_t priMask)
{
    register uint32_t __regPriMask           __ASM("primask");
    __regPriMask = (priMask);
}

#if      (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300)

/** \brief Enable FIQ

    This function enables FIQ interrupts by clearing the F-bit in the
CPSR.
    Can only be executed in Privileged modes.
*/
#define __enable_fault_irq           __enable_fiq

/** \brief Disable FIQ

    This function disables FIQ interrupts by setting the F-bit in the
CPSR.
    Can only be executed in Privileged modes.
*/
#define __disable_fault_irq          __disable_fiq

/** \brief Get Base Priority

    This function returns the current value of the Base Priority
register.

        \return          Base Priority register value
*/
STATIC_INLINE uint32_t __get_BASEPRI(void)
{
    register uint32_t __regBasePri           __ASM("basepri");
    return(__regBasePri);
}

/** \brief Set Base Priority

    This function assigns the given value to the Base Priority register.

        \param [in]     basePri  Base Priority value to set
*/

```

```

_ STATIC_INLINE void __set_BASEPRI(uint32_t basePri)
{
    register uint32_t __regBasePri           __ASM("basepri");
    __regBasePri = (basePri & 0xff);
}

/** \brief Set Base Priority with condition

    This function assigns the given value to the Base Priority register
    only if BASEPRI masking is disabled,
    or the new value increases the BASEPRI priority level.

        \param [in]    basePri  Base Priority value to set
*/
_ STATIC_INLINE void __set_BASEPRI_MAX(uint32_t basePri)
{
    register uint32_t __regBasePriMax         __ASM("basepri_max");
    __regBasePriMax = (basePri & 0xff);
}

/** \brief Get Fault Mask

    This function returns the current value of the Fault Mask register.

        \return          Fault Mask register value
*/
_ STATIC_INLINE uint32_t __get_FAULTMASK(void)
{
    register uint32_t __regFaultMask          __ASM("faultmask");
    return(__regFaultMask);
}

/** \brief Set Fault Mask

    This function assigns the given value to the Fault Mask register.

        \param [in]    faultMask  Fault Mask value to set
*/
_ STATIC_INLINE void __set_FAULTMASK(uint32_t faultMask)
{
    register uint32_t __regFaultMask          __ASM("faultmask");
    __regFaultMask = (faultMask & (uint32_t)1);
}

#endif /* (__CORTEX_M >= 0x03) || (__CORTEX_SC >= 300) */

#if      (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)

/** \brief Get FPSCR

```

This function returns the current value of the Floating Point Status/Control register.

```
\return          Floating Point Status/Control register value
*/
__STATIC_INLINE uint32_t __get_FPSCR(void)
{
#if (_FPU_PRESENT == 1) && (_FPU_USED == 1)
    register uint32_t __regfpSCR      __ASM("fpSCR");
    return(__regfpSCR);
#else
    return(0);
#endif
}
```

/\*\* \brief Set FPSCR

This function assigns the given value to the Floating Point Status/Control register.

```
\param [in]    fpSCR  Floating Point Status/Control value to set
*/
__STATIC_INLINE void __set_FPSCR(uint32_t fpSCR)
{
#if (_FPU_PRESENT == 1) && (_FPU_USED == 1)
    register uint32_t __regfpSCR      __ASM("fpSCR");
    __regfpSCR = (fpSCR);
#endif
}

#endif /* (_CORTEX_M == 0x04) || (_CORTEX_M == 0x07) */

#ifndef __GNUC__ /*----- GNU Compiler -----
/* GNU gcc specific functions */

/** \brief Enable IRQ Interrupts

This function enables IRQ interrupts by clearing the I-bit in the CPSR.
Can only be executed in Privileged modes.
*/
__attribute__(( always_inline )) __STATIC_INLINE void
__enable_irq(void)
{
    __ASM volatile ("cpsie i" : : : "memory");
}
```

/\*\* \brief Disable IRQ Interrupts

This function disables IRQ interrupts by setting the I-bit in the CPSR.
Can only be executed in Privileged modes.

```


/*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__disable_irq(void)
{
    __ASM volatile ("cpsid i" : : : "memory");
}

/** \brief Get Control Register

This function returns the content of the Control Register.

\return Control Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_CONTROL(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, control" : "=r" (result) );
    return(result);
}

/** \brief Set Control Register

This function writes the given value to the Control Register.

\param [in] control Control Register value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_CONTROL(uint32_t control)
{
    __ASM volatile ("MSR control, %0" : : "r" (control) : "memory");
}

/** \brief Get IPSR Register

This function returns the content of the IPSR Register.

\return IPSR Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_IPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, ipsr" : "=r" (result) );
    return(result);
}

/** \brief Get APSR Register


```

This function returns the content of the APSR Register.

```
\return          APSR Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
_get(APSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, apsr" : "=r" (result) );
    return(result);
}
```

/\*\* \brief Get xPSR Register

This function returns the content of the xPSR Register.

```
\return          xPSR Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
_get_xPSR(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, xpsr" : "=r" (result) );
    return(result);
}
```

/\*\* \brief Get Process Stack Pointer

This function returns the current value of the Process Stack Pointer (PSP).

```
\return          PSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
_get_PSP(void)
{
    register uint32_t result;

    __ASM volatile ("MRS %0, psp\n" : "=r" (result) );
    return(result);
}
```

/\*\* \brief Set Process Stack Pointer

This function assigns the given value to the Process Stack Pointer (PSP).

\param [in] topOfProcStack Process Stack Pointer value to set

```


        */
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_PSP(uint32_t topOfProcStack)
{
    __ASM volatile ("MSR psp, %0\n" : : "r" (topOfProcStack) : "sp");
}

/** \brief Get Main Stack Pointer

    This function returns the current value of the Main Stack Pointer
    (MSP).

        \return MSP Register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_MSP(void)
{
    register uint32_t result;

    __ASM volatile ("MRS %0, msp\n" : "=r" (result) );
    return(result);
}

/** \brief Set Main Stack Pointer

    This function assigns the given value to the Main Stack Pointer
    (MSP).

        \param [in]    topOfMainStack  Main Stack Pointer value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_MSP(uint32_t topOfMainStack)
{
    __ASM volatile ("MSR msp, %0\n" : : "r" (topOfMainStack) : "sp");
}

/** \brief Get Priority Mask

    This function returns the current state of the priority mask bit from
    the Priority Mask Register.

        \return Priority Mask value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_PRIMASK(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, primask" : "=r" (result) );
    return(result);
}


```

```

/** \brief Set Priority Mask

    This function assigns the given value to the Priority Mask Register.

    \param [in]     priMask  Priority Mask
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_PRIMASK(uint32_t priMask)
{
    __ASM volatile ("MSR primask, %0" : : "r" (priMask) : "memory");
}

#if      (__CORTEX_M >= 0x03)

/** \brief Enable FIQ

    This function enables FIQ interrupts by clearing the F-bit in the
CPSR.
    Can only be executed in Privileged modes.
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__enable_fault_irq(void)
{
    __ASM volatile ("cpsie f" : : : "memory");
}

/** \brief Disable FIQ

    This function disables FIQ interrupts by setting the F-bit in the
CPSR.
    Can only be executed in Privileged modes.
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__disable_fault_irq(void)
{
    __ASM volatile ("cpsid f" : : : "memory");
}

/** \brief Get Base Priority

    This function returns the current value of the Base Priority
register.

    \return          Base Priority register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_BASEPRI(void)
{
    uint32_t result;
}

```

```

    __ASM volatile ("MRS %0, basepri" : "=r" (result) );
    return(result);
}

/** \brief Set Base Priority

    This function assigns the given value to the Base Priority register.

    \param [in]    basePri  Base Priority value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_BASEPRI(uint32_t value)
{
    __ASM volatile ("MSR basepri, %0" : : "r" (value) : "memory");
}

/** \brief Set Base Priority with condition

    This function assigns the given value to the Base Priority register
only if BASEPRI masking is disabled,
or the new value increases the BASEPRI priority level.

    \param [in]    basePri  Base Priority value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_BASEPRI_MAX(uint32_t value)
{
    __ASM volatile ("MSR basepri_max, %0" : : "r" (value) : "memory");
}

/** \brief Get Fault Mask

    This function returns the current value of the Fault Mask register.

    \return          Fault Mask register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_FAULTMASK(void)
{
    uint32_t result;

    __ASM volatile ("MRS %0, faultmask" : "=r" (result) );
    return(result);
}

/** \brief Set Fault Mask

    This function assigns the given value to the Fault Mask register.

```

```

        \param [in]      faultMask  Fault Mask value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_FAULTMASK(uint32_t faultMask)
{
    __ASM volatile ("MSR faultmask, %0" : : "r" (faultMask) : "memory");
}

#endif /* (__CORTEX_M >= 0x03) */

#if         (__CORTEX_M == 0x04) || (__CORTEX_M == 0x07)

/** \brief Get FPSCR

    This function returns the current value of the Floating Point
    Status/Control register.

    \return          Floating Point Status/Control register value
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__get_FPSCR(void)
{
#if (_FPU_PRESENT == 1) && (_FPU_USED == 1)
    uint32_t result;

    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMRS %0, fpscr" : "=r" (result) );
    __ASM volatile ("");
    return(result);
#else
    return(0);
#endif
}

/** \brief Set FPSCR

    This function assigns the given value to the Floating Point
    Status/Control register.

    \param [in]      fpscr  Floating Point Status/Control value to set
*/
__attribute__( ( always_inline ) ) __STATIC_INLINE void
__set_FPSCR(uint32_t fpscr)
{
#if (_FPU_PRESENT == 1) && (_FPU_USED == 1)
    /* Empty asm statement works as a scheduling barrier */
    __ASM volatile ("");
    __ASM volatile ("VMSR fpscr, %0" : : "r" (fpscr) : "vfpcc");
    __ASM volatile ("");
#endif
}

```

```

#endif /* (_CORTEX_M == 0x04) || (_CORTEX_M == 0x07) */

#ifndef __CMSIS_H
#define __CMSIS_H

#if defined ( __CORTEX_M ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING arm specific functions */
/*
 * The CMSIS functions have been implemented as intrinsics in the
compiler.
 * Please use "carm -?i" to get an up to date list of all intrinsics,
 * Including the CMSIS ones.
 */

#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif /* __CMSIS_H */

/*@} end of CMSIS_Core_RegAccFunctions */

#endif /* __CORE_CMFUNC_H */
*****//**
 * @file      core_cm0plus.h
 * @brief     CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
 * @version   V4.10
 * @date      18. March 2015
 *
 * @note
 *
*****/
/* Copyright (c) 2009 - 2015 ARM LIMITED

All rights reserved.
Redistribution and use in source and binary forms, with or without

```

modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  - Neither the name of ARM nor the names of its contributors may be used
- to endorse or promote products derived from this software without specific prior written permission.
- \*

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS BE

LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.

-----  
----\*/

```
#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA
check */
#endif
```

```
#ifndef __CORE_CM0PLUS_H_GENERIC
#define __CORE_CM0PLUS_H_GENERIC
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions
CMSIS violates the following MISRA-C:2004 rules:
```

\li Required Rule 8.5, object/function definition in header file.<br>
Function definitions in header files are used to allow 'inlining'.

```
\li Required Rule 18.4, declaration of union type or object of union
type: '{...}'.<br>
```

```
    Unions are used for effective representation of core registers.
```

```
\li Advisory Rule 19.7, Function-like macro defined.<br>
```

```
    Function-like macros are used to allow more efficient code.
```

```
*/
```

```
*****  
* CMSIS definitions  
*****  
** \ingroup Cortex-M0+  
@{  
/* CMSIS CM0P definitions */  
#define __CM0PLUS_CMSIS_VERSION_MAIN (0x04)  
/*!< [31:16] CMSIS HAL main version */  
#define __CM0PLUS_CMSIS_VERSION_SUB (0x00)  
/*!< [15:0] CMSIS HAL sub version */  
#define __CM0PLUS_CMSIS_VERSION ((__CM0PLUS_CMSIS_VERSION_MAIN <<  
16) | \  
/*!< CMSIS HAL version number */  
/* CMSIS HAL version number */  
  
#define __CORTEX_M (0x00)  
/*!< Cortex-M Core */  
  
#if defined (__CC_ARM )  
#define __ASM __asm  
/*!< asm keyword for ARM Compiler */  
#define __INLINE __inline  
/*!< inline keyword for ARM Compiler */  
#define __STATIC_INLINE static __inline  
  
#elif defined ( __GNUC__ )  
#define __ASM __asm  
/*!< asm keyword for GNU Compiler */  
#define __INLINE __inline  
/*!< inline keyword for GNU Compiler */  
#define __STATIC_INLINE static inline  
  
#elif defined ( __ICCARM__ )  
#define __ASM __asm  
/*!< asm keyword for IAR Compiler */  
#define __INLINE __inline  
/*!< inline keyword for IAR Compiler. Only available in High optimization  
mode! */  
#define __STATIC_INLINE static inline
```

```

#endif

#ifndef __ASM_H
#define __ASM_H

/*< asm keyword for TI CCS Compiler      */
#define __STATIC_INLINE static inline

/*< inline keyword for TASKING Compiler   */
#define __INLINE inline
/*< inline keyword for TASKING Compiler   */
#define __STATIC_INLINE static inline

/*< packed keyword for COSMIC Compiler    */
#define __packed
/*< asm keyword for COSMIC Compiler      */
#define __ASM asm
/*< inline keyword for COSMIC Compiler    */
#define __INLINE inline
/*< inline keyword for COSMIC Compiler    */
#define __STATIC_INLINE static inline

/*< !< inline keyword for COSMIC Compiler */
#endif

/*< FPU_USED indicates whether an FPU is used or not.
   This core does not support an FPU at all
*/
#define __FPU_USED 0

#if defined ( __CC_ARM )
  #if defined __TARGET_FPU_VFP
    #warning "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
  #endif

  #elif defined ( __GNUC__ )
    #if defined ( __VFP_FP__ ) && !defined( __SOFTFP__ )
      #warning "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
    #endif

  #elif defined ( __ICCARM__ )
    #if defined __ARMVFP__
      #warning "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
    #endif

  #elif defined ( __TMS470__ )
    #if defined __TI_VFP_SUPPORT__
      #warning "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
    #endif

  #elif defined ( __TASKING__ )
    #if defined __FPU_VFP__

```

```

    #error "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
#endif

#elif defined ( __CSMC__ )           /* Cosmic */
#if ( __CSMC__ & 0x400)           // FPU present for parser
    #error "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
#endif
#endif

#include <stdint.h>                  /* standard types definitions
*/
#include <core_cmInstr.h>             /* Core Instruction Access
*/
#include <core_cmFunc.h>              /* Core Function Access
*/
#endif

#ifndef __cplusplus
}
#endif

#endif /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

#ifndef __cplusplus
extern "C" {
#endif

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
    #ifndef __CM0PLUS_REV
        #define __CM0PLUS_REV          0x0000
        #warning "__CM0PLUS_REV not defined in device header file; using
default!"
    #endif

    #ifndef __MPU_PRESENT
        #define __MPU_PRESENT          0
        #warning "__MPU_PRESENT not defined in device header file; using
default!"
    #endif

    #ifndef __VTOR_PRESENT
        #define __VTOR_PRESENT          0
        #warning "__VTOR_PRESENT not defined in device header file; using
default!"
    #endif

#ifndef __NVIC_PRIO_BITS

```

```

#define __NVIC_PRIO_BITS          2
#warning "__NVIC_PRIO_BITS not defined in device header file; using
default!"
#endif

#ifndef __Vendor_SysTickConfig
#define __Vendor_SysTickConfig      0
#warning "__Vendor_SysTickConfig not defined in device header file;
using default!"
#endif
#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
    \defgroup CMSIS_glob_defs CMSIS Global Defines

    <strong>IO Type Qualifiers</strong> are used
    \li to specify the access to peripheral variables.
    \li for automatic generation of peripheral register debug
information.
*/
#ifdef __cplusplus
#define __I     volatile           /*!< Defines 'read only'
permissions */ 
#else
#define __I     volatile const    /*!< Defines 'read only'
permissions */ 
#endif
#define __O     volatile           /*!< Defines 'write only'
permissions */ 
#define __IO    volatile           /*!< Defines 'read / write'
permissions */ 

/*@} end of group Cortex-M0+ */

```

```

*****
*             Register Abstraction
Core Register contain:
- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core MPU Register

*****
/** \defgroup CMSIS_core_register Defines and Type Definitions
\brief Type definitions and defines for Cortex-M processor based
devices.
*/

```

```

/** \ingroup CMSIS_core_register
 * \defgroup CMSIS_CORE Status and Control Registers
 * \brief Core Register type definitions.
 @{
 */

/** \brief Union type to access the Application Program Status Register (APSR).
 */
typedef union
{
    struct
    {
        uint32_t _reserved0:28;           /*!< bit: 0..27 Reserved
*/
        uint32_t V:1;                  /*!< bit: 28 Overflow
        condition code flag          */
        uint32_t C:1;                  /*!< bit: 29 Carry
        condition code flag          */
        uint32_t Z:1;                  /*!< bit: 30 Zero condition
        code flag                   */
        uint32_t N:1;                  /*!< bit: 31 Negative
        condition code flag          */
        } b;
    access                         /*!< Structure used for bit
        uint32_t w;                  /*!< Type used for word
    access                         */
} APSR_Type;

/* APSR Register Definitions */
#define APSR_N_Pos                 31
/*!< APSR: N Position */
#define APSR_N_Msk                (1UL << APSR_N_Pos)
/*!< APSR: N Mask */

#define APSR_Z_Pos                 30
/*!< APSR: Z Position */
#define APSR_Z_Msk                (1UL << APSR_Z_Pos)
/*!< APSR: Z Mask */

#define APSR_C_Pos                 29
/*!< APSR: C Position */
#define APSR_C_Msk                (1UL << APSR_C_Pos)
/*!< APSR: C Mask */

#define APSR_V_Pos                 28
/*!< APSR: V Position */
#define APSR_V_Msk                (1UL << APSR_V_Pos)
/*!< APSR: V Mask */

/** \brief Union type to access the Interrupt Program Status Register (IPSR).
 */

```

```

typedef union
{
    struct
    {
        uint32_t ISR:9;                                /*!< bit: 0.. 8 Exception
number                               */
        uint32_t _reserved0:23;                         /*!< bit: 9..31 Reserved
*/
    } b;
access                                */
    uint32_t w;                                     /*!< Type      used for word
access                               */
} IPSR_Type;

/* IPSR Register Definitions */
#define IPSR_ISR_Pos          0
/*!< IPSR: ISR Position */
#define IPSR_ISR_Msk          (0x1FFUL /*<< IPSR_ISR_Pos*/)
/*!< IPSR: ISR Mask */

/** \brief Union type to access the Special-Purpose Program Status
Registers (xPSR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;                                /*!< bit: 0.. 8 Exception
number                               */
        uint32_t _reserved0:15;                          /*!< bit: 9..23 Reserved
*/
        uint32_t T:1;                                   /*!< bit: 24 Thumb bit
(read 0)                           */
        uint32_t _reserved1:3;                          /*!< bit: 25..27 Reserved
*/
        uint32_t V:1;                                   /*!< bit: 28 Overflow
condition code flag               */
        uint32_t C:1;                                   /*!< bit: 29 Carry
condition code flag               */
        uint32_t Z:1;                                   /*!< bit: 30 Zero condition
code flag                            */
        uint32_t N:1;                                   /*!< bit: 31 Negative
condition code flag               */
        uint32_t _reserved2:1;                          /*!< bit: 32 Reserved
*/
    } b;
access                                */
    uint32_t w;                                     /*!< Type      used for word
access                               */
} xPSR_Type;

/* xPSR Register Definitions */
#define xPSR_N_Pos            31
/*!< xPSR: N Position */

```

```

#define xPSR_N_Msk (1UL << xPSR_N_Pos)
/*!< xPSR: N Mask */

#define xPSR_Z_Pos 30
/*!< xPSR: Z Position */
#define xPSR_Z_Msk (1UL << xPSR_Z_Pos)
/*!< xPSR: Z Mask */

#define xPSR_C_Pos 29
/*!< xPSR: C Position */
#define xPSR_C_Msk (1UL << xPSR_C_Pos)
/*!< xPSR: C Mask */

#define xPSR_V_Pos 28
/*!< xPSR: V Position */
#define xPSR_V_Msk (1UL << xPSR_V_Pos)
/*!< xPSR: V Mask */

#define xPSR_T_Pos 24
/*!< xPSR: T Position */
#define xPSR_T_Msk (1UL << xPSR_T_Pos)
/*!< xPSR: T Mask */

#define xPSR_ISR_Pos 0
/*!< xPSR: ISR Position */
#define xPSR_ISR_Msk (0x1FFUL /*<< xPSR_ISR_Pos*/)
/*!< xPSR: ISR Mask */

/** \brief Union type to access the Control Registers (CONTROL).
 */
typedef union
{
    struct
    {
        uint32_t nPRIV:1; /*!< bit: 0 Execution
privilege in Thread mode */
        uint32_t SPSEL:1; /*!< bit: 1 Stack to be
used */
        uint32_t _reserved1:30; /*!< bit: 2..31 Reserved
*/
        } b; /*!< Structure used for bit
access */
        uint32_t w; /*!< Type used for word
access */
    } CONTROL_Type;
}

/* CONTROL Register Definitions */
#define CONTROL_SPSEL_Pos 1
/*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk (1UL << CONTROL_SPSEL_Pos)
/*!< CONTROL: SPSEL Mask */

```

```

#define CONTROL_nPRIV_Pos 0
/*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk (1UL /*<< CONTROL_nPRIV_Pos*/)
/*!< CONTROL: nPRIV Mask */

/*@} end of group CMSIS_CORE */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
\brief Type definitions for the NVIC Registers
{@{
*/
typedef struct
{
    __IO uint32_t ISER[1]; /*!< Offset: 0x000 (R/W) */
    Interrupt Set Enable Register
    uint32_t RESERVED0[31];

    __IO uint32_t ICER[1]; /*!< Offset: 0x080 (R/W) */
    Interrupt Clear Enable Register
    uint32_t RSERVED1[31];

    __IO uint32_t ISPR[1]; /*!< Offset: 0x100 (R/W) */
    Interrupt Set Pending Register
    uint32_t RESERVED2[31];

    __IO uint32_t ICPR[1]; /*!< Offset: 0x180 (R/W) */
    Interrupt Clear Pending Register
    uint32_t RESERVED3[31];
    uint32_t RESERVED4[64];

    __IO uint32_t IP[8]; /*!< Offset: 0x300 (R/W) */
    Interrupt Priority Register
} NVIC_Type;

/*@} end of group CMSIS_NVIC */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers
{@{
*/
typedef struct
{
    __I uint32_t CPUID; /*!< Offset: 0x000 (R/ ) CPUID */
    Base Register
    __IO uint32_t ICSR; /*!< Offset: 0x004 (R/W) */
    Interrupt Control and State Register
#if (__VTOR_PRESENT == 1)

```

```

    __IO uint32_t VTOR;                                /*!< Offset: 0x008 (R/W)  Vector
Table Offset Register
*/
#else
    uint32_t RESERVED0;
#endif
    __IO uint32_t AIRCR;                               /*!< Offset: 0x00C (R/W)
Application Interrupt and Reset Control Register */
    __IO uint32_t SCR;                                /*!< Offset: 0x010 (R/W)  System
Control Register
*/
    __IO uint32_t CCR;                                /*!< Offset: 0x014 (R/W)
Configuration Control Register
*/
    uint32_t RESERVED1;
    __IO uint32_t SHP[2];                             /*!< Offset: 0x01C (R/W)  System
Handlers Priority Registers. [0] is RESERVED */
    __IO uint32_t SHCSR;                             /*!< Offset: 0x024 (R/W)  System
Handler Control and State Register
*/
} SCB_Type;

/* SCB CPUID Register Definitions */
#define SCB_CPUID_IMPLEMENTER_Pos                  24
/*!< SCB CPUID: IMPLEMENTER Position */
#define SCB_CPUID_IMPLEMENTER_Msk                (0xFFUL <<
SCB_CPUID_IMPLEMENTER_Pos)                   /*!< SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos                    20
/*!< SCB CPUID: VARIANT Position */
#define SCB_CPUID_VARIANT_Msk                  (0xFUL <<
SCB_CPUID_VARIANT_Pos)                     /*!< SCB CPUID: VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos            16
/*!< SCB CPUID: ARCHITECTURE Position */
#define SCB_CPUID_ARCHITECTURE_Msk          (0xFUL <<
SCB_CPUID_ARCHITECTURE_Pos)             /*!< SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos                 4
/*!< SCB CPUID: PARTNO Position */
#define SCB_CPUID_PARTNO_Msk               (0xFFFFUL <<
SCB_CPUID_PARTNO_Pos)                  /*!< SCB CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos              0
/*!< SCB CPUID: REVISION Position */
#define SCB_CPUID_REVISION_Msk            (0xFUL /*<<
SCB_CPUID_REVISION_Pos*/)             /*!< SCB CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */
#define SCB_ICSR_NMIPENDSET_Pos           31
/*!< SCB ICSR: NMIPENDSET Position */
#define SCB_ICSR_NMIPENDSET_Msk         (1UL <<
SCB_ICSR_NMIPENDSET_Pos)            /*!< SCB ICSR: NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos          28
/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk        (1UL <<
SCB_ICSR_PENDSVSET_Pos)            /*!< SCB ICSR: PENDSVSET Mask */

```

```

#define SCB_ICSR_PENDSVCLR_Pos 27
/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk (1UL <<
SCB_ICSR_PENDSVCLR_Pos) /*!< SCB ICSR: PENDSVCLR Mask */

#define SCB_ICSR_PENDSTSET_Pos 26
/*!< SCB ICSR: PENDSTSET Position */
#define SCB_ICSR_PENDSTSET_Msk (1UL <<
SCB_ICSR_PENDSTSET_Pos) /*!< SCB ICSR: PENDSTSET Mask */

#define SCB_ICSR_PENDSTCLR_Pos 25
/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk (1UL <<
SCB_ICSR_PENDSTCLR_Pos) /*!< SCB ICSR: PENDSTCLR Mask */

#define SCB_ICSR_ISRPREEMPT_Pos 23
/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk (1UL <<
SCB_ICSR_ISRPREEMPT_Pos) /*!< SCB ICSR: ISRPREEMPT Mask */

#define SCB_ICSR_ISRPENDING_Pos 22
/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk (1UL <<
SCB_ICSR_ISRPENDING_Pos) /*!< SCB ICSR: ISRPENDING Mask */

#define SCB_ICSR_VECTPENDING_Pos 12
/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk (0x1FFUL <<
SCB_ICSR_VECTPENDING_Pos) /*!< SCB ICSR: VECTPENDING Mask */

#define SCB_ICSR_VECTACTIVE_Pos 0
/*!< SCB ICSR: VECTACTIVE Position */
#define SCB_ICSR_VECTACTIVE_Msk (0x1FFUL /*<<
SCB_ICSR_VECTACTIVE_Pos*/) /*!< SCB ICSR: VECTACTIVE Mask */

#if (_VTOR_PRESENT == 1)
/* SCB Interrupt Control State Register Definitions */
#define SCB_VTOR_TBLOFF_Pos 8
/*!< SCB VTOR: TBLOFF Position */
#define SCB_VTOR_TBLOFF_Msk (0xFFFFFUL <<
SCB_VTOR_TBLOFF_Pos) /*!< SCB VTOR: TBLOFF Mask */
#endif

/* SCB Application Interrupt and Reset Control Register Definitions */
#define SCB_AIRCR_VECTKEY_Pos 16
/*!< SCB AIRCR: VECTKEY Position */
#define SCB_AIRCR_VECTKEY_Msk (0xFFFFFUL <<
SCB_AIRCR_VECTKEY_Pos) /*!< SCB AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos 16
/*!< SCB AIRCR: VECTKEYSTAT Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk (0xFFFFFUL <<
SCB_AIRCR_VECTKEYSTAT_Pos) /*!< SCB AIRCR: VECTKEYSTAT Mask */

```

```

#define SCB_AIRCR_ENDIANESS_Pos           15
/*!< SCB AIRCR: ENDIANESS Position */
#define SCB_AIRCR_ENDIANESS_Msk          (1UL <<
SCB_AIRCR_ENDIANESS_Pos)             /*!< SCB AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos        2
/*!< SCB AIRCR: SYSRESETREQ Position */
#define SCB_AIRCR_SYSRESETREQ_Msk        (1UL <<
SCB_AIRCR_SYSRESETREQ_Pos)          /*!< SCB AIRCR: SYSRESETREQ Mask */
*/

#define SCB_AIRCR_VECTCLRACTIVE_Pos      1
/*!< SCB AIRCR: VECTCLRACTIVE Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk      (1UL <<
SCB_AIRCR_VECTCLRACTIVE_Pos)        /*!< SCB AIRCR: VECTCLRACTIVE Mask */
*/

/* SCB System Control Register Definitions */
#define SCB_SCR_SEVONPEND_Pos           4
/*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk          (1UL << SCB_SCR_SEVONPEND_Pos)
/*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos           2
/*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk          (1UL << SCB_SCR_SLEEPDEEP_Pos)
/*!< SCB SCR: SLEEPDEEP Mask */

#define SCB_SCR_SLEEPONEXIT_Pos         1
/*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk        (1UL <<
SCB_SCR_SLEEPONEXIT_Pos)           /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */
#define SCB_CCR_STKALIGN_Pos            9
/*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk          (1UL << SCB_CCR_STKALIGN_Pos)
/*!< SCB CCR: STKALIGN Mask */

#define SCB_CCR_UNALIGN_TRP_Pos         3
/*!< SCB CCR: UNALIGN_TRP Position */
#define SCB_CCR_UNALIGN_TRP_Msk        (1UL <<
SCB_CCR_UNALIGN_TRP_Pos)           /*!< SCB CCR: UNALIGN_TRP Mask */

/* SCB System Handler Control and State Register Definitions */
#define SCB_SHCSR_SVCALLPENDED_Pos     15
/*!< SCB SHCSR: SVCALLPENDED Position */
#define SCB_SHCSR_SVCALLPENDED_Msk     (1UL <<
SCB_SHCSR_SVCALLPENDED_Pos)        /*!< SCB SHCSR: SVCALLPENDED Mask */
*/

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
 * \defgroup CMSIS_SysTick System Tick Timer (SysTick)
 * \brief Type definitions for the System Timer Registers.
 @{
 */

/** \brief Structure type to access the System Timer (SysTick).
 */
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) */
    SysTick Control and Status Register */

    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) */
    SysTick Reload Value Register */

    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) */
    SysTick Current Value Register */

    __I uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) */
    SysTick Calibration Register */
} SysTick_Type;

/* SysTick Control / Status Register Definitions */
#define SysTick_CTRL_COUNTFLAG_Pos      16
/*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk      (1UL <<
SysTick_CTRL_COUNTFLAG_Pos)        /*!< SysTick CTRL: COUNTFLAG Mask */
 */

#define SysTick_CTRL_CLKSOURCE_Pos      2
/*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk      (1UL <<
SysTick_CTRL_CLKSOURCE_Pos)        /*!< SysTick CTRL: CLKSOURCE Mask */
 */

#define SysTick_CTRL_TICKINT_Pos       1
/*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk       (1UL <<
SysTick_CTRL_TICKINT_Pos)         /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos        0
/*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk        (1UL /*<<
SysTick_CTRL_ENABLE_Pos*/)        /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */
#define SysTick_LOAD_RELOAD_Pos        0
/*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk        (0xFFFFFFFFUL /*<<
SysTick_LOAD_RELOAD_Pos*/)        /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */
#define SysTick_VAL_CURRENT_Pos        0
/*!< SysTick VAL: CURRENT Position */

```

```

#define SysTick_VAL_CURRENT_Msk          (0xFFFFFFFFUL /*<<
SysTick_VAL_CURRENT_Pos*/) /*!< SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */
#define SysTick_CALIB_NOREF_Pos         31
/*!< SysTick CALIB: NOREF Position */
#define SysTick_CALIB_NOREF_Msk         (1UL <<
SysTick_CALIB_NOREF_Pos) /*!< SysTick CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos          30
/*!< SysTick CALIB: SKEW Position */
#define SysTick_CALIB_SKEW_Msk          (1UL <<
SysTick_CALIB_SKEW_Pos) /*!< SysTick CALIB: SKEW Mask */

#define SysTick_CALIB_TENMS_Pos         0
/*!< SysTick CALIB: TENMS Position */
#define SysTick_CALIB_TENMS_Msk         (0xFFFFFFFFUL /*<<
SysTick_CALIB_TENMS_Pos*/) /*!< SysTick CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

#if (_MPU_PRESENT == 1)
/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUs Memory Protection Unit (MPU)
\brief Type definitions for the Memory Protection Unit (MPU)
{@{
*/
/** \brief Structure type to access the Memory Protection Unit (MPU).
*/
typedef struct
{
    __I uint32_t TYPE;                      /*!< Offset: 0x000 (R/ ) MPU
                                             */
    __IO uint32_t CTRL;                     /*!< Offset: 0x004 (R/W) MPU
                                             */
    Control Register
    __IO uint32_t RNR;                      /*!< Offset: 0x008 (R/W) MPU
                                             */
    Region RNRber Register
    __IO uint32_t RBAR;                     /*!< Offset: 0x00C (R/W) MPU
                                             */
    Region Base Address Register
    __IO uint32_t RASR;                     /*!< Offset: 0x010 (R/W) MPU
                                             */
    Region Attribute and Size Register
} MPU_Type;

/* MPU Type Register */
#define MPU_TYPE_IREGION_Pos              16
/*!< MPU TYPE: IREGION Position */
#define MPU_TYPE_IREGION_Msk              (0xFFUL <<
MPU_TYPE_IREGION_Pos) /*!< MPU TYPE: IREGION Mask */

#define MPU_TYPE_DREGION_Pos              8
/*!< MPU TYPE: DREGION Position */
#define MPU_TYPE_DREGION_Msk              (0xFFUL <<
MPU_TYPE_DREGION_Pos) /*!< MPU TYPE: DREGION Mask */

```

```

#define MPU_TYPE_SEPARATE_Pos 0
/*!< MPU TYPE: SEPARATE Position */
#define MPU_TYPE_SEPARATE_Msk (1UL /*<<
MPU_TYPE_SEPARATE_Pos*) /*!< MPU TYPE: SEPARATE Mask */

/* MPU Control Register */
#define MPU_CTRL_PRIVDEFENA_Pos 2
/*!< MPU CTRL: PRIVDEFENA Position */
#define MPU_CTRL_PRIVDEFENA_Msk (1UL <<
MPU_CTRL_PRIVDEFENA_Pos) /*!< MPU CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos 1
/*!< MPU CTRL: HFNMIENA Position */
#define MPU_CTRL_HFNMIENA_Msk (1UL << MPU_CTRL_HFNMIENA_Pos)
/*!< MPU CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos 0
/*!< MPU CTRL: ENABLE Position */
#define MPU_CTRL_ENABLE_Msk (1UL /*<<
MPU_CTRL_ENABLE_Pos) /*!< MPU CTRL: ENABLE Mask */

/* MPU Region Number Register */
#define MPU_RNR_REGION_Pos 0
/*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk (0xFFFUL /*<<
MPU_RNR_REGION_Pos) /*!< MPU RNR: REGION Mask */

/* MPU Region Base Address Register */
#define MPU_RBAR_ADDR_Pos 8
/*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk (0xFFFFFUL <<
MPU_RBAR_ADDR_Pos) /*!< MPU RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos 4
/*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk (1UL << MPU_RBAR_VALID_Pos)
/*!< MPU RBAR: VALID Mask */

#define MPU_RBAR_REGION_Pos 0
/*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk (0xFUL /*<<
MPU_RBAR_REGION_Pos) /*!< MPU RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */
#define MPU_RASR_ATTRS_Pos 16
/*!< MPU RASR: MPU Region Attribute field Position */
#define MPU_RASR_ATTRS_Msk (0xFFFFFUL <<
MPU_RASR_ATTRS_Pos) /*!< MPU RASR: MPU Region Attribute
field Mask */

#define MPU_RASR_XN_Pos 28
/*!< MPU RASR: ATTRS.XN Position */

```

```

#define MPU_RASR_XN_Msk          (1UL << MPU_RASR_XN_Pos)
/*!< MPU RASR: ATTRS.XN Mask */

#define MPU_RASR_AP_Pos          24
/*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk          (0x7UL << MPU_RASR_AP_Pos)
/*!< MPU RASR: ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos          19
/*!< MPU RASR: ATTRS.TEX Position */
#define MPU_RASR_TEX_Msk          (0x7UL << MPU_RASR_TEX_Pos)
/*!< MPU RASR: ATTRS.TEX Mask */

#define MPU_RASR_S_Pos             18
/*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk             (1UL << MPU_RASR_S_Pos)
/*!< MPU RASR: ATTRS.S Mask */

#define MPU_RASR_C_Pos             17
/*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk             (1UL << MPU_RASR_C_Pos)
/*!< MPU RASR: ATTRS.C Mask */

#define MPU_RASR_B_Pos             16
/*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk             (1UL << MPU_RASR_B_Pos)
/*!< MPU RASR: ATTRS.B Mask */

#define MPU_RASR_SRD_Pos            8
/*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk           (0xFFUL << MPU_RASR_SRD_Pos)
/*!< MPU RASR: Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos           1
/*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk           (0x1FUL << MPU_RASR_SIZE_Pos)
/*!< MPU RASR: Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos          0
/*!< MPU RASR: Region enable bit Position */
#define MPU_RASR_ENABLE_Msk          (1UL /*<<
MPU_RASR_ENABLE_Pos*/)           /*!< MPU RASR: Region enable bit
Disable Mask */

/*@} end of group CMSIS_MPUS */
#endif

/** \ingroup CMSIS_core_register
\defgroup CMSIS_CoreDebug      Core Debug Registers (CoreDebug)
\brief      Cortex-M0+ Core Debug Registers (DCB registers, SHCSR,
and DFSR)
are only accessible over DAP and not via processor.
Therefore

```

```

    they are not covered by the Cortex-M0 header file.

{@{
*/
/*@} end of group CMSIS_CoreDebug */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_core_base Core Definitions
 \brief Definitions for base addresses, unions, and structures.
{@{
*/
/*
 * Memory mapping of Cortex-M0+ Hardware */
#define SCS_BASE (0xE000E000UL)
/*!< System Control Space Base Address */
#define SysTick_BASE (SCS_BASE + 0x0010UL)
/*!< SysTick Base Address */
#define NVIC_BASE (SCS_BASE + 0x0100UL)
/*!< NVIC Base Address */
#define SCB_BASE (SCS_BASE + 0x0D00UL)
/*!< System Control Block Base Address */

#define SCB ((SCB_Type *) SCB_BASE)
/*!< SCB configuration struct */
#define SysTick ((SysTick_Type *) SysTick_BASE)
/*!< SysTick configuration struct */
#define NVIC ((NVIC_Type *) NVIC_BASE)
/*!< NVIC configuration struct */

#if (_MPU_PRESENT == 1)
#define MPU_BASE (SCS_BASE + 0x0D90UL)
/*!< Memory Protection Unit */
#define MPU ((MPU_Type *) MPU_BASE)
/*!< Memory Protection Unit */
#endif

/*@} */

/*****
*          Hardware Abstraction Layer
Core Function Interface contains:
- Core NVIC Functions
- Core SysTick Functions
- Core Register Access Functions

*****/
/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/

```

```

/* ##### NVIC functions
#####
*/
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions
\brief Functions that manage interrupts and exceptions via the
NVIC.
{
}

/* Interrupt Priorities are WORD accessible only under ARMv6M
*/
/* The following MACROS handle generation of the register offset and byte
masks */
#define _BIT_SHIFT(IRQn)          ( (((uint32_t)(int32_t)(IRQn)) \
) & 0x03UL) * 8UL)
#define _SHP_IDX(IRQn)           ( (((((uint32_t)(int32_t)(IRQn)) &
0x0FUL)-8UL) >> 2UL) )
#define _IP_IDX(IRQn)            ( (((uint32_t)(int32_t)(IRQn)) \
>> 2UL) )

/** \brief Enable External Interrupt

The function enables a device-specific interrupt in the NVIC
interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be
negative.
*/
STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
    NVIC->ISER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &
0x1FUL));
}

/** \brief Disable External Interrupt

The function disables a device-specific interrupt in the NVIC
interrupt controller.

\param [in] IRQn External interrupt number. Value cannot be
negative.
*/
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
{
    NVIC->ICER[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) &
0x1FUL));
}

/** \brief Get Pending Interrupt

```

The function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

```
\param [in]      IRQn  Interrupt number.

\return          0  Interrupt status is not pending.
\return          1  Interrupt status is pending.
*/
STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
{
    return((uint32_t)((NVIC->ISPR[0] & (1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0UL);
}
```

/\*\* \brief Set Pending Interrupt

The function sets the pending bit of an external interrupt.

```
\param [in]      IRQn  Interrupt number. Value cannot be negative.
*/
STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ISPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

/\*\* \brief Clear Pending Interrupt

The function clears the pending bit of an external interrupt.

```
\param [in]      IRQn  External interrupt number. Value cannot be negative.
*/
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ICPR[0] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

/\*\* \brief Set Interrupt Priority

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

```
\param [in]      IRQn  Interrupt number.
\param [in]      priority  Priority to set.
*/
STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
```

```

{
    if((int32_t)(IRQn) < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = ((uint32_t)(SCB->SHP[_SHP_IDX(IRQn)] &
~(0xFFUL << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) <<
            _BIT_SHIFT(IRQn)));
    }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = ((uint32_t)(NVIC->IP[_IP_IDX(IRQn)] &
~(0xFFUL << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL) <<
            _BIT_SHIFT(IRQn)));
    }
}

```

/\*\* \brief Get Interrupt Priority

The function reads the priority of an interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

```

\param [in]    IRQn   Interrupt number.
\return         Interrupt Priority. Value is aligned
automatically to the implemented
                           priority bits of the microcontroller.
*/
STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{

    if((int32_t)(IRQn) < 0) {
        return((uint32_t)((SCB->SHP[_SHP_IDX(IRQn)] >> _BIT_SHIFT(IRQn)) &
(uint32_t)0xFFUL) >> (8 - __NVIC_PRIO_BITS));
    }
    else {
        return((uint32_t)((NVIC->IP[_IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn)) &
(uint32_t)0xFFUL) >> (8 - __NVIC_PRIO_BITS));
    }
}

```

/\*\* \brief System Reset

```

The function initiates a system reset request to reset the MCU.
*/
STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                     /* Ensure
all outstanding memory accesses included

buffered write are completed before reset */
    SCB->AIRCR = ((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                    SCB_AIRCR_SYSRESETREQ_Msk);
}

```

```

    __DSB();                                     /* Ensure
completion of memory access */
    while(1) { __NOP(); }                         /* wait
until reset */
}

/*@} end of CMSIS_Core_NVICFunctions */

/* ##### SysTick function
#####
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_SysTickFunctions SysTick Functions
\brief Functions that configure the System.
{@{
*/
#endif

#if (__Vendor_SysTickConfig == 0)

/** \brief System Tick Configuration

The function initializes the System Timer and its interrupt, and
starts the System Tick Timer.
Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.
\return 1 Function failed.

\note When the variable <b>__Vendor_SysTickConfig</b> is set to
1, then the
function <b>SysTick_Config</b> is not included. In this case, the
file <b><i>device</i>.h</b>
must contain a vendor-specific implementation of this function.

*/
STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) {return (1UL); }           /*
Reload value impossible */

    SysTick->LOAD = (uint32_t)(ticks - 1UL);                                /*
set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for Systick Interrupt */
    SysTick->VAL = 0UL;                                                 /*
Load the SysTick Counter Value */
    SysTick->CTRL = SysTick_CTRL_CLKSOURCE_Msk |
                    SysTick_CTRL_TICKINT_Msk |                               /*
                    SysTick_CTRL_ENABLE_Msk;                                */
    Enable SysTick IRQ and SysTick Timer */
}

```

```

    return (0UL);                                /*
Function successful */
}

#endif

/*@} end of CMSIS_Core_SysTickFunctions */

#ifndef __cplusplus
}
#endif /* __CORE_CM0PLUS_H_DEPENDANT */

#endif /* __CMSIS_GENERIC */
*****//**
 * @file      core_cm4.h
 * @brief     CMSIS Cortex-M4 Core Peripheral Access Layer Header File
 * @version   V4.10
 * @date      18. March 2015
 *
 * @note
 *
*****/
/* Copyright (c) 2009 - 2015 ARM LIMITED

All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
- Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.
- Neither the name of ARM nor the names of its contributors may be
used
  to endorse or promote products derived from this software without
  specific prior written permission.
*
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS
BE

```

LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-----

----\*/

```
#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA
check */
#endif

#ifndef __CORE_CM4_H_GENERIC
#define __CORE_CM4_H_GENERIC

#ifdef __cplusplus
extern "C" {
#endif

/** \page CMSIS_MISRA_Exceptions MISRA-C:2004 Compliance Exceptions
CMSIS violates the following MISRA-C:2004 rules:

\li Required Rule 8.5, object/function definition in header file.<br>
Function definitions in header files are used to allow 'inlining'.

\li Required Rule 18.4, declaration of union type or object of union
type: '{...}'.<br>
Unions are used for effective representation of core registers.

\li Advisory Rule 19.7, Function-like macro defined.<br>
Function-like macros are used to allow more efficient code.
*/



/********************* CMSIS definitions
*****/


/** \ingroup Cortex_M4
@{
*/
/* CMSIS CM4 definitions */
```

```

#define __CM4_CMSIS_VERSION_MAIN (0x04)
/*!< [31:16] CMSIS HAL main version */
#define __CM4_CMSIS_VERSION_SUB (0x00)
/*!< [15:0] CMSIS HAL sub version */
#define __CM4_CMSIS_VERSION (((__CM4_CMSIS_VERSION_MAIN << 16) | \
                           __CM4_CMSIS_VERSION_SUB))
/*!< CMSIS HAL version number */

#define __CORTEX_M (0x04)
/*!< Cortex-M Core */

#if defined (__CC_ARM)
#define __ASM __asm
/*!< asm keyword for ARM Compiler */
#define __INLINE __inline
/*!< inline keyword for ARM Compiler */
#define __STATIC_INLINE static __inline

#elif defined (__GNUC__)
#define __ASM __asm
/*!< asm keyword for GNU Compiler */
#define __INLINE __inline
/*!< inline keyword for GNU Compiler */
#define __STATIC_INLINE static inline

#elif defined (__ICCARM__)
#define __ASM __asm
/*!< asm keyword for IAR Compiler */
#define __INLINE __inline
/*!< inline keyword for IAR Compiler. Only available in High optimization
mode! */
#define __STATIC_INLINE static inline

#elif defined (__TMS470__)
#define __ASM __asm
/*!< asm keyword for TI CCS Compiler */
#define __STATIC_INLINE static inline

#elif defined (__TASKING__)
#define __ASM __asm
/*!< asm keyword for TASKING Compiler */
#define __INLINE __inline
/*!< inline keyword for TASKING Compiler */
#define __STATIC_INLINE static inline

#elif defined (__CSMC__)
#define __packed
#define __ASM __asm
/*!< asm keyword for COSMIC Compiler */
#define __INLINE __inline
/*use -pc99 on compile line !< inline keyword for COSMIC Compiler */
#define __STATIC_INLINE static inline
/*!<

```

```

#endif

/** __FPU_USED indicates whether an FPU is used or not.
   For this, __FPU_PRESENT has to be checked prior to making use of FPU
   specific registers and functions.
*/
#if defined ( __CC_ARM )
  #if defined __TARGET_FPU_VFP
    #if (__FPU_PRESENT == 1)
      #define __FPU_USED          1
    #else
      #warning "Compiler generates FPU instructions for a device without
an FPU (check __FPU_PRESENT)"
      #define __FPU_USED          0
    #endif
  #else
    #define __FPU_USED          0
  #endif

#elif defined ( __GNUC__ )
  #if defined ( __VFP_FP__ ) && !defined( __SOFTFP__ )
    #if (__FPU_PRESENT == 1)
      #define __FPU_USED          1
    #else
      #warning "Compiler generates FPU instructions for a device without
an FPU (check __FPU_PRESENT)"
      #define __FPU_USED          0
    #endif
  #else
    #define __FPU_USED          0
  #endif

#elif defined ( __ICCARM__ )
  #if defined __ARMVFP__
    #if (__FPU_PRESENT == 1)
      #define __FPU_USED          1
    #else
      #warning "Compiler generates FPU instructions for a device without
an FPU (check __FPU_PRESENT)"
      #define __FPU_USED          0
    #endif
  #else
    #define __FPU_USED          0
  #endif

#elif defined ( __TMS470__ )
  #if defined __TI_VFP_SUPPORT__
    #if (__FPU_PRESENT == 1)
      #define __FPU_USED          1
    #else
      #warning "Compiler generates FPU instructions for a device without
an FPU (check __FPU_PRESENT)"
      #define __FPU_USED          0
    #endif
  #endif

```

```

#else
    #define __FPU_USED          0
#endif

#elif defined ( __TASKING__ )
    #if defined __FPU_VFP
        #if ( __FPU_PRESENT == 1)
            #define __FPU_USED      1
        #else
            #error "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
            #define __FPU_USED      0
        #endif
    #else
        #define __FPU_USED      0
    #endif
#endif

#elif defined ( __CSMC__ )           /* Cosmic */
    #if ( __CSMC__ & 0x400)          // FPU present for parser
        #if ( __FPU_PRESENT == 1)
            #define __FPU_USED      1
        #else
            #error "Compiler generates FPU instructions for a device without an
FPU (check __FPU_PRESENT)"
            #define __FPU_USED      0
        #endif
    #else
        #define __FPU_USED      0
    #endif
#endif

#endif

#include <stdint.h>                  /* standard types definitions
*/
#include <core_cmInstr.h>              /* Core Instruction Access
*/
#include <core_cmFunc.h>                /* Core Function Access
*/
#include <core_cmSimd.h>                /* Compiler specific SIMD
Intrinsics
*/
#endif

#ifndef __cplusplus
#endif

#endif /* __CORE_CM4_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM4_H_DEPENDANT
#define __CORE_CM4_H_DEPENDANT

#ifndef __cplusplus
    extern "C" {
#endif
#endif

```

```

/* check device defines and use defaults */
#if defined __CHECK_DEVICE_DEFINES
    #ifndef __CM4_REV
        #define __CM4_REV          0x0000
        #warning "__CM4_REV not defined in device header file; using
default!"
    #endif

    #ifndef __FPU_PRESENT
        #define __FPU_PRESENT      0
        #warning "__FPU_PRESENT not defined in device header file; using
default!"
    #endif

    #ifndef __MPU_PRESENT
        #define __MPU_PRESENT      0
        #warning "__MPU_PRESENT not defined in device header file; using
default!"
    #endif

    #ifndef __NVIC_PRIO_BITS
        #define __NVIC_PRIO_BITS    4
        #warning "__NVIC_PRIO_BITS not defined in device header file; using
default!"
    #endif

    #ifndef __Vendor_SysTickConfig
        #define __Vendor_SysTickConfig 0
        #warning "__Vendor_SysTickConfig not defined in device header file;
using default!"
    #endif
#endif

/* IO definitions (access restrictions to peripheral registers) */
/***
    \defgroup CMSIS_glob_defs CMSIS Global Defines

    <strong>IO Type Qualifiers</strong> are used
    \li to specify the access to peripheral variables.
    \li for automatic generation of peripheral register debug
information.
*/
#ifndef __cplusplus
    #define __I     volatile           /*!< Defines 'read only'
permissions                         */
#else
    #define __I     volatile const    /*!< Defines 'read only'
permissions                         */
#endif
#define __O     volatile           /*!< Defines 'write only'
permissions                         */
#define __IO    volatile           /*!< Defines 'read / write'
permissions                         */

```

```

/*@} end of group Cortex_M4 */

*****  

*          Register Abstraction  

Core Register contain:  

- Core Register  

- Core NVIC Register  

- Core SCB Register  

- Core SysTick Register  

- Core Debug Register  

- Core MPU Register  

- Core FPU Register

*****/  

/** \defgroup CMSIS_core_register Defines and Type Definitions
    \brief Type definitions and defines for Cortex-M processor based
devices.  

*/  

/** \ingroup CMSIS_core_register
\defgroup CMSIS_CORE Status and Control Registers
\brief Core Register type definitions.  

{@  

*/  

/** \brief Union type to access the Application Program Status Register
(APSR) .  

*/
typedef union
{
    struct
    {
        uint32_t _reserved0:16;           /*!< bit: 0..15 Reserved
*/
        uint32_t GE:4;                  /*!< bit: 16..19 Greater than
or Equal flags           */
        uint32_t _reserved1:7;           /*!< bit: 20..26 Reserved
*/
        uint32_t Q:1;                  /*!< bit: 27 Saturation
condition flag           */
        uint32_t V:1;                  /*!< bit: 28 Overflow
condition code flag      */
        uint32_t C:1;                  /*!< bit: 29 Carry
condition code flag      */
        uint32_t Z:1;                  /*!< bit: 30 Zero condition
code flag                */
        uint32_t N:1;                  /*!< bit: 31 Negative
condition code flag      */
    }
}

```

```

    } b;
    access          /* !< Structure used for bit
    uint32_t w;      */
    access          /* !< Type       used for word
} APSR_Type;

/* APSR Register Definitions */
#define APSR_N_Pos           31
/*!< APSR: N Position */
#define APSR_N_Msk          (1UL << APSR_N_Pos)
/*!< APSR: N Mask */

#define APSR_Z_Pos           30
/*!< APSR: Z Position */
#define APSR_Z_Msk          (1UL << APSR_Z_Pos)
/*!< APSR: Z Mask */

#define APSR_C_Pos           29
/*!< APSR: C Position */
#define APSR_C_Msk          (1UL << APSR_C_Pos)
/*!< APSR: C Mask */

#define APSR_V_Pos           28
/*!< APSR: V Position */
#define APSR_V_Msk          (1UL << APSR_V_Pos)
/*!< APSR: V Mask */

#define APSR_Q_Pos           27
/*!< APSR: Q Position */
#define APSR_Q_Msk          (1UL << APSR_Q_Pos)
/*!< APSR: Q Mask */

#define APSR_GE_Pos          16
/*!< APSR: GE Position */
#define APSR_GE_Msk          (0xFUL << APSR_GE_Pos)
/*!< APSR: GE Mask */

/** \brief Union type to access the Interrupt Program Status Register
(IPSRR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;           /*!< bit: 0.. 8 Exception
number                    */
        uint32_t _reserved0:23;   /*!< bit: 9..31 Reserved
*/                                */
        } b;
    access          /* !< Structure used for bit
    uint32_t w;      */
    access          /* !< Type       used for word
} IPSR_Type;

```

```

/* IPSR Register Definitions */
#define IPSR_ISR_Pos 0
/*!< IPSR: ISR Position */
#define IPSR_ISR_Msk (0x1FFUL /*<< IPSR_ISR_Pos*/)
/*!< IPSR: ISR Mask */

/** \brief Union type to access the Special-Purpose Program Status
Registers (xPSR).
*/
typedef union
{
    struct
    {
        uint32_t ISR:9; /*!< bit: 0.. 8 Exception
number */                                /*!< bit: 9..15 Reserved
*/                                         /*!< bit: 16..19 Greater than
or Equal flags */                         /*!< bit: 20..23 Reserved
*/                                         /*!< bit: 24 Thumb bit
        uint32_t GE:4; */                      /*!< bit: 25..26 saved IT state
        uint32_t _reserved0:7; */                /*!< bit: 27 Saturation
        uint32_t _reserved1:4; */                /*!< bit: 28 Overflow
*/                                         /*!< bit: 29 Carry
        uint32_t T:1; */                      /*!< bit: 30 Zero condition
(read 0) */ */                           /*!< bit: 31 Negative
        uint32_t IT:2; */                      /*!< bit: 31 Structure used for bit
(read 0) */ */                           /*!< Type used for word
        uint32_t Q:1; */                      /*!< bit: 31
condition flag */ */                     /*!< bit: 31
        uint32_t V:1; */                      /*!< bit: 31
condition code flag */ */                 /*!< bit: 31
        uint32_t C:1; */                      /*!< bit: 31
condition code flag */ */                 /*!< bit: 31
        uint32_t Z:1; */                      /*!< bit: 31
code flag */ */                           /*!< bit: 31
        uint32_t N:1; */                      /*!< bit: 31
condition code flag */ */                 /*!< bit: 31
        } b;
        access */                            /*!< bit: 31
        uint32_t w; */                      /*!< bit: 31
        access */ */                         /*!< bit: 31
    } xPSR_Type;

/* xPSR Register Definitions */
#define xPSR_N_Pos 31
/*!< xPSR: N Position */
#define xPSR_N_Msk (1UL << xPSR_N_Pos)
/*!< xPSR: N Mask */

#define xPSR_Z_Pos 30
/*!< xPSR: Z Position */
#define xPSR_Z_Msk (1UL << xPSR_Z_Pos)
/*!< xPSR: Z Mask */

```

```

#define xPSR_C_Pos          29
/*!< xPSR: C Position */
#define xPSR_C_Msk          (1UL << xPSR_C_Pos)
/*!< xPSR: C Mask */

#define xPSR_V_Pos          28
/*!< xPSR: V Position */
#define xPSR_V_Msk          (1UL << xPSR_V_Pos)
/*!< xPSR: V Mask */

#define xPSR_Q_Pos          27
/*!< xPSR: Q Position */
#define xPSR_Q_Msk          (1UL << xPSR_Q_Pos)
/*!< xPSR: Q Mask */

#define xPSR_IT_Pos          25
/*!< xPSR: IT Position */
#define xPSR_IT_Msk          (3UL << xPSR_IT_Pos)
/*!< xPSR: IT Mask */

#define xPSR_T_Pos          24
/*!< xPSR: T Position */
#define xPSR_T_Msk          (1UL << xPSR_T_Pos)
/*!< xPSR: T Mask */

#define xPSR_GE_Pos          16
/*!< xPSR: GE Position */
#define xPSR_GE_Msk          (0xFUL << xPSR_GE_Pos)
/*!< xPSR: GE Mask */

#define xPSR_ISR_Pos          0
/*!< xPSR: ISR Position */
#define xPSR_ISR_Msk          (0x1FFUL /*<< xPSR_ISR_Pos*/)
/*!< xPSR: ISR Mask */

/** \brief Union type to access the Control Registers (CONTROL).
 */
typedef union
{
    struct
    {
        uint32_t nPRIV:1;                                /*!< bit:      0 Execution
        privilege in Thread mode */
        uint32_t SPSEL:1;                                /*!< bit:      1 Stack to be
        used           */
        uint32_t FPCA:1;                                /*!< bit:      2 FP extension
        active flag   */
        uint32_t _reserved0:29;                            /*!< bit:  3..31 Reserved
    */
        } b;
    access
    uint32_t w;                                      /*!< Structure used for bit
    access           */
    /*!< Type       used for word
    access           */
}

```

```

} CONTROL_Type;

/* CONTROL Register Definitions */
#define CONTROL_FPCA_Pos           2
/*!< CONTROL: FPCA Position */
#define CONTROL_FPCA_Msk          (1UL << CONTROL_FPCA_Pos)

#define CONTROL_SPSEL_Pos          1
/*!< CONTROL: SPSEL Position */
#define CONTROL_SPSEL_Msk         (1UL << CONTROL_SPSEL_Pos)

#define CONTROL_nPRIV_Pos          0
/*!< CONTROL: nPRIV Position */
#define CONTROL_nPRIV_Msk        (1UL /*<< CONTROL_nPRIV_Pos*/)

/*@} end of group CMSIS_CORE */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_NVIC Nested Vectored Interrupt Controller (NVIC)
 \brief Type definitions for the NVIC Registers
 @{
 */

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
 */
typedef struct
{
    __IO uint32_t ISER[8];                      /*!< Offset: 0x000 (R/W) */
    Interrupt Set Enable Register
    uint32_t RESERVED0[24];

    __IO uint32_t ICER[8];                      /*!< Offset: 0x080 (R/W) */
    Interrupt Clear Enable Register
    uint32_t RSERVED1[24];

    __IO uint32_t ISPR[8];                      /*!< Offset: 0x100 (R/W) */
    Interrupt Set Pending Register
    uint32_t RESERVED2[24];

    __IO uint32_t ICPR[8];                      /*!< Offset: 0x180 (R/W) */
    Interrupt Clear Pending Register
    uint32_t RESERVED3[24];

    __IO uint32_t IABR[8];                      /*!< Offset: 0x200 (R/W) */
    Interrupt Active bit Register
    uint32_t RESERVED4[56];

    __IO uint8_t  IP[240];                      /*!< Offset: 0x300 (R/W) */
    Interrupt Priority Register (8Bit wide)
    uint32_t RESERVED5[644];

    __O  uint32_t STIR;                         /*!< Offset: 0xE00 ( /W) */
    Software Trigger Interrupt Register
} NVIC_Type;

```

```

/* Software Triggered Interrupt Register Definitions */
#define NVIC_STIR_INTID_Pos 0
/*!< STIR: INTLINESNUM Position */
#define NVIC_STIR_INTID_Msk (0x1FFUL /*<<
NVIC_STIR_INTID_Pos*/) /*!< STIR: INTLINESNUM Mask */

/*@} end of group CMSIS_NVIC */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB System Control Block (SCB)
\brief Type definitions for the System Control Block Registers
{@
*/

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct
{
    __I uint32_t CPUID; /*!< Offset: 0x000 (R/ ) CPUID
Base Register */
    __IO uint32_t ICSR; /*!< Offset: 0x004 (R/W) */
    /* Interrupt Control and State Register */
    __IO uint32_t VTOR; /*!< Offset: 0x008 (R/W) */
    /* Table Offset Register */
    __IO uint32_t AIRCR; /*!< Offset: 0x00C (R/W) */
    /* Application Interrupt and Reset Control Register */
    __IO uint32_t SCR; /*!< Offset: 0x010 (R/W) */
    /* System Control Register */
    __IO uint32_t CCR; /*!< Offset: 0x014 (R/W) */
    /* Configuration Control Register */
    __IO uint8_t SHP[12]; /*!< Offset: 0x018 (R/W) */
    /* Handlers Priority Registers (4-7, 8-11, 12-15) */
    __IO uint32_t SHCSR; /*!< Offset: 0x024 (R/W) */
    /* Handler Control and State Register */
    __IO uint32_t CFSR; /*!< Offset: 0x028 (R/W) */
    /* Configurable Fault Status Register */
    __IO uint32_t HFSR; /*!< Offset: 0x02C (R/W) */
    /* HardFault Status Register */
    __IO uint32_t DFSR; /*!< Offset: 0x030 (R/W) */
    /* Fault Status Register */
    __IO uint32_t MMFAR; /*!< Offset: 0x034 (R/W) */
    /* MemManage Fault Address Register */
    __IO uint32_t BFAR; /*!< Offset: 0x038 (R/W) */
    /* BusFault Address Register */
    __IO uint32_t AFSR; /*!< Offset: 0x03C (R/W) */
    /* Auxiliary Fault Status Register */
    __I uint32_t PFR[2]; /*!< Offset: 0x040 (R/ ) */
    /* Processor Feature Register */
    __I uint32_t DFR; /*!< Offset: 0x048 (R/ ) */
    /* Feature Register */
    __I uint32_t ADR; /*!< Offset: 0x04C (R/ ) */
    /* Auxiliary Feature Register */
}

```

```

    __I  uint32_t MMFR[4];          /*!< Offset: 0x050 (R/ )  Memory
Model Feature Register           */
    __I  uint32_t ISAR[5];          /*!< Offset: 0x060 (R/ )  */
Instruction Set Attributes Register   */
        uint32_t RESERVED0[5];      /*/ */
    __IO uint32_t CPACR;           /*!< Offset: 0x088 (R/W) */
Coprocessor Access Control Register  */
} SCB_Type;

/* SCB CPUID Register Definitions */
#define SCB_CPUID_IMPLEMENTER_Pos      24
/*!< SCB CPUID: IMPLEMENTER Position */
#define SCB_CPUID_IMPLEMENTER_Msk      (0xFFUL <<
SCB_CPUID_IMPLEMENTER_Pos)           /*!< SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos         20
/*!< SCB CPUID: VARIANT Position */
#define SCB_CPUID_VARIANT_Msk         (0xFUL <<
SCB_CPUID_VARIANT_Pos)             /*!< SCB CPUID: VARIANT Mask */

#define SCB_CPUID_ARCHITECTURE_Pos     16
/*!< SCB CPUID: ARCHITECTURE Position */
#define SCB_CPUID_ARCHITECTURE_Msk     (0xFUL <<
SCB_CPUID_ARCHITECTURE_Pos)         /*!< SCB CPUID: ARCHITECTURE Mask */

#define SCB_CPUID_PARTNO_Pos          4
/*!< SCB CPUID: PARTNO Position */
#define SCB_CPUID_PARTNO_Msk          (0xFFFFUL <<
SCB_CPUID_PARTNO_Pos)              /*!< SCB CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos         0
/*!< SCB CPUID: REVISION Position */
#define SCB_CPUID_REVISION_Msk         (0xFUL /*<<
SCB_CPUID_REVISION_Pos*/)          /*!< SCB CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */
#define SCB_ICSR_NMIPENDSET_Pos       31
/*!< SCB ICSR: NMIPENDSET Position */
#define SCB_ICSR_NMIPENDSET_Msk       (1UL <<
SCB_ICSR_NMIPENDSET_Pos)           /*!< SCB ICSR: NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos        28
/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk        (1UL <<
SCB_ICSR_PENDSVSET_Pos)           /*!< SCB ICSR: PENDSVSET Mask */

#define SCB_ICSR_PENDSVCLR_Pos        27
/*!< SCB ICSR: PENDSVCLR Position */
#define SCB_ICSR_PENDSVCLR_Msk        (1UL <<
SCB_ICSR_PENDSVCLR_Pos)           /*!< SCB ICSR: PENDSVCLR Mask */

#define SCB_ICSR_PENDSTSET_Pos        26
/*!< SCB ICSR: PENDSTSET Position */

```

```

#define SCB_ICSR_PENDSTSET_Msk          (1UL <<
SCB_ICSR_PENDSTSET_Pos)             /*!< SCB ICSR: PENDSTSET Mask */

#define SCB_ICSR_PENDSTCLR_Pos          25
/*!< SCB ICSR: PENDSTCLR Position */
#define SCB_ICSR_PENDSTCLR_Msk          (1UL <<
SCB_ICSR_PENDSTCLR_Pos)             /*!< SCB ICSR: PENDSTCLR Mask */

#define SCB_ICSR_ISRPREEMPT_Pos         23
/*!< SCB ICSR: ISRPREEMPT Position */
#define SCB_ICSR_ISRPREEMPT_Msk          (1UL <<
SCB_ICSR_ISRPREEMPT_Pos)             /*!< SCB ICSR: ISRPREEMPT Mask */

#define SCB_ICSR_ISRPENDING_Pos          22
/*!< SCB ICSR: ISRPENDING Position */
#define SCB_ICSR_ISRPENDING_Msk          (1UL <<
SCB_ICSR_ISRPENDING_Pos)             /*!< SCB ICSR: ISRPENDING Mask */

#define SCB_ICSR_VECTPENDING_Pos         12
/*!< SCB ICSR: VECTPENDING Position */
#define SCB_ICSR_VECTPENDING_Msk          (0x1FFUL <<
SCB_ICSR_VECTPENDING_Pos)             /*!< SCB ICSR: VECTPENDING Mask */

#define SCB_ICSR_RETTOBASE_Pos           11
/*!< SCB ICSR: RETTOBASE Position */
#define SCB_ICSR_RETTOBASE_Msk          (1UL <<
SCB_ICSR_RETTOBASE_Pos)             /*!< SCB ICSR: RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos          0
/*!< SCB ICSR: VECTACTIVE Position */
#define SCB_ICSR_VECTACTIVE_Msk          (0x1FFFUL /*<<
SCB_ICSR_VECTACTIVE_Pos*/)           /*!< SCB ICSR: VECTACTIVE Mask */

/* SCB Vector Table Offset Register Definitions */
#define SCB_VTOR_TBLOFF_Pos              7
/*!< SCB VTOR: TBLOFF Position */
#define SCB_VTOR_TBLOFF_Msk              (0x1FFFFFFUL <<
SCB_VTOR_TBLOFF_Pos)                /*!< SCB VTOR: TBLOFF Mask */

/* SCB Application Interrupt and Reset Control Register Definitions */
#define SCB_AIRCR_VECTKEY_Pos            16
/*!< SCB AIRCR: VECTKEY Position */
#define SCB_AIRCR_VECTKEY_Msk             (0xFFFFUL <<
SCB_AIRCR_VECTKEY_Pos)               /*!< SCB AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos        16
/*!< SCB AIRCR: VECTKEYSTAT Position */
#define SCB_AIRCR_VECTKEYSTAT_Msk          (0xFFFFUL <<
SCB_AIRCR_VECTKEYSTAT_Pos)            /*!< SCB AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos          15
/*!< SCB AIRCR: ENDIANESS Position */
#define SCB_AIRCR_ENDIANESS_Msk             (1UL <<
SCB_AIRCR_ENDIANESS_Pos)               /*!< SCB AIRCR: ENDIANESS Mask */

```

```

#define SCB_AIRCR_PRIGROUP_Pos           8
/*!< SCB AIRCR: PRIGROUP Position */
#define SCB_AIRCR_PRIGROUP_Msk          (7UL <<
SCB_AIRCR_PRIGROUP_Pos)             /*!< SCB AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos       2
/*!< SCB AIRCR: SYSRESETREQ Position */
#define SCB_AIRCR_SYSRESETREQ_Msk      (1UL <<
SCB_AIRCR_SYSRESETREQ_Pos)         /*!< SCB AIRCR: SYSRESETREQ Mask
*/

#define SCB_AIRCR_VECTCLRACTIVE_Pos     1
/*!< SCB AIRCR: VECTCLRACTIVE Position */
#define SCB_AIRCR_VECTCLRACTIVE_Msk    (1UL <<
SCB_AIRCR_VECTCLRACTIVE_Pos)       /*!< SCB AIRCR: VECTCLRACTIVE Mask
*/

#define SCB_AIRCR_VECTRESET_Pos        0
/*!< SCB AIRCR: VECTRESET Position */
#define SCB_AIRCR_VECTRESET_Msk        (1UL /*<<
SCB_AIRCR_VECTRESET_Pos*/)        /*!< SCB AIRCR: VECTRESET Mask */

/* SCB System Control Register Definitions */
#define SCB_SCR_SEVONPEND_Pos          4
/*!< SCB SCR: SEVONPEND Position */
#define SCB_SCR_SEVONPEND_Msk         (1UL << SCB_SCR_SEVONPEND_Pos)

#define SCB_SCR_SLEEPDEEP_Pos          2
/*!< SCB SCR: SLEEPDEEP Position */
#define SCB_SCR_SLEEPDEEP_Msk         (1UL << SCB_SCR_SLEEPDEEP_Pos)

#define SCB_SCR_SLEEPONEXIT_Pos        1
/*!< SCB SCR: SLEEPONEXIT Position */
#define SCB_SCR_SLEEPONEXIT_Msk       (1UL <<
SCB_SCR_SLEEPONEXIT_Pos)          /*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */
#define SCB_CCR_STKALIGN_Pos           9
/*!< SCB CCR: STKALIGN Position */
#define SCB_CCR_STKALIGN_Msk          (1UL << SCB_CCR_STKALIGN_Pos)

#define SCB_CCR_BFHFNIGN_Pos           8
/*!< SCB CCR: BFHFNMIGN Position */
#define SCB_CCR_BFHFNIGN_Msk          (1UL << SCB_CCR_BFHFNIGN_Pos)

#define SCB_CCR_DIV_0_TRP_Pos          4
/*!< SCB CCR: DIV_0_TRP Position */
#define SCB_CCR_DIV_0_TRP_Msk         (1UL << SCB_CCR_DIV_0_TRP_Pos)

```

```

#define SCB_CCR_UNALIGN_TRP_Pos 3
/*!< SCB CCR: UNALIGN_TRP Position */
#define SCB_CCR_UNALIGN_TRP_Msk (1UL <<
SCB_CCR_UNALIGN_TRP_Pos) /*!< SCB CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETMPEND_Pos 1
/*!< SCB CCR: USERSETMPEND Position */
#define SCB_CCR_USERSETMPEND_Msk (1UL <<
SCB_CCR_USERSETMPEND_Pos) /*!< SCB CCR: USERSETMPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos 0
/*!< SCB CCR: NONBASETHRDENA Position */
#define SCB_CCR_NONBASETHRDENA_Msk (1UL /*<<
SCB_CCR_NONBASETHRDENA_Pos) /*!< SCB CCR: NONBASETHRDENA Mask */

/* SCB System Handler Control and State Register Definitions */
#define SCB_SHCSR_USGFAULTENA_Pos 18
/*!< SCB SHCSR: USGFAULTENA Position */
#define SCB_SHCSR_USGFAULTENA_Msk (1UL <<
SCB_SHCSR_USGFAULTENA_Pos) /*!< SCB SHCSR: USGFAULTENA Mask */
*/

#define SCB_SHCSR_BUSFAULTENA_Pos 17
/*!< SCB SHCSR: BUSFAULTENA Position */
#define SCB_SHCSR_BUSFAULTENA_Msk (1UL <<
SCB_SHCSR_BUSFAULTENA_Pos) /*!< SCB SHCSR: BUSFAULTENA Mask */
*/

#define SCB_SHCSR_MEMFAULTENA_Pos 16
/*!< SCB SHCSR: MEMFAULTENA Position */
#define SCB_SHCSR_MEMFAULTENA_Msk (1UL <<
SCB_SHCSR_MEMFAULTENA_Pos) /*!< SCB SHCSR: MEMFAULTENA Mask */
*/

#define SCB_SHCSR_SVCALLPENDED_Pos 15
/*!< SCB SHCSR: SVCALLPENDED Position */
#define SCB_SHCSR_SVCALLPENDED_Msk (1UL <<
SCB_SHCSR_SVCALLPENDED_Pos) /*!< SCB SHCSR: SVCALLPENDED Mask */
*/

#define SCB_SHCSR_BUSFAULTPENDED_Pos 14
/*!< SCB SHCSR: BUSFAULTPENDED Position */
#define SCB_SHCSR_BUSFAULTPENDED_Msk (1UL <<
SCB_SHCSR_BUSFAULTPENDED_Pos) /*!< SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos 13
/*!< SCB SHCSR: MEMFAULTPENDED Position */
#define SCB_SHCSR_MEMFAULTPENDED_Msk (1UL <<
SCB_SHCSR_MEMFAULTPENDED_Pos) /*!< SCB SHCSR: MEMFAULTPENDED Mask */

```

```

#define SCB_SHCSR_USGFAULTPENDED_Pos 12
/*!< SCB SHCSR: USGFAULTPENDED Position */
#define SCB_SHCSR_USGFAULTPENDED_Msk (1UL <<
SCB_SHCSR_USGFAULTPENDED_Pos) /*!< SCB SHCSR: USGFAULTPENDED
Mask */

#define SCB_SHCSR_SYSTICKACT_Pos 11
/*!< SCB SHCSR: SYSTICKACT Position */
#define SCB_SHCSR_SYSTICKACT_Msk (1UL <<
SCB_SHCSR_SYSTICKACT_Pos) /*!< SCB SHCSR: SYSTICKACT Mask */

#define SCB_SHCSR_PENDSVACT_Pos 10
/*!< SCB SHCSR: PENDSVACT Position */
#define SCB_SHCSR_PENDSVACT_Msk (1UL <<
SCB_SHCSR_PENDSVACT_Pos) /*!< SCB SHCSR: PENDSVACT Mask */

#define SCB_SHCSR_MONITORACT_Pos 8
/*!< SCB SHCSR: MONITORACT Position */
#define SCB_SHCSR_MONITORACT_Msk (1UL <<
SCB_SHCSR_MONITORACT_Pos) /*!< SCB SHCSR: MONITORACT Mask */

#define SCB_SHCSR_SVCALLACT_Pos 7
/*!< SCB SHCSR: SVCALLACT Position */
#define SCB_SHCSR_SVCALLACT_Msk (1UL <<
SCB_SHCSR_SVCALLACT_Pos) /*!< SCB SHCSR: SVCALLACT Mask */

#define SCB_SHCSR_USGFAULTACT_Pos 3
/*!< SCB SHCSR: USGFAULTACT Position */
#define SCB_SHCSR_USGFAULTACT_Msk (1UL <<
SCB_SHCSR_USGFAULTACT_Pos) /*!< SCB SHCSR: USGFAULTACT Mask
*/

#define SCB_SHCSR_BUSFAULTACT_Pos 1
/*!< SCB SHCSR: BUSFAULTACT Position */
#define SCB_SHCSR_BUSFAULTACT_Msk (1UL <<
SCB_SHCSR_BUSFAULTACT_Pos) /*!< SCB SHCSR: BUSFAULTACT Mask
*/

#define SCB_SHCSR_MEMFAULTACT_Pos 0
/*!< SCB SHCSR: MEMFAULTACT Position */
#define SCB_SHCSR_MEMFAULTACT_Msk (1UL /*<<
SCB_SHCSR_MEMFAULTACT_Pos*/) /*!< SCB SHCSR: MEMFAULTACT Mask */

/* SCB Configurable Fault Status Registers Definitions */
#define SCB_CFSR_USGFAULTSR_Pos 16
/*!< SCB CFSR: Usage Fault Status Register Position */
#define SCB_CFSR_USGFAULTSR_Msk (0xFFFFUL <<
SCB_CFSR_USGFAULTSR_Pos) /*!< SCB CFSR: Usage Fault Status
Register Mask */

#define SCB_CFSR_BUSFAULTSR_Pos 8
/*!< SCB CFSR: Bus Fault Status Register Position */

```

```

#define SCB_CFSR_BUSFAULTSR_Msk          (0xFFUL <<
SCB_CFSR_BUSFAULTSR_Pos)             /*!< SCB CFSR: Bus Fault Status
Register Mask */

#define SCB_CFSR_MEMFAULTSR_Pos          0
/*!< SCB CFSR: Memory Manage Fault Status Register Position */
#define SCB_CFSR_MEMFAULTSR_Msk          (0xFFUL /*<<
SCB_CFSR_MEMFAULTSR_Pos*/)           /*!< SCB CFSR: Memory Manage Fault
Status Register Mask */

/* SCB Hard Fault Status Registers Definitions */
#define SCB_HFSR_DEBUGEVT_Pos            31
/*!< SCB HFSR: DEBUGEVT Position */
#define SCB_HFSR_DEBUGEVT_Msk            (1UL << SCB_HFSR_DEBUGEVT_Pos)

#define SCB_HFSR_FORCED_Pos              30
/*!< SCB HFSR: FORCED Position */
#define SCB_HFSR_FORCED_Msk              (1UL << SCB_HFSR_FORCED_Pos)

#define SCB_HFSR_VECTTBL_Pos             1
/*!< SCB HFSR: VECTTBL Position */
#define SCB_HFSR_VECTTBL_Msk              (1UL << SCB_HFSR_VECTTBL_Pos)

/* SCB Debug Fault Status Register Definitions */
#define SCB_DFSR_EXTERNAL_Pos            4
/*!< SCB DFSR: EXTERNAL Position */
#define SCB_DFSR_EXTERNAL_Msk            (1UL << SCB_DFSR_EXTERNAL_Pos)

#define SCB_DFSR_VCATCH_Pos              3
/*!< SCB DFSR: VCATCH Position */
#define SCB_DFSR_VCATCH_Msk              (1UL << SCB_DFSR_VCATCH_Pos)

#define SCB_DFSR_DWTTRAP_Pos             2
/*!< SCB DFSR: DWTTRAP Position */
#define SCB_DFSR_DWTTRAP_Msk              (1UL << SCB_DFSR_DWTTRAP_Pos)

#define SCB_DFSR_BKPT_Pos                1
/*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk                (1UL << SCB_DFSR_BKPT_Pos)

#define SCB_DFSR_HALTED_Pos               0
/*!< SCB DFSR: HALTED Position */
#define SCB_DFSR_HALTED_Msk              (1UL /*<<
SCB_DFSR_HALTED_Pos*/)           /*!< SCB DFSR: HALTED Mask */

/*@} end of group CMSIS_SCB */

```

```

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_SCnSCB System Controls not in SCB (SCnSCB)
 \brief Type definitions for the System Control and ID Register
 not in the SCB
 @{
 */

/** \brief Structure type to access the System Control and ID Register
not in the SCB.
 */
typedef struct
{
    uint32_t RESERVED0[1];
    __I uint32_t ICTR;                      /*!< Offset: 0x004 (R/ ) */
    Interrupt Controller Type Register        */
    __IO uint32_t ACTLR;                     /*!< Offset: 0x008 (R/W) */
    Auxiliary Control Register                */
} SCnSCB_Type;

/* Interrupt Controller Type Register Definitions */
#define SCnSCB_ICTR_INTLINESNUM_Pos          0
/*!< ICTR: INTLINESNUM Position */
#define SCnSCB_ICTR_INTLINESNUM_Msk          (0xFUL /*<<
SCnSCB_ICTR_INTLINESNUM_Pos*/) /*!< ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */
#define SCnSCB_ACTLR_DISOOFP_Pos            9
/*!< ACTLR: DISOOFP Position */
#define SCnSCB_ACTLR_DISOOFP_Msk            (1UL <<
SCnSCB_ACTLR_DISOOFP_Pos) /*!< ACTLR: DISOOFP Mask */

#define SCnSCB_ACTLR_DISFPCA_Pos           8
/*!< ACTLR: DISFPCA Position */
#define SCnSCB_ACTLR_DISFPCA_Msk           (1UL <<
SCnSCB_ACTLR_DISFPCA_Pos) /*!< ACTLR: DISFPCA Mask */

#define SCnSCB_ACTLR_DISFOLD_Pos          2
/*!< ACTLR: DISFOLD Position */
#define SCnSCB_ACTLR_DISFOLD_Msk          (1UL <<
SCnSCB_ACTLR_DISFOLD_Pos) /*!< ACTLR: DISFOLD Mask */

#define SCnSCB_ACTLR_DISDEFWBUF_Pos       1
/*!< ACTLR: DISDEFWBUF Position */
#define SCnSCB_ACTLR_DISDEFWBUF_Msk       (1UL <<
SCnSCB_ACTLR_DISDEFWBUF_Pos) /*!< ACTLR: DISDEFWBUF Mask */

#define SCnSCB_ACTLR_DISMCYCINT_Pos      0
/*!< ACTLR: DISMCYCINT Position */
#define SCnSCB_ACTLR_DISMCYCINT_Msk      (1UL /*<<
SCnSCB_ACTLR_DISMCYCINT_Pos*/) /*!< ACTLR: DISMCYCINT Mask */

/*@} end of group CMSIS_SCnotSCB */

```

```

/** \ingroup CMSIS_core_register
 * \defgroup CMSIS_SysTick System Tick Timer (SysTick)
 * \brief Type definitions for the System Timer Registers.
 *
 */

/** \brief Structure type to access the System Timer (SysTick).
 */
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) */
    SysTick Control and Status Register */

    __IO uint32_t LOAD;          /*!< Offset: 0x004 (R/W) */
    SysTick Reload Value Register */

    __IO uint32_t VAL;           /*!< Offset: 0x008 (R/W) */
    SysTick Current Value Register */

    __I uint32_t CALIB;         /*!< Offset: 0x00C (R/ ) */
    SysTick Calibration Register */
} SysTick_Type;

/* SysTick Control / Status Register Definitions */
#define SysTick_CTRL_COUNTFLAG_Pos      16
/*!< SysTick CTRL: COUNTFLAG Position */
#define SysTick_CTRL_COUNTFLAG_Msk      (1UL <<
SysTick_CTRL_COUNTFLAG_Pos)        /*!< SysTick CTRL: COUNTFLAG Mask
 */

#define SysTick_CTRL_CLKSOURCE_Pos      2
/*!< SysTick CTRL: CLKSOURCE Position */
#define SysTick_CTRL_CLKSOURCE_Msk      (1UL <<
SysTick_CTRL_CLKSOURCE_Pos)        /*!< SysTick CTRL: CLKSOURCE Mask
 */

#define SysTick_CTRL_TICKINT_Pos       1
/*!< SysTick CTRL: TICKINT Position */
#define SysTick_CTRL_TICKINT_Msk       (1UL <<
SysTick_CTRL_TICKINT_Pos)         /*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos        0
/*!< SysTick CTRL: ENABLE Position */
#define SysTick_CTRL_ENABLE_Msk        (1UL /*<<
SysTick_CTRL_ENABLE_Pos*/)        /*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */
#define SysTick_LOAD_RELOAD_Pos        0
/*!< SysTick LOAD: RELOAD Position */
#define SysTick_LOAD_RELOAD_Msk        (0xFFFFFFFFUL /*<<
SysTick_LOAD_RELOAD_Pos*/)        /*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */
#define SysTick_VAL_CURRENT_Pos        0
/*!< SysTick VAL: CURRENT Position */

```

```

#define SysTick_VAL_CURRENT_Msk          (0xFFFFFFFFUL /*<<
SysTick_VAL_CURRENT_Pos*/) /*!< SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */
#define SysTick_CALIB_NOREF_Pos         31
/*!< SysTick CALIB: NOREF Position */
#define SysTick_CALIB_NOREF_Msk         (1UL <<
SysTick_CALIB_NOREF_Pos) /*!< SysTick CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos          30
/*!< SysTick CALIB: SKEW Position */
#define SysTick_CALIB_SKEW_Msk          (1UL <<
SysTick_CALIB_SKEW_Pos) /*!< SysTick CALIB: SKEW Mask */

#define SysTick_CALIB_TENMS_Pos         0
/*!< SysTick CALIB: TENMS Position */
#define SysTick_CALIB_TENMS_Msk         (0xFFFFFFFFUL /*<<
SysTick_CALIB_TENMS_Pos*/) /*!< SysTick CALIB: TENMS Mask */

/*@} end of group CMSIS_SysTick */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_ITM      Instrumentation Trace Macrocell (ITM)
 \brief      Type definitions for the Instrumentation Trace Macrocell
 (ITM)
 @{
 */

/** \brief Structure type to access the Instrumentation Trace Macrocell
 Register (ITM).
 */
typedef struct
{
    __O union
    {
        __O uint8_t      u8;
        __O uint16_t     u16;
        __O uint32_t     u32;
        Stimulus Port 32-bit
        { PORT [32];
        Stimulus Port Registers
            uint32_t RESERVED0[864];
        __IO uint32_t TER;
        Trace Enable Register
            uint32_t RESERVED1[15];
        __IO uint32_t TPR;
        Trace Privilege Register
            uint32_t RESERVED2[15];
        __IO uint32_t TCR;
        Trace Control Register
            uint32_t RESERVED3[29];
        /*!< Offset: 0x000 ( /W) ITM
        */
        /*!< Offset: 0xE00 (R/W) ITM
        */
        /*!< Offset: 0xE40 (R/W) ITM
        */
        /*!< Offset: 0xE80 (R/W) ITM
        */
    }
}

```

```

    __O uint32_t IWR;
Integration Write Register
    __I uint32_t IRR;
Integration Read Register
    __IO uint32_t IMCR;
Integration Mode Control Register
        uint32_t RESERVED4[43];
    __O uint32_t LAR;
Lock Access Register
    __I uint32_t LSR;
Lock Status Register
        uint32_t RESERVED5[6];
    __I uint32_t PID4;
Peripheral Identification Register #4 */
    __I uint32_t PID5;
Peripheral Identification Register #5 */
    __I uint32_t PID6;
Peripheral Identification Register #6 */
    __I uint32_t PID7;
Peripheral Identification Register #7 */
    __I uint32_t PID0;
Peripheral Identification Register #0 */
    __I uint32_t PID1;
Peripheral Identification Register #1 */
    __I uint32_t PID2;
Peripheral Identification Register #2 */
    __I uint32_t PID3;
Peripheral Identification Register #3 */
    __I uint32_t CID0;
Component Identification Register #0 */
    __I uint32_t CID1;
Component Identification Register #1 */
    __I uint32_t CID2;
Component Identification Register #2 */
    __I uint32_t CID3;
Component Identification Register #3 */
} ITM_Type;

/* ITM Trace Privilege Register Definitions */
#define ITM_TPR_PRIVMASK_Pos          0
/*!< ITM TPR: PRIVMASK Position */
#define ITM_TPR_PRIVMASK_Msk          (0xFUL /*<<
ITM_TPR_PRIVMASK_Pos*/)           /*!< ITM TPR: PRIVMASK Mask */

/* ITM Trace Control Register Definitions */
#define ITM_TCR_BUSY_Pos             23
/*!< ITM TCR: BUSY Position */
#define ITM_TCR_BUSY_Msk              (1UL << ITM_TCR_BUSY_Pos)
/*!< ITM TCR: BUSY Mask */

#define ITM_TCR_TraceBusID_Pos       16
/*!< ITM TCR: ATBID Position */
#define ITM_TCR_TraceBusID_Msk         (0x7FUL <<
ITM_TCR_TraceBusID_Pos)           /*!< ITM TCR: ATBID Mask */

```

```

#define ITM_TCR_GTSFREQ_Pos          10
/*!< ITM TCR: Global timestamp frequency Position */
#define ITM_TCR_GTSFREQ_Msk          (3UL << ITM_TCR_GTSFREQ_Pos)
/*!< ITM TCR: Global timestamp frequency Mask */

#define ITM_TCR_TSPrescale_Pos       8
/*!< ITM TCR: TSPrescale Position */
#define ITM_TCR_TSPrescale_Msk       (3UL <<
ITM_TCR_TSPrescale_Pos)           /*!< ITM TCR: TSPrescale Mask */

#define ITM_TCR_SWOENA_Pos           4
/*!< ITM TCR: SWOENA Position */
#define ITM_TCR_SWOENA_Msk           (1UL << ITM_TCR_SWOENA_Pos)
/*!< ITM TCR: SWOENA Mask */

#define ITM_TCR_DWTENA_Pos           3
/*!< ITM TCR: DWTENA Position */
#define ITM_TCR_DWTENA_Msk           (1UL << ITM_TCR_DWTENA_Pos)
/*!< ITM TCR: DWTENA Mask */

#define ITM_TCR_SYNCENA_Pos          2
/*!< ITM TCR: SYNCENA Position */
#define ITM_TCR_SYNCENA_Msk           (1UL << ITM_TCR_SYNCENA_Pos)
/*!< ITM TCR: SYNCENA Mask */

#define ITM_TCR_TSENA_Pos             1
/*!< ITM TCR: TSENA Position */
#define ITM_TCR_TSENA_Msk             (1UL << ITM_TCR_TSENA_Pos)
/*!< ITM TCR: TSENA Mask */

#define ITM_TCR_ITMENA_Pos            0
/*!< ITM TCR: ITM Enable bit Position */
#define ITM_TCR_ITMENA_Msk             (1UL /*<<
ITM_TCR_ITMENA_Pos*/)           /*!< ITM TCR: ITM Enable bit Mask */

/* ITM Integration Write Register Definitions */
#define ITM_IWR_ATVALIDDM_Pos         0
/*!< ITM IWR: ATVALIDDM Position */
#define ITM_IWR_ATVALIDDM_Msk          (1UL /*<<
ITM_IWR_ATVALIDDM_Pos*/)        /*!< ITM IWR: ATVALIDDM Mask */

/* ITM Integration Read Register Definitions */
#define ITM_IRR_ATREADYM_Pos          0
/*!< ITM IRR: ATREADYM Position */
#define ITM_IRR_ATREADYM_Msk          (1UL /*<<
ITM_IRR_ATREADYM_Pos*/)        /*!< ITM IRR: ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */
#define ITM_IMCR_INTEGRATION_Pos       0
/*!< ITM IMCR: INTEGRATION Position */
#define ITM_IMCR_INTEGRATION_Msk       (1UL /*<<
ITM_IMCR_INTEGRATION_Pos*/)     /*!< ITM IMCR: INTEGRATION Mask */

```

```

/* ITM Lock Status Register Definitions */
#define ITM_LSR_ByteAcc_Pos 2
/*!< ITM LSR: ByteAcc Position */
#define ITM_LSR_ByteAcc_Msk (1UL << ITM_LSR_ByteAcc_Pos)
/*!< ITM LSR: ByteAcc Mask */

#define ITM_LSR_Access_Pos 1
/*!< ITM LSR: Access Position */
#define ITM_LSR_Access_Msk (1UL << ITM_LSR_Access_Pos)
/*!< ITM LSR: Access Mask */

#define ITM_LSR_Present_Pos 0
/*!< ITM LSR: Present Position */
#define ITM_LSR_Present_Msk (1UL /*<<
ITM_LSR_Present_Pos*/) /*!< ITM LSR: Present Mask */

/*@} */ /* end of group CMSIS_ITM */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_DWT Data Watchpoint and Trace (DWT)
\brief Type definitions for the Data Watchpoint and Trace (DWT)
{@{
*/
/** \brief Structure type to access the Data Watchpoint and Trace Register (DWT).
 */
typedef struct
{
    __IO uint32_t CTRL;           /*!< Offset: 0x000 (R/W) */
    Control Register             /* */
    __IO uint32_t CYCCNT;        /*!< Offset: 0x004 (R/W) Cycle */
    Count Register               /* */
    __IO uint32_t CPICNT;        /*!< Offset: 0x008 (R/W) CPI */
    Count Register               /* */
    __IO uint32_t EXCCNT;        /*!< Offset: 0x00C (R/W) */
    Exception Overhead Count Register /* */
    __IO uint32_t SLEEPCNT;      /*!< Offset: 0x010 (R/W) Sleep */
    Count Register               /* */
    __IO uint32_t LSUCNT;        /*!< Offset: 0x014 (R/W) LSU */
    Count Register               /* */
    __IO uint32_t FOLDCNT;       /*!< Offset: 0x018 (R/W) */
    Folded-instruction Count Register /* */
    __I  uint32_t PCSR;          /*!< Offset: 0x01C (R/ ) */
    Program Counter Sample Register /* */
    __IO uint32_t COMPO;         /*!< Offset: 0x020 (R/W) */
    Comparator Register 0       /* */
    __IO uint32_t MASK0;         /*!< Offset: 0x024 (R/W) Mask */
    Register 0                  /* */
    __IO uint32_t FUNCTION0;     /*!< Offset: 0x028 (R/W) */
    Function Register 0          /* */
    uint32_t RESERVED0[1];
}
*/
```

```

    __IO uint32_t COMP1;                      /* !< Offset: 0x030 (R/W)
    Comparator Register 1                     */
    __IO uint32_t MASK1;                      /* !< Offset: 0x034 (R/W) Mask
Register 1                                */
    __IO uint32_t FUNCTION1;                  /* !< Offset: 0x038 (R/W)
Function Register 1                         */
        uint32_t RESERVED1[1];                /* */
    __IO uint32_t COMP2;                      /* !< Offset: 0x040 (R/W)
Comparator Register 2                      */
    __IO uint32_t MASK2;                      /* !< Offset: 0x044 (R/W) Mask
Register 2                                */
    __IO uint32_t FUNCTION2;                  /* !< Offset: 0x048 (R/W)
Function Register 2                         */
        uint32_t RESERVED2[1];                /* */
    __IO uint32_t COMP3;                      /* !< Offset: 0x050 (R/W)
Comparator Register 3                      */
    __IO uint32_t MASK3;                      /* !< Offset: 0x054 (R/W) Mask
Register 3                                */
    __IO uint32_t FUNCTION3;                  /* !< Offset: 0x058 (R/W)
Function Register 3                         */
} DWT_Type;

/* DWT Control Register Definitions */
#define DWT_CTRL_NUMCOMP_Pos                 28
/*!< DWT CTRL: NUMCOMP Position */
#define DWT_CTRL_NUMCOMP_Msk                (0xFUL <<
DWT_CTRL_NUMCOMP_Pos)                  /*!< DWT CTRL: NUMCOMP Mask */

#define DWT_CTRL_NOTRCPKT_Pos               27
/*!< DWT CTRL: NOTRCPKT Position */
#define DWT_CTRL_NOTRCPKT_Msk              (0x1UL <<
DWT_CTRL_NOTRCPKT_Pos)                /*!< DWT CTRL: NOTRCPKT Mask */

#define DWT_CTRL_NOEXTTRIG_Pos              26
/*!< DWT CTRL: NOEXTTRIG Position */
#define DWT_CTRL_NOEXTTRIG_Msk             (0x1UL <<
DWT_CTRL_NOEXTTRIG_Pos)               /*!< DWT CTRL: NOEXTTRIG Mask */

#define DWT_CTRL_NOCYCCNT_Pos              25
/*!< DWT CTRL: NOCYCCNT Position */
#define DWT_CTRL_NOCYCCNT_Msk             (0x1UL <<
DWT_CTRL_NOCYCCNT_Pos)               /*!< DWT CTRL: NOCYCCNT Mask */

#define DWT_CTRL_NOPRFCNT_Pos              24
/*!< DWT CTRL: NOPRFCNT Position */
#define DWT_CTRL_NOPRFCNT_Msk             (0x1UL <<
DWT_CTRL_NOPRFCNT_Pos)               /*!< DWT CTRL: NOPRFCNT Mask */

#define DWT_CTRL_CYCEVTENA_Pos             22
/*!< DWT CTRL: CYCEVTENA Position */
#define DWT_CTRL_CYCEVTENA_Msk            (0x1UL <<
DWT_CTRL_CYCEVTENA_Pos)              /*!< DWT CTRL: CYCEVTENA Mask */

```

```

#define DWT_CTRL_FOLDEVTENA_Pos           21
/*!< DWT CTRL: FOLDEVTENA Position */
#define DWT_CTRL_FOLDEVTENA_Msk          (0x1UL <<
DWT_CTRL_FOLDEVTENA_Pos)             /*!< DWT CTRL: FOLDEVTENA Mask */

#define DWT_CTRL_LSUEVTENA_Pos           20
/*!< DWT CTRL: LSUEVTENA Position */
#define DWT_CTRL_LSUEVTENA_Msk          (0x1UL <<
DWT_CTRL_LSUEVTENA_Pos)             /*!< DWT CTRL: LSUEVTENA Mask */

#define DWT_CTRL_SLEEPEVTENA_Pos         19
/*!< DWT CTRL: SLEEPEVTENA Position */
#define DWT_CTRL_SLEEPEVTENA_Msk         (0x1UL <<
DWT_CTRL_SLEEPEVTENA_Pos)           /*!< DWT CTRL: SLEEPEVTENA Mask */

#define DWT_CTRL_EXCEVTENA_Pos           18
/*!< DWT CTRL: EXCEVTENA Position */
#define DWT_CTRL_EXCEVTENA_Msk          (0x1UL <<
DWT_CTRL_EXCEVTENA_Pos)             /*!< DWT CTRL: EXCEVTENA Mask */

#define DWT_CTRL_CPIEVTENA_Pos           17
/*!< DWT CTRL: CPIEVTENA Position */
#define DWT_CTRL_CPIEVTENA_Msk          (0x1UL <<
DWT_CTRL_CPIEVTENA_Pos)             /*!< DWT CTRL: CPIEVTENA Mask */

#define DWT_CTRL_EXCTRCENA_Pos           16
/*!< DWT CTRL: EXCTRCENA Position */
#define DWT_CTRL_EXCTRCENA_Msk          (0x1UL <<
DWT_CTRL_EXCTRCENA_Pos)             /*!< DWT CTRL: EXCTRCENA Mask */

#define DWT_CTRL_PCSAMPLENA_Pos           12
/*!< DWT CTRL: PCSAMPLENA Position */
#define DWT_CTRL_PCSAMPLENA_Msk          (0x1UL <<
DWT_CTRL_PCSAMPLENA_Pos)             /*!< DWT CTRL: PCSAMPLENA Mask */

#define DWT_CTRL_SYNCTAP_Pos              10
/*!< DWT CTRL: SYNCTAP Position */
#define DWT_CTRL_SYNCTAP_Msk             (0x3UL <<
DWT_CTRL_SYNCTAP_Pos)               /*!< DWT CTRL: SYNCTAP Mask */

#define DWT_CTRL_CYCTAP_Pos               9
/*!< DWT CTRL: CYCTAP Position */
#define DWT_CTRL_CYCTAP_Msk              (0x1UL << DWT_CTRL_CYCTAP_Pos)
/*!< DWT CTRL: CYCTAP Mask */

#define DWT_CTRL_POSTINIT_Pos             5
/*!< DWT CTRL: POSTINIT Position */
#define DWT_CTRL_POSTINIT_Msk            (0xFUL <<
DWT_CTRL_POSTINIT_Pos)               /*!< DWT CTRL: POSTINIT Mask */

#define DWT_CTRL_POSTPRESET_Pos           1
/*!< DWT CTRL: POSTPRESET Position */
#define DWT_CTRL_POSTPRESET_Msk          (0xFUL <<
DWT_CTRL_POSTPRESET_Pos)             /*!< DWT CTRL: POSTPRESET Mask */

```

```

#define DWT_CTRL_CYCCNTENA_Pos 0
/*!< DWT CTRL: CYCCNTENA Position */
#define DWT_CTRL_CYCCNTENA_Msk (0x1UL /*<<
DWT_CTRL_CYCCNTENA_Pos*/) /*!< DWT CTRL: CYCCNTENA Mask */

/* DWT CPI Count Register Definitions */
#define DWT_CPICNT_CPICNT_Pos 0
/*!< DWT CPICNT: CPICNT Position */
#define DWT_CPICNT_CPICNT_Msk (0xFFUL /*<<
DWT_CPICNT_CPICNT_Pos*/) /*!< DWT CPICNT: CPICNT Mask */

/* DWT Exception Overhead Count Register Definitions */
#define DWT_EXCCNT_EXCCNT_Pos 0
/*!< DWT EXCCNT: EXCCNT Position */
#define DWT_EXCCNT_EXCCNT_Msk (0xFFUL /*<<
DWT_EXCCNT_EXCCNT_Pos*/) /*!< DWT EXCCNT: EXCCNT Mask */

/* DWT Sleep Count Register Definitions */
#define DWT_SLEEP_CNT_SLEEP_CNT_Pos 0
/*!< DWT SLEEP_CNT: SLEEP_CNT Position */
#define DWT_SLEEP_CNT_SLEEP_CNT_Msk (0xFFUL /*<<
DWT_SLEEP_CNT_SLEEP_CNT_Pos*/) /*!< DWT SLEEP_CNT: SLEEP_CNT Mask */

/* DWT LSU Count Register Definitions */
#define DWT_LSUCNT_LSUCNT_Pos 0
/*!< DWT LSUCNT: LSUCNT Position */
#define DWT_LSUCNT_LSUCNT_Msk (0xFFUL /*<<
DWT_LSUCNT_LSUCNT_Pos*/) /*!< DWT LSUCNT: LSUCNT Mask */

/* DWT Folded-instruction Count Register Definitions */
#define DWT_FOLDCNT_FOLDCNT_Pos 0
/*!< DWT FOLDCNT: FOLDCNT Position */
#define DWT_FOLDCNT_FOLDCNT_Msk (0xFFUL /*<<
DWT_FOLDCNT_FOLDCNT_Pos*/) /*!< DWT FOLDCNT: FOLDCNT Mask */

/* DWT Comparator Mask Register Definitions */
#define DWT_MASK_MASK_Pos 0
/*!< DWT MASK: MASK Position */
#define DWT_MASK_MASK_Msk (0x1FUL /*<<
DWT_MASK_MASK_Pos*/) /*!< DWT MASK: MASK Mask */

/* DWT Comparator Function Register Definitions */
#define DWT_FUNCTION_MATCHED_Pos 24
/*!< DWT FUNCTION: MATCHED Position */
#define DWT_FUNCTION_MATCHED_Msk (0x1UL <<
DWT_FUNCTION_MATCHED_Pos) /*!< DWT FUNCTION: MATCHED Mask */

#define DWT_FUNCTION_DATAVADDR1_Pos 16
/*!< DWT FUNCTION: DATAVADDR1 Position */
#define DWT_FUNCTION_DATAVADDR1_Msk (0xFUL <<
DWT_FUNCTION_DATAVADDR1_Pos) /*!< DWT FUNCTION: DATAVADDR1 Mask */

```

```

#define DWT_FUNCTION_DATAVADDR0_Pos           12
/*!< DWT FUNCTION: DATAVADDR0 Position */
#define DWT_FUNCTION_DATAVADDR0_Msk          (0xFUL <<
DWT_FUNCTION_DATAVADDR0_Pos)             /*!< DWT FUNCTION: DATAVADDR0 Mask */

#define DWT_FUNCTION_DATAVSIZE_Pos          10
/*!< DWT FUNCTION: DATAVSIZE Position */
#define DWT_FUNCTION_DATAVSIZE_Msk          (0x3UL <<
DWT_FUNCTION_DATAVSIZE_Pos)             /*!< DWT FUNCTION: DATAVSIZE Mask */

#define DWT_FUNCTION_LNK1ENA_Pos            9
/*!< DWT FUNCTION: LNK1ENA Position */
#define DWT_FUNCTION_LNK1ENA_Msk            (0x1UL <<
DWT_FUNCTION_LNK1ENA_Pos)              /*!< DWT FUNCTION: LNK1ENA Mask */

#define DWT_FUNCTION_DATAVMATCH_Pos         8
/*!< DWT FUNCTION: DATAVMATCH Position */
#define DWT_FUNCTION_DATAVMATCH_Msk         (0x1UL <<
DWT_FUNCTION_DATAVMATCH_Pos)           /*!< DWT FUNCTION: DATAVMATCH Mask */

#define DWT_FUNCTION_CYCMATCH_Pos           7
/*!< DWT FUNCTION: CYCMATCH Position */
#define DWT_FUNCTION_CYCMATCH_Msk           (0x1UL <<
DWT_FUNCTION_CYCMATCH_Pos)             /*!< DWT FUNCTION: CYCMATCH Mask */

#define DWT_FUNCTION_EMITRANGE_Pos          5
/*!< DWT FUNCTION: EMITRANGE Position */
#define DWT_FUNCTION_EMITRANGE_Msk          (0x1UL <<
DWT_FUNCTION_EMITRANGE_Pos)             /*!< DWT FUNCTION: EMITRANGE Mask */

#define DWT_FUNCTION_FUNCTION_Pos           0
/*!< DWT FUNCTION: FUNCTION Position */
#define DWT_FUNCTION_FUNCTION_Msk           (0xFUL /*<<
DWT_FUNCTION_FUNCTION_Pos*/)           /*!< DWT FUNCTION: FUNCTION Mask */

/*}@ */ /* end of group CMSIS_DWT */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_TPI      Trace Port Interface (TPI)
 \brief      Type definitions for the Trace Port Interface (TPI)
 @{
 */

/** \brief Structure type to access the Trace Port Interface Register
(TPI).
 */
typedef struct
{
    __IO uint32_t SSPSR;                  /*!< Offset: 0x000 (R/ )
Supported Parallel Port Size Register   */
    __IO uint32_t CSPSR;                  /*!< Offset: 0x004 (R/W)
Current Parallel Port Size Register */
    uint32_t RESERVED0[2];
}

```

```

    __IO uint32_t ACPR;                      /*!< Offset: 0x010 (R/W)
Asynchronous Clock Prescaler Register */
    uint32_t RESERVED1[55];
    __IO uint32_t SPPR;                      /*!< Offset: 0x0F0 (R/W)
Selected Pin Protocol Register */
    uint32_t RESERVED2[131];
    __I  uint32_t FFSR;                      /*!< Offset: 0x300 (R/ )
Formatter and Flush Status Register */
    __IO uint32_t FFCR;                      /*!< Offset: 0x304 (R/W)
Formatter and Flush Control Register */
    __I  uint32_t FSCR;                      /*!< Offset: 0x308 (R/ )
Formatter Synchronization Counter Register */
    uint32_t RESERVED3[759];
    __I  uint32_t TRIGGER;                   /*!< Offset: 0xEE8 (R/ )
TRIGGER */
    __I  uint32_t FIFO0;                     /*!< Offset: 0xEEC (R/ )
Integration ETM Data */
    __I  uint32_t ITATBCTR2;                 /*!< Offset: 0xEF0 (R/ )
ITATBCTR2 */
    uint32_t RESERVED4[1];
    __I  uint32_t ITATBCTR0;                 /*!< Offset: 0xEF8 (R/ )
ITATBCTR0 */
    __I  uint32_t FIFO1;                     /*!< Offset: 0xEFC (R/ )
Integration ITM Data */
    __IO uint32_t ITCTRL;                   /*!< Offset: 0xF00 (R/W)
Integration Mode Control */
    uint32_t RESERVED5[39];
    __IO uint32_t CLAIMSET;                  /*!< Offset: 0xFA0 (R/W) Claim
tag set */
    __IO uint32_t CLAIMCLR;                  /*!< Offset: 0xFA4 (R/W) Claim
tag clear */
    uint32_t RESERVED7[8];
    __I  uint32_t DEVID;                    /*!< Offset: 0xFC8 (R/ )
TPIU_DEVID */
    __I  uint32_t DEVTYPE;                  /*!< Offset: 0xFCC (R/ )
TPIU_DEVTYPE */
} TPI_Type;

/* TPI Asynchronous Clock Prescaler Register Definitions */
#define TPI_ACPR_PRESCALER_Pos          0
/*!< TPI ACPR: PRESCALER Position */
#define TPI_ACPR_PRESCALER_Msk         (0x1FFFUL /*<<
TPI_ACPR_PRESCALER_Pos*/) /*!< TPI ACPR: PRESCALER Mask */

/* TPI Selected Pin Protocol Register Definitions */
#define TPI_SPPR_TXMODE_Pos           0
/*!< TPI SPPR: TXMODE Position */
#define TPI_SPPR_TXMODE_Msk          (0x3UL /*<<
TPI_SPPR_TXMODE_Pos*/) /*!< TPI SPPR: TXMODE Mask */

/* TPI Formatter and Flush Status Register Definitions */
#define TPI_FFSR_FtNonStop_Pos        3
/*!< TPI FFSR: FtNonStop Position */

```

```

#define TPI_FFSR_FtNonStop_Msk          (0x1UL <<
TPI_FFSR_FtNonStop_Pos)             /*!< TPI FFSR: FtNonStop Mask */

#define TPI_FFSR_TCPresent_Pos         2
/*!< TPI FFSR: TCPresent Position */
#define TPI_FFSR_TCPresent_Msk        (0x1UL <<
TPI_FFSR_TCPresent_Pos)            /*!< TPI FFSR: TCPresent Mask */

#define TPI_FFSR_FtStopped_Pos        1
/*!< TPI FFSR: FtStopped Position */
#define TPI_FFSR_FtStopped_Msk        (0x1UL <<
TPI_FFSR_FtStopped_Pos)            /*!< TPI FFSR: FtStopped Mask */

#define TPI_FFSR_FlInProg_Pos          0
/*!< TPI FFSR: FlInProg Position */
#define TPI_FFSR_FlInProg_Msk         (0x1UL /*<<
TPI_FFSR_FlInProg_Pos*/)           /*!< TPI FFSR: FlInProg Mask */

/* TPI Formatter and Flush Control Register Definitions */
#define TPI_FFCR_TrigIn_Pos           8
/*!< TPI FFCR: TrigIn Position */
#define TPI_FFCR_TrigIn_Msk          (0x1UL << TPI_FFCR_TrigIn_Pos)
/*!< TPI FFCR: TrigIn Mask */

#define TPI_FFCR_EnFCont_Pos          1
/*!< TPI FFCR: EnFCont Position */
#define TPI_FFCR_EnFCont_Msk         (0x1UL <<
TPI_FFCR_EnFCont_Pos)              /*!< TPI FFCR: EnFCont Mask */

/* TPI TRIGGER Register Definitions */
#define TPI_TRIGGER_TRIGGER_Pos        0
/*!< TPI TRIGGER: TRIGGER Position */
#define TPI_TRIGGER_TRIGGER_Msk       (0x1UL /*<<
TPI_TRIGGER_TRIGGER_Pos*/)         /*!< TPI TRIGGER: TRIGGER Mask */

/* TPI Integration ETM Data Register Definitions (FIFO0) */
#define TPI_FIFO0_ITM_ATVALID_Pos     29
/*!< TPI FIFO0: ITM_ATVALID Position */
#define TPI_FIFO0_ITM_ATVALID_Msk     (0x3UL <<
TPI_FIFO0_ITM_ATVALID_Pos)          /*!< TPI FIFO0: ITM_ATVALID Mask */

#define TPI_FIFO0_ITM_bytecount_Pos    27
/*!< TPI FIFO0: ITM_bytecount Position */
#define TPI_FIFO0_ITM_bytecount_Msk    (0x3UL <<
TPI_FIFO0_ITM_bytecount_Pos)          /*!< TPI FIFO0: ITM_bytecount Mask */

#define TPI_FIFO0_ETM_ATVALID_Pos     26
/*!< TPI FIFO0: ETM_ATVALID Position */
#define TPI_FIFO0_ETM_ATVALID_Msk     (0x3UL <<
TPI_FIFO0_ETM_ATVALID_Pos)          /*!< TPI FIFO0: ETM_ATVALID Mask */

#define TPI_FIFO0_ETM_bytecount_Pos    24
/*!< TPI FIFO0: ETM_bytecount Position */

```

```

#define TPI_FIFO0_ETM_bytecount_Msk          (0x3UL <<
TPI_FIFO0_ETM_bytecount_Pos)             /*!< TPI FIFO0: ETM_bytecount Mask */

#define TPI_FIFO0_ETM2_Pos                  16
/*!< TPI FIFO0: ETM2 Position */
#define TPI_FIFO0_ETM2_Msk                (0xFFUL << TPI_FIFO0_ETM2_Pos)
/*!< TPI FIFO0: ETM2 Mask */

#define TPI_FIFO0_ETM1_Pos                  8
/*!< TPI FIFO0: ETM1 Position */
#define TPI_FIFO0_ETM1_Msk                (0xFFUL << TPI_FIFO0_ETM1_Pos)
/*!< TPI FIFO0: ETM1 Mask */

#define TPI_FIFO0_ETM0_Pos                  0
/*!< TPI FIFO0: ETM0 Position */
#define TPI_FIFO0_ETM0_Msk                (0xFFUL /*<<
TPI_FIFO0_ETM0_Pos*/)             /*!< TPI FIFO0: ETM0 Mask */

/* TPI ITATBCTR2 Register Definitions */
#define TPI_ITATBCTR2_ATREADY_Pos         0
/*!< TPI ITATBCTR2: ATREADY Position */
#define TPI_ITATBCTR2_ATREADY_Msk        (0x1UL /*<<
TPI_ITATBCTR2_ATREADY_Pos*/)      /*!< TPI ITATBCTR2: ATREADY Mask */

/* TPI Integration ITM Data Register Definitions (FIFO1) */
#define TPI_FIFO1_ITM_ATVALID_Pos        29
/*!< TPI FIFO1: ITM_ATVALID Position */
#define TPI_FIFO1_ITM_ATVALID_Msk        (0x3UL <<
TPI_FIFO1_ITM_ATVALID_Pos)         /*!< TPI FIFO1: ITM_ATVALID Mask */

#define TPI_FIFO1_ITM_bytecount_Pos       27
/*!< TPI FIFO1: ITM_bytecount Position */
#define TPI_FIFO1_ITM_bytecount_Msk     (0x3UL <<
TPI_FIFO1_ITM_bytecount_Pos)        /*!< TPI FIFO1: ITM_bytecount Mask */

#define TPI_FIFO1_ETM_ATVALID_Pos        26
/*!< TPI FIFO1: ETM_ATVALID Position */
#define TPI_FIFO1_ETM_ATVALID_Msk        (0x3UL <<
TPI_FIFO1_ETM_ATVALID_Pos)         /*!< TPI FIFO1: ETM_ATVALID Mask */

#define TPI_FIFO1_ETM_bytecount_Pos       24
/*!< TPI FIFO1: ETM_bytecount Position */
#define TPI_FIFO1_ETM_bytecount_Msk     (0x3UL <<
TPI_FIFO1_ETM_bytecount_Pos)        /*!< TPI FIFO1: ETM_bytecount Mask */

#define TPI_FIFO1_ITM2_Pos                 16
/*!< TPI FIFO1: ITM2 Position */
#define TPI_FIFO1_ITM2_Msk               (0xFFUL << TPI_FIFO1_ITM2_Pos)
/*!< TPI FIFO1: ITM2 Mask */

#define TPI_FIFO1_ITM1_Pos                 8
/*!< TPI FIFO1: ITM1 Position */
#define TPI_FIFO1_ITM1_Msk               (0xFFUL << TPI_FIFO1_ITM1_Pos)
/*!< TPI FIFO1: ITM1 Mask */

```

```

#define TPI_FIFO1_ITM0_Pos 0
/*!< TPI FIFO1: ITM0 Position */
#define TPI_FIFO1_ITM0_Msk (0xFFUL /*<<
TPI_FIFO1_ITM0_Pos*/) /*!< TPI FIFO1: ITM0 Mask */

/* TPI ITATBCTR0 Register Definitions */
#define TPI_ITATBCTR0_ATREADY_Pos 0
/*!< TPI ITATBCTR0: ATREADY Position */
#define TPI_ITATBCTR0_ATREADY_Msk (0x1UL /*<<
TPI_ITATBCTR0_ATREADY_Pos*/) /*!< TPI ITATBCTR0: ATREADY Mask */

/* TPI Integration Mode Control Register Definitions */
#define TPI_ITCTRL_Mode_Pos 0
/*!< TPI ITCTRL: Mode Position */
#define TPI_ITCTRL_Mode_Msk (0x1UL /*<<
TPI_ITCTRL_Mode_Pos*/) /*!< TPI ITCTRL: Mode Mask */

/* TPI DEVID Register Definitions */
#define TPI_DEVID_NRZVALID_Pos 11
/*!< TPI DEVID: NRZVALID Position */
#define TPI_DEVID_NRZVALID_Msk (0x1UL <<
TPI_DEVID_NRZVALID_Pos) /*!< TPI DEVID: NRZVALID Mask */

#define TPI_DEVID_MANCVALID_Pos 10
/*!< TPI DEVID: MANCVALID Position */
#define TPI_DEVID_MANCVALID_Msk (0x1UL <<
TPI_DEVID_MANCVALID_Pos) /*!< TPI DEVID: MANCVALID Mask */

#define TPI_DEVID_PTINVALIDID_Pos 9
/*!< TPI DEVID: PTINVALIDID Position */
#define TPI_DEVID_PTINVALIDID_Msk (0x1UL <<
TPI_DEVID_PTINVALIDID_Pos) /*!< TPI DEVID: PTINVALIDID Mask */

#define TPI_DEVID_MinBufSz_Pos 6
/*!< TPI DEVID: MinBufSz Position */
#define TPI_DEVID_MinBufSz_Msk (0x7UL <<
TPI_DEVID_MinBufSz_Pos) /*!< TPI DEVID: MinBufSz Mask */

#define TPI_DEVID_AsynClkIn_Pos 5
/*!< TPI DEVID: AsynClkIn Position */
#define TPI_DEVID_AsynClkIn_Msk (0x1UL <<
TPI_DEVID_AsynClkIn_Pos) /*!< TPI DEVID: AsynClkIn Mask */

#define TPI_DEVID_NrTraceInput_Pos 0
/*!< TPI DEVID: NrTraceInput Position */
#define TPI_DEVID_NrTraceInput_Msk (0x1FUL /*<<
TPI_DEVID_NrTraceInput_Pos*/) /*!< TPI DEVID: NrTraceInput Mask */

/* TPI DEVTYPE Register Definitions */
#define TPI_DEVTYPE_MajorType_Pos 4
/*!< TPI DEVTYPE: MajorType Position */
#define TPI_DEVTYPE_MajorType_Msk (0xFUL <<
TPI_DEVTYPE_MajorType_Pos) /*!< TPI DEVTYPE: MajorType Mask */

```

```

#define TPI_DEVTYPE_SubType_Pos 0
/*!< TPI DEVTYPE: SubType Position */
#define TPI_DEVTYPE_SubType_Msk (0xFUL /*<<
TPI_DEVTYPE_SubType_Pos*/) /*!< TPI DEVTYPE: SubType Mask */

/*@}/*/ /* end of group CMSIS_TPI */

#if (_MPU_PRESENT == 1)
/** \ingroup CMSIS_core_register
 \defgroup CMSIS_MPU Memory Protection Unit (MPU)
 \brief Type definitions for the Memory Protection Unit (MPU)
 @{
 */

/** \brief Structure type to access the Memory Protection Unit (MPU).
 */
typedef struct
{
    __I uint32_t TYPE;                                /*!< Offset: 0x000 (R/ ) MPU
Type Register                                     */
    __IO uint32_t CTRL;                               /*!< Offset: 0x004 (R/W) MPU
Control Register                                    */
    __IO uint32_t RNR;                                /*!< Offset: 0x008 (R/W) MPU
Region RNRber Register                           */
    __IO uint32_t RBAR;                               /*!< Offset: 0x00C (R/W) MPU
Region Base Address Register                      */
    __IO uint32_t RASR;                               /*!< Offset: 0x010 (R/W) MPU
Region Attribute and Size Register               */
    __IO uint32_t RBAR_A1;                            /*!< Offset: 0x014 (R/W) MPU
Alias 1 Region Base Address Register            */
    __IO uint32_t RASR_A1;                            /*!< Offset: 0x018 (R/W) MPU
Alias 1 Region Attribute and Size Register     */
    __IO uint32_t RBAR_A2;                            /*!< Offset: 0x01C (R/W) MPU
Alias 2 Region Base Address Register            */
    __IO uint32_t RASR_A2;                            /*!< Offset: 0x020 (R/W) MPU
Alias 2 Region Attribute and Size Register     */
    __IO uint32_t RBAR_A3;                            /*!< Offset: 0x024 (R/W) MPU
Alias 3 Region Base Address Register            */
    __IO uint32_t RASR_A3;                            /*!< Offset: 0x028 (R/W) MPU
Alias 3 Region Attribute and Size Register     */
} MPU_Type;

/* MPU Type Register */
#define MPU_TYPE_IREGION_Pos 16
/*!< MPU TYPE: IREGION Position */
#define MPU_TYPE_IREGION_Msk (0xFFUL <<
MPU_TYPE_IREGION_Pos) /*!< MPU TYPE: IREGION Mask */

#define MPU_TYPE_DREGION_Pos 8
/*!< MPU TYPE: DREGION Position */
#define MPU_TYPE_DREGION_Msk (0xFFUL <<
MPU_TYPE_DREGION_Pos) /*!< MPU TYPE: DREGION Mask */

```

```

#define MPU_TYPE_SEPARATE_Pos 0
/*!< MPU TYPE: SEPARATE Position */
#define MPU_TYPE_SEPARATE_Msk (1UL /*<<
MPU_TYPE_SEPARATE_Pos*) /*!< MPU TYPE: SEPARATE Mask */

/* MPU Control Register */
#define MPU_CTRL_PRIVDEFENA_Pos 2
/*!< MPU CTRL: PRIVDEFENA Position */
#define MPU_CTRL_PRIVDEFENA_Msk (1UL <<
MPU_CTRL_PRIVDEFENA_Pos) /*!< MPU CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos 1
/*!< MPU CTRL: HFNMIENA Position */
#define MPU_CTRL_HFNMIENA_Msk (1UL << MPU_CTRL_HFNMIENA_Pos)
/*!< MPU CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos 0
/*!< MPU CTRL: ENABLE Position */
#define MPU_CTRL_ENABLE_Msk (1UL /*<<
MPU_CTRL_ENABLE_Pos) /*!< MPU CTRL: ENABLE Mask */

/* MPU Region Number Register */
#define MPU_RNR_REGION_Pos 0
/*!< MPU RNR: REGION Position */
#define MPU_RNR_REGION_Msk (0xFFFUL /*<<
MPU_RNR_REGION_Pos) /*!< MPU RNR: REGION Mask */

/* MPU Region Base Address Register */
#define MPU_RBAR_ADDR_Pos 5
/*!< MPU RBAR: ADDR Position */
#define MPU_RBAR_ADDR_Msk (0x7FFFFFFUL <<
MPU_RBAR_ADDR_Pos) /*!< MPU RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos 4
/*!< MPU RBAR: VALID Position */
#define MPU_RBAR_VALID_Msk (1UL << MPU_RBAR_VALID_Pos)
/*!< MPU RBAR: VALID Mask */

#define MPU_RBAR_REGION_Pos 0
/*!< MPU RBAR: REGION Position */
#define MPU_RBAR_REGION_Msk (0xFUL /*<<
MPU_RBAR_REGION_Pos) /*!< MPU RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */
#define MPU_RASR_ATTRS_Pos 16
/*!< MPU RASR: MPU Region Attribute field Position */
#define MPU_RASR_ATTRS_Msk (0xFFFFFUL <<
MPU_RASR_ATTRS_Pos) /*!< MPU RASR: MPU Region Attribute
field Mask */

#define MPU_RASR_XN_Pos 28
/*!< MPU RASR: ATTRS.XN Position */

```

```

#define MPU_RASR_XN_Msk          (1UL << MPU_RASR_XN_Pos)
/*!< MPU RASR: ATTRS.XN Mask */

#define MPU_RASR_AP_Pos          24
/*!< MPU RASR: ATTRS.AP Position */
#define MPU_RASR_AP_Msk          (0x7UL << MPU_RASR_AP_Pos)
/*!< MPU RASR: ATTRS.AP Mask */

#define MPU_RASR_TEX_Pos          19
/*!< MPU RASR: ATTRS.TEX Position */
#define MPU_RASR_TEX_Msk          (0x7UL << MPU_RASR_TEX_Pos)
/*!< MPU RASR: ATTRS.TEX Mask */

#define MPU_RASR_S_Pos             18
/*!< MPU RASR: ATTRS.S Position */
#define MPU_RASR_S_Msk             (1UL << MPU_RASR_S_Pos)
/*!< MPU RASR: ATTRS.S Mask */

#define MPU_RASR_C_Pos             17
/*!< MPU RASR: ATTRS.C Position */
#define MPU_RASR_C_Msk             (1UL << MPU_RASR_C_Pos)
/*!< MPU RASR: ATTRS.C Mask */

#define MPU_RASR_B_Pos             16
/*!< MPU RASR: ATTRS.B Position */
#define MPU_RASR_B_Msk             (1UL << MPU_RASR_B_Pos)
/*!< MPU RASR: ATTRS.B Mask */

#define MPU_RASR_SRD_Pos            8
/*!< MPU RASR: Sub-Region Disable Position */
#define MPU_RASR_SRD_Msk           (0xFFUL << MPU_RASR_SRD_Pos)
/*!< MPU RASR: Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos           1
/*!< MPU RASR: Region Size Field Position */
#define MPU_RASR_SIZE_Msk           (0x1FUL << MPU_RASR_SIZE_Pos)
/*!< MPU RASR: Region Size Field Mask */

#define MPU_RASR_ENABLE_Pos          0
/*!< MPU RASR: Region enable bit Position */
#define MPU_RASR_ENABLE_Msk          (1UL /*<<
MPU_RASR_ENABLE_Pos*/ /*!< MPU RASR: Region enable bit
Disable Mask */

/*@} end of group CMSIS_MPUS */
#endif

#if (_FPU_PRESENT == 1)
/** \ingroup CMSIS_core_register
 \defgroup CMSIS_FPU Floating Point Unit (FPU)
 \brief Type definitions for the Floating Point Unit (FPU)
 @{
 */

```

```

/** \brief Structure type to access the Floating Point Unit (FPU).
 */
typedef struct
{
    uint32_t RESERVED0[1];
    __IO uint32_t FPCCR;           /*!< Offset: 0x004 (R/W) */
    Floating-Point Context Control Register
    __IO uint32_t FPCAR;           /*!< Offset: 0x008 (R/W) */
    Floating-Point Context Address Register
    __IO uint32_t FPDSCR;          /*!< Offset: 0x00C (R/W) */
    Floating-Point Default Status Control Register
    __I  uint32_t MVFR0;           /*!< Offset: 0x010 (R/ ) Media
and FP Feature Register 0
    __I  uint32_t MVFR1;           /*!< Offset: 0x014 (R/ ) Media
and FP Feature Register 1
} FPU_Type;

/* Floating-Point Context Control Register */
#define FPU_FPCCR_AS PEN_Pos           31
/*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_AS PEN_Msk            (1UL << FPU_FPCCR_AS PEN_Pos)
/*!< FPCCR: ASPEN bit Mask */

#define FPU_FPCCR_LSPEN_Pos           30
/*!< FPCCR: LSPEN Position */
#define FPU_FPCCR_LSPEN_Msk            (1UL << FPU_FPCCR_LSPEN_Pos)
/*!< FPCCR: LSPEN bit Mask */

#define FPU_FPCCR_MONRDY_Pos          8
/*!< FPCCR: MONRDY Position */
#define FPU_FPCCR_MONRDY_Msk            (1UL << FPU_FPCCR_MONRDY_Pos)
/*!< FPCCR: MONRDY bit Mask */

#define FPU_FPCCR_BFRDY_Pos           6
/*!< FPCCR: BFRDY Position */
#define FPU_FPCCR_BFRDY_Msk            (1UL << FPU_FPCCR_BFRDY_Pos)
/*!< FPCCR: BFRDY bit Mask */

#define FPU_FPCCR_MMRDY_Pos           5
/*!< FPCCR: MMRDY Position */
#define FPU_FPCCR_MMRDY_Msk            (1UL << FPU_FPCCR_MMRDY_Pos)
/*!< FPCCR: MMRDY bit Mask */

#define FPU_FPCCR_HFRDY_Pos           4
/*!< FPCCR: HFRDY Position */
#define FPU_FPCCR_HFRDY_Msk            (1UL << FPU_FPCCR_HFRDY_Pos)
/*!< FPCCR: HFRDY bit Mask */

#define FPU_FPCCR_THREAD_Pos          3
/*!< FPCCR: processor mode bit Position */
#define FPU_FPCCR_THREAD_Msk            (1UL << FPU_FPCCR_THREAD_Pos)
/*!< FPCCR: processor mode active bit Mask */

```

```

#define FPU_FPCCR_USER_Pos 1
/*!< FPCCR: privilege level bit Position */
#define FPU_FPCCR_USER_Msk (1UL << FPU_FPCCR_USER_Pos)
/*!< FPCCR: privilege level bit Mask */

#define FPU_FPCCR_LSPACT_Pos 0
/*!< FPCCR: Lazy state preservation active bit Position */
#define FPU_FPCCR_LSPACT_Msk (1UL /*<<
FPU_FPCCR_LSPACT_Pos*/) /*!< FPCCR: Lazy state preservation
active bit Mask */

/* Floating-Point Context Address Register */
#define FPCAR_ADDRESS_Pos 3
/*!< FPCAR: ADDRESS bit Position */
#define FPCAR_ADDRESS_Msk (0x1FFFFFFFUL <<
FPCAR_ADDRESS_Pos) /*!< FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */
#define FPDSCR_AHP_Pos 26
/*!< FPDSCR: AHP bit Position */
#define FPDSCR_AHP_Msk (1UL << FPDSCR_AHP_Pos)
/*!< FPDSCR: AHP bit Mask */

#define FPDSCR_DN_Pos 25
/*!< FPDSCR: DN bit Position */
#define FPDSCR_DN_Msk (1UL << FPDSCR_DN_Pos)
/*!< FPDSCR: DN bit Mask */

#define FPDSCR_FZ_Pos 24
/*!< FPDSCR: FZ bit Position */
#define FPDSCR_FZ_Msk (1UL << FPDSCR_FZ_Pos)
/*!< FPDSCR: FZ bit Mask */

#define FPDSCR_RMode_Pos 22
/*!< FPDSCR: RMode bit Position */
#define FPDSCR_RMode_Msk (3UL << FPDSCR_RMode_Pos)
/*!< FPDSCR: RMode bit Mask */

/* Media and FP Feature Register 0 */
#define MVFR0_FP_rounding_modes_Pos 28
/*!< MVFR0: FP rounding modes bits Position */
#define MVFR0_FP_rounding_modes_Msk (0xFUL <<
MVFR0_FP_rounding_modes_Pos) /*!< MVFR0: FP rounding modes bits
Mask */

#define MVFR0_Short_vectors_Pos 24
/*!< MVFR0: Short vectors bits Position */
#define MVFR0_Short_vectors_Msk (0xFUL <<
MVFR0_Short_vectors_Pos) /*!< MVFR0: Short vectors bits Mask */

#define MVFR0_Square_root_Pos 20
/*!< MVFR0: Square root bits Position */

```

```

#define FPU_MVFR0_Square_root_Msk          (0xFUL <<
FPU_MVFR0_Square_root_Pos)             /*!< MVFR0: Square root bits Mask */

#define FPU_MVFR0_Divide_Pos              16
/*!< MVFR0: Divide bits Position */
#define FPU_MVFR0_Divide_Msk            (0xFUL <<
FPU_MVFR0_Divide_Pos)                /*!< MVFR0: Divide bits Mask */

#define FPU_MVFR0_FP_excep_trapping_Pos 12
/*!< MVFR0: FP exception trapping bits Position */
#define FPU_MVFR0_FP_excep_trapping_Msk  (0xFUL <<
FPU_MVFR0_FP_excep_trapping_Pos)     /*!< MVFR0: FP exception trapping
bits Mask */

#define FPU_MVFR0_Double_precision_Pos    8
/*!< MVFR0: Double-precision bits Position */
#define FPU_MVFR0_Double_precision_Msk   (0xFUL <<
FPU_MVFR0_Double_precision_Pos)      /*!< MVFR0: Double-precision bits
Mask */

#define FPU_MVFR0_Single_precision_Pos    4
/*!< MVFR0: Single-precision bits Position */
#define FPU_MVFR0_Single_precision_Msk   (0xFUL <<
FPU_MVFR0_Single_precision_Pos)      /*!< MVFR0: Single-precision bits
Mask */

#define FPU_MVFR0_A SIMD_registers_Pos    0
/*!< MVFR0: A SIMD registers bits Position */
#define FPU_MVFR0_A SIMD_registers_Msk   (0xFUL /*<<
FPU_MVFR0_A SIMD_registers_Pos*/)    /*!< MVFR0: A SIMD registers bits Mask
*/

/* Media and FP Feature Register 1 */
#define FPU_MVFR1_FP_fused_MAC_Pos       28
/*!< MVFR1: FP fused MAC bits Position */
#define FPU_MVFR1_FP_fused_MAC_Msk      (0xFUL <<
FPU_MVFR1_FP_fused_MAC_Pos)         /*!< MVFR1: FP fused MAC bits Mask
*/

#define FPU_MVFR1_FP_HPFP_Pos           24
/*!< MVFR1: FP HPFP bits Position */
#define FPU_MVFR1_FP_HPFP_Msk          (0xFUL <<
FPU_MVFR1_FP_HPFP_Pos)             /*!< MVFR1: FP HPFP bits Mask */

#define FPU_MVFR1_D_NaN_mode_Pos        4
/*!< MVFR1: D_NaN mode bits Position */
#define FPU_MVFR1_D_NaN_mode_Msk       (0xFUL <<
FPU_MVFR1_D_NaN_mode_Pos)          /*!< MVFR1: D_NaN mode bits Mask */

#define FPU_MVFR1_Ftz_mode_Pos          0
/*!< MVFR1: Ftz mode bits Position */
#define FPU_MVFR1_Ftz_mode_Msk         (0xFUL /*<<
FPU_MVFR1_Ftz_mode_Pos*/)          /*!< MVFR1: Ftz mode bits Mask */

```

```

/*@} end of group CMSIS_FPU */
#endif

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CoreDebug Core Debug Registers (CoreDebug)
 \brief Type definitions for the Core Debug Registers
 @{
 */

/** \brief Structure type to access the Core Debug Register (CoreDebug) .
 */
typedef struct
{
    __IO uint32_t DHCSR;           /*!< Offset: 0x000 (R/W)  Debug
Halting Control and Status Register */
    __O uint32_t DCRSR;           /*!< Offset: 0x004 ( /W)  Debug
Core Register Selector Register */
    __IO uint32_t DCRDR;           /*!< Offset: 0x008 (R/W)  Debug
Core Register Data Register */
    __IO uint32_t DEMCR;           /*!< Offset: 0x00C (R/W)  Debug
Exception and Monitor Control Register */
} CoreDebug_Type;

/* Debug Halting Control and Status Register */
#define CoreDebug_DHCSR_DBGKEY_Pos      16
/*!< CoreDebug DHCSR: DBGKEY Position */
#define CoreDebug_DHCSR_DBGKEY_Msk      (0xFFFFUL <<
CoreDebug_DHCSR_DBGKEY_Pos)        /*!< CoreDebug DHCSR: DBGKEY Mask */

#define CoreDebug_DHCSR_S_RESET_ST_Pos   25
/*!< CoreDebug DHCSR: S_RESET_ST Position */
#define CoreDebug_DHCSR_S_RESET_ST_Msk   (1UL <<
CoreDebug_DHCSR_S_RESET_ST_Pos)     /*!< CoreDebug DHCSR: S_RESET_ST
Mask */

#define CoreDebug_DHCSR_S_RETIRE_ST_Pos  24
/*!< CoreDebug DHCSR: S_RETIRE_ST Position */
#define CoreDebug_DHCSR_S_RETIRE_ST_Msk  (1UL <<
CoreDebug_DHCSR_S_RETIRE_ST_Pos)    /*!< CoreDebug DHCSR: S_RETIRE_ST
Mask */

#define CoreDebug_DHCSR_S_LOCKUP_Pos     19
/*!< CoreDebug DHCSR: S_LOCKUP Position */
#define CoreDebug_DHCSR_S_LOCKUP_Msk     (1UL <<
CoreDebug_DHCSR_S_LOCKUP_Pos)       /*!< CoreDebug DHCSR: S_LOCKUP
Mask */

#define CoreDebug_DHCSR_S_SLEEP_Pos      18
/*!< CoreDebug DHCSR: S_SLEEP Position */
#define CoreDebug_DHCSR_S_SLEEP_Msk      (1UL <<
CoreDebug_DHCSR_S_SLEEP_Pos)        /*!< CoreDebug DHCSR: S_SLEEP Mask
*/

```

```

#define CoreDebug_DHCSR_S_HALT_Pos           17
/*!< CoreDebug DHCSR: S_HALT Position */
#define CoreDebug_DHCSR_S_HALT_Msk          (1UL <<
CoreDebug_DHCSR_S_HALT_Pos)             /*!< CoreDebug DHCSR: S_HALT Mask
*/
/* */

#define CoreDebug_DHCSR_S_REGRDY_Pos        16
/*!< CoreDebug DHCSR: S_REGRDY Position */
#define CoreDebug_DHCSR_S_REGRDY_Msk        (1UL <<
CoreDebug_DHCSR_S_REGRDY_Pos)           /*!< CoreDebug DHCSR: S_REGRDY
Mask */

#define CoreDebug_DHCSR_C_SNAPSTALL_Pos     5
/*!< CoreDebug DHCSR: C_SNAPSTALL Position */
#define CoreDebug_DHCSR_C_SNAPSTALL_Msk     (1UL <<
CoreDebug_DHCSR_C_SNAPSTALL_Pos)         /*!< CoreDebug DHCSR: C_SNAPSTALL
Mask */

#define CoreDebug_DHCSR_C_MASKINTS_Pos      3
/*!< CoreDebug DHCSR: C_MASKINTS Position */
#define CoreDebug_DHCSR_C_MASKINTS_Msk      (1UL <<
CoreDebug_DHCSR_C_MASKINTS_Pos)          /*!< CoreDebug DHCSR: C_MASKINTS
Mask */

#define CoreDebug_DHCSR_C_STEP_Pos          2
/*!< CoreDebug DHCSR: C_STEP Position */
#define CoreDebug_DHCSR_C_STEP_Msk          (1UL <<
CoreDebug_DHCSR_C_STEP_Pos)             /*!< CoreDebug DHCSR: C_STEP Mask
*/
/* */

#define CoreDebug_DHCSR_C_HALT_Pos          1
/*!< CoreDebug DHCSR: C_HALT Position */
#define CoreDebug_DHCSR_C_HALT_Msk          (1UL <<
CoreDebug_DHCSR_C_HALT_Pos)             /*!< CoreDebug DHCSR: C_HALT Mask
*/
/* */

#define CoreDebug_DHCSR_C_DEBUGEN_Pos       0
/*!< CoreDebug DHCSR: C_DEBUGEN Position */
#define CoreDebug_DHCSR_C_DEBUGEN_Msk       (1UL /*<<
CoreDebug_DHCSR_C_DEBUGEN_Pos*/)        /*!< CoreDebug DHCSR: C_DEBUGEN Mask
*/
/* */

/* Debug Core Register Selector Register */
#define CoreDebug_DCRSR_REGWnR_Pos         16
/*!< CoreDebug DCRSR: REGWnR Position */
#define CoreDebug_DCRSR_REGWnR_Msk         (1UL <<
CoreDebug_DCRSR_REGWnR_Pos)            /*!< CoreDebug DCRSR: REGWnR Mask
*/
/* */

#define CoreDebug_DCRSR_REGSEL_Pos         0
/*!< CoreDebug DCRSR: REGSEL Position */
#define CoreDebug_DCRSR_REGSEL_Msk         (0x1FUL /*<<
CoreDebug_DCRSR_REGSEL_Pos*/)          /*!< CoreDebug DCRSR: REGSEL Mask */

```

```

/* Debug Exception and Monitor Control Register */
#define CoreDebug_DEMCR_TRCENA_Pos           24
/*!< CoreDebug DEMCR: TRCENA Position */
#define CoreDebug_DEMCR_TRCENA_Msk            (1UL <<
CoreDebug_DEMCR_TRCENA_Pos)                  /*!< CoreDebug DEMCR: TRCENA Mask
*/
#define CoreDebug_DEMCR_MON_REQ_Pos           19
/*!< CoreDebug DEMCR: MON_REQ Position */
#define CoreDebug_DEMCR_MON_REQ_Msk            (1UL <<
CoreDebug_DEMCR_MON_REQ_Pos)                  /*!< CoreDebug DEMCR: MON_REQ Mask
*/
#define CoreDebug_DEMCR_MON_STEP_Pos           18
/*!< CoreDebug DEMCR: MON_STEP Position */
#define CoreDebug_DEMCR_MON_STEP_Msk            (1UL <<
CoreDebug_DEMCR_MON_STEP_Pos)                  /*!< CoreDebug DEMCR: MON_STEP
Mask */
#define CoreDebug_DEMCR_MON_PEND_Pos           17
/*!< CoreDebug DEMCR: MON_PEND Position */
#define CoreDebug_DEMCR_MON_PEND_Msk            (1UL <<
CoreDebug_DEMCR_MON_PEND_Pos)                  /*!< CoreDebug DEMCR: MON_PEND
Mask */
#define CoreDebug_DEMCR_MON_EN_Pos             16
/*!< CoreDebug DEMCR: MON_EN Position */
#define CoreDebug_DEMCR_MON_EN_Msk              (1UL <<
CoreDebug_DEMCR_MON_EN_Pos)                  /*!< CoreDebug DEMCR: MON_EN Mask
*/
#define CoreDebug_DEMCR_VC_HARDERR_Pos          10
/*!< CoreDebug DEMCR: VC_HARDERR Position */
#define CoreDebug_DEMCR_VC_HARDERR_Msk           (1UL <<
CoreDebug_DEMCR_VC_HARDERR_Pos)                /*!< CoreDebug DEMCR: VC_HARDERR
Mask */
#define CoreDebug_DEMCR_VC_INTERR_Pos           9
/*!< CoreDebug DEMCR: VC_INTERR Position */
#define CoreDebug_DEMCR_VC_INTERR_Msk            (1UL <<
CoreDebug_DEMCR_VC_INTERR_Pos)                /*!< CoreDebug DEMCR: VC_INTERR
Mask */
#define CoreDebug_DEMCR_VC_BUSERR_Pos           8
/*!< CoreDebug DEMCR: VC_BUSERR Position */
#define CoreDebug_DEMCR_VC_BUSERR_Msk            (1UL <<
CoreDebug_DEMCR_VC_BUSERR_Pos)                /*!< CoreDebug DEMCR: VC_BUSERR
Mask */
#define CoreDebug_DEMCR_VC_STATERR_Pos          7
/*!< CoreDebug DEMCR: VC_STATERR Position */
#define CoreDebug_DEMCR_VC_STATERR_Msk           (1UL <<
CoreDebug_DEMCR_VC_STATERR_Pos)                /*!< CoreDebug DEMCR: VC_STATERR
Mask */

```

```

#define CoreDebug_DEMCR_VC_CHKERR_Pos      6
/*!< CoreDebug DEMCR: VC_CHKERR Position */
#define CoreDebug_DEMCR_VC_CHKERR_Msk      (1UL <<
CoreDebug_DEMCR_VC_CHKERR_Pos)          /*!< CoreDebug DEMCR: VC_CHKERR
Mask */

#define CoreDebug_DEMCR_VC_NOCPERR_Pos     5
/*!< CoreDebug DEMCR: VC_NOCPERR Position */
#define CoreDebug_DEMCR_VC_NOCPERR_Msk     (1UL <<
CoreDebug_DEMCR_VC_NOCPERR_Pos)          /*!< CoreDebug DEMCR: VC_NOCPERR
Mask */

#define CoreDebug_DEMCR_VC_MMERR_Pos       4
/*!< CoreDebug DEMCR: VC_MMERR Position */
#define CoreDebug_DEMCR_VC_MMERR_Msk       (1UL <<
CoreDebug_DEMCR_VC_MMERR_Pos)          /*!< CoreDebug DEMCR: VC_MMERR
Mask */

#define CoreDebug_DEMCR_VC_CORERESET_Pos   0
/*!< CoreDebug DEMCR: VC_CORERESET Position */
#define CoreDebug_DEMCR_VC_CORERESET_Msk   (1UL /*<<
CoreDebug_DEMCR_VC_CORERESET_Pos*/)    /*!< CoreDebug DEMCR: VC_CORERESET
Mask */

/*@} end of group CMSIS_CoreDebug */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_core_base Core Definitions
 \brief Definitions for base addresses, unions, and structures.
 @{
 */

/* Memory mapping of Cortex-M4 Hardware */
#define SCS_BASE           (0xE000E000UL)
/*!< System Control Space Base Address */
#define ITM_BASE            (0xE0000000UL)
/*!< ITM Base Address */
#define DWT_BASE            (0xE0001000UL)
/*!< DWT Base Address */
#define TPI_BASE            (0xE0040000UL)
/*!< TPI Base Address */
#define CoreDebug_BASE       (0xE000EDF0UL)
/*!< Core Debug Base Address */
#define SysTick_BASE         (SCS_BASE + 0x0010UL)
/*!< SysTick Base Address */
#define NVIC_BASE            (SCS_BASE + 0x0100UL)
/*!< NVIC Base Address */
#define SCB_BASE             (SCS_BASE + 0x0D00UL)
/*!< System Control Block Base Address */

#define SCnSCB              ((SCnSCB_Type *) SCS_BASE)
/*!< System control Register not in SCB */

```

```

#define SCB           ((SCB_Type      *) SCB_BASE)
/*!< SCB configuration struct */
#define SysTick       ((SysTick_Type  *) SysTick_BASE)
/*!< SysTick configuration struct */
#define NVIC          ((NVIC_Type    *) NVIC_BASE)
/*!< NVIC configuration struct */
#define ITM           ((ITM_Type     *) ITM_BASE)
/*!< ITM configuration struct */
#define DWT           ((DWT_Type    *) DWT_BASE)
/*!< DWT configuration struct */
#define TPI           ((TPI_Type    *) TPI_BASE)
/*!< TPI configuration struct */
#define CoreDebug     ((CoreDebug_Type *) CoreDebug_BASE)
/*!< Core Debug configuration struct */

#if (__MPU_PRESENT == 1)
#define MPU_BASE      (SCS_BASE + 0x0D90UL)
/*!< Memory Protection Unit */
#define MPU          ((MPU_Type     *) MPU_BASE)
/*!< Memory Protection Unit */
#endif

#if (__FPU_PRESENT == 1)
#define FPU_BASE      (SCS_BASE + 0x0F30UL)
/*!< Floating Point Unit */
#define FPU          ((FPU_Type     *) FPU_BASE)
/*!< Floating Point Unit */
#endif

/*@} */

/********************* Hardware Abstraction Layer
Core Function Interface contains:
- Core NVIC Functions
- Core SysTick Functions
- Core Debug Functions
- Core Register Access Functions
****/



/** \defgroup CMSIS_Core_FunctionInterface Functions and Instructions Reference
*/
/* ##### NVIC functions
#####
\ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions NVIC Functions

```

```

        \brief      Functions that manage interrupts and exceptions via the
NVIC.
    @{
}

/** \brief Set Priority Grouping

The function sets the priority grouping field using the required unlock
sequence.
The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8]
PRIGROUP field.
Only values from 0..7 are used.
In case of a conflict between priority grouping and available
priority bits (__NVIC_PRIO_BITS), the smallest possible priority group
is set.

    \param [in]      PriorityGroup  Priority grouping field.
*/
STATIC_INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
    uint32_t reg_value;
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL);
/* only values 0..7 are used */

    reg_value = SCB->AIRCR;
/* read old register configuration */
    reg_value &= ~((uint32_t)(SCB_AIRCR_VECTKEY_Msk |
SCB_AIRCR_PRIGROUP_Msk)); /* clear bits to change */
/*
    reg_value = (reg_value
                  | ((uint32_t)0x5FAUL << SCB_AIRCR_VECTKEY_Pos)
                  | (PriorityGroupTmp << 8));
/* Insert write key and priority group */
    SCB->AIRCR = reg_value;
}

/** \brief Get Priority Grouping

The function reads the priority grouping field from the NVIC Interrupt
Controller.

    \return          Priority grouping field (SCB->AIRCR [10:8]
PRIGROUP field).
*/
STATIC_INLINE uint32_t NVIC_GetPriorityGrouping(void)
{
    return ((uint32_t)((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >>
SCB_AIRCR_PRIGROUP_Pos));
}

/** \brief Enable External Interrupt

```

The function enables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]      IRQn  External interrupt number. Value cannot be
negative.
*/
STATIC_INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
    NVIC->ISER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL <<
((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

/\*\* \brief Disable External Interrupt

The function disables a device-specific interrupt in the NVIC interrupt controller.

```
\param [in]      IRQn  External interrupt number. Value cannot be
negative.
*/
STATIC_INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
{
    NVIC->ICER[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL <<
((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
```

/\*\* \brief Get Pending Interrupt

The function reads the pending register in the NVIC and returns the pending bit  
for the specified interrupt.

```
\param [in]      IRQn  Interrupt number.

\return          0  Interrupt status is not pending.
\return          1  Interrupt status is pending.
*/
STATIC_INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
{
    return((uint32_t)((NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL]) &
(1UL << ((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0UL);
}
```

/\*\* \brief Set Pending Interrupt

The function sets the pending bit of an external interrupt.

```
\param [in]      IRQn  Interrupt number. Value cannot be negative.
*/
STATIC_INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)
{
```

```

NVIC->ISPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL <<
((uint32_t)(int32_t)IRQn) & 0x1FUL));
}

/** \brief Clear Pending Interrupt

The function clears the pending bit of an external interrupt.

\param [in] IRQn External interrupt number. Value cannot be negative.
*/
STATIC_INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ICPR[((uint32_t)(int32_t)IRQn) >> 5UL] = (uint32_t)(1UL <<
((uint32_t)(int32_t)IRQn) & 0x1FUL));
}

/** \brief Get Active Interrupt

The function reads the active register in NVIC and returns the active bit.

\param [in] IRQn Interrupt number.

\return 0 Interrupt status is not active.
\return 1 Interrupt status is active.
*/
STATIC_INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
{
    return((uint32_t)((NVIC->IABR[((uint32_t)(int32_t)IRQn) >> 5UL]) &
(1UL << ((uint32_t)(int32_t)IRQn) & 0x1FUL))) != 0UL) ? 1UL : 0UL);
}

/** \brief Set Interrupt Priority

The function sets the priority of an interrupt.

\note The priority cannot be set for every core interrupt.

\param [in] IRQn Interrupt number.
\param [in] priority Priority to set.
*/
STATIC_INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
    if((int32_t)IRQn < 0) {
        SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL] =
(uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL);
    }
    else {
        NVIC->IP[((uint32_t)(int32_t)IRQn)] =
(uint8_t)((priority << (8 - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL);
    }
}

```

```

    }

}

/** \brief Get Interrupt Priority

The function reads the priority of an interrupt. The interrupt
number can be positive to specify an external (device specific)
interrupt, or negative to specify an internal (core) interrupt.

\param [in]    IRQn  Interrupt number.
\return         Interrupt Priority. Value is aligned
automatically to the implemented
                           priority bits of the microcontroller.
*/
STATIC_INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{
    if((int32_t)IRQn < 0) {
        return(((uint32_t)SCB->SHP[((uint32_t)(int32_t)IRQn) & 0xFUL]-4UL]
>> (8 - __NVIC_PRIO_BITS));
    }
    else {
        return(((uint32_t)NVIC->IP[((uint32_t)(int32_t)IRQn)]
>> (8 - __NVIC_PRIO_BITS)));
    }
}

/** \brief Encode Priority

The function encodes the priority for an interrupt with the given
priority group,
preemptive priority value, and subpriority value.
In case of a conflict between priority grouping and available
priority bits (__NVIC_PRIO_BITS), the smallest possible priority
group is set.

\param [in]      PriorityGroup  Used priority group.
\param [in]      PreemptPriority  Preemptive priority value (starting
from 0).
\param [in]      SubPriority   Subpriority value (starting from 0).
\return          Encoded priority. Value can be used in
the function \ref NVIC_SetPriority().
*/
STATIC_INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup,
uint32_t PreemptPriority, uint32_t SubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t)0x07UL); /**
only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;
}

```

```

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) >
    (uint32_t) (__NVIC_PRIO_BITS)) ? (uint32_t) (__NVIC_PRIO_BITS) :
    (uint32_t) (7UL - PriorityGroupTmp);
    SubPriorityBits      = ((PriorityGroupTmp +
    (uint32_t) (__NVIC_PRIO_BITS)) < (uint32_t) 7UL) ? (uint32_t) 0UL :
    (uint32_t) ((PriorityGroupTmp - 7UL) + (uint32_t) (__NVIC_PRIO_BITS));

    return (
        ((PreemptPriority & (uint32_t) ((1UL << (PreemptPriorityBits))
- 1UL)) << SubPriorityBits) |
        ((SubPriority      & (uint32_t) ((1UL << (SubPriorityBits
- 1UL)))) |
    );
}

```

/\*\* \brief Decode Priority

The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value. In case of a conflict between priority grouping and available priority bits (NVIC\_PRIO\_BITS) the smallest possible priority group is set.

```

\param [in]          Priority  Priority value, which can be retrieved
with the function \ref NVIC_GetPriority().
\param [in]          PriorityGroup  Used priority group.
\param [out] pPreemptPriority  Preemptive priority value (starting
from 0).
\param [out]          pSubPriority  Subpriority value (starting from 0).
*/
STATIC_INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t
PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & (uint32_t) 0x07UL); /**
only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7UL - PriorityGroupTmp) >
    (uint32_t) (__NVIC_PRIO_BITS)) ? (uint32_t) (__NVIC_PRIO_BITS) :
    (uint32_t) (7UL - PriorityGroupTmp);
    SubPriorityBits      = ((PriorityGroupTmp +
    (uint32_t) (__NVIC_PRIO_BITS)) < (uint32_t) 7UL) ? (uint32_t) 0UL :
    (uint32_t) ((PriorityGroupTmp - 7UL) + (uint32_t) (__NVIC_PRIO_BITS));

    *pPreemptPriority = (Priority >> SubPriorityBits) & (uint32_t) ((1UL <<
(PreemptPriorityBits)) - 1UL);
    *pSubPriority      = (Priority                  ) & (uint32_t) ((1UL <<
(SubPriorityBits      )) - 1UL);
}

```

```

/** \brief System Reset

    The function initiates a system reset request to reset the MCU.
*/
STATIC_INLINE void NVIC_SystemReset(void)
{
    __DSB();                                         /* Ensure all outstanding memory accesses included

buffered write are completed before reset */
    SCB->AIRCR = (uint32_t)((0x5FAUL << SCB_AIRCR_VECTKEY_Pos) |
                           (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
                           SCB_AIRCR_SYSRESETREQ_Msk);           /* Keep priority group unchanged */
    __DSB();                                         /* Ensure completion of memory access */
    while(1) { __NOP(); }                            /* wait until reset */
}
/*@} end of CMSIS_Core_NVICFunctions */

```

```

/* ##### SysTick function
#####
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_SysTickFunctions SysTick Functions
\brief Functions that configure the System.
{@{
*/
#endif

#if (__Vendor_SysTickConfig == 0)

/** \brief System Tick Configuration

    The function initializes the System Timer and its interrupt, and
starts the System Tick Timer.
    Counter is in free running mode to generate periodic interrupts.

\param [in] ticks Number of ticks between two interrupts.

\return 0 Function succeeded.
\return 1 Function failed.

\note When the variable <b>__Vendor_SysTickConfig</b> is set to
1, then the
    function <b>SysTick_Config</b> is not included. In this case, the
file <b><i>device</i>.h</b>
    must contain a vendor-specific implementation of this function.

*/
STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{

```

```

    if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk) { return (1UL); }      /*
Reload value impossible */

    SysTick->LOAD  = (uint32_t)(ticks - 1UL);                                /*
set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /*

set Priority for Systick Interrupt */
    SysTick->VAL   = 0UL;                                                 /*
Load the SysTick Counter Value */
    SysTick->CTRL  = SysTick_CTRL_CLKSOURCE_Msk |                         /*
                    SysTick_CTRL_TICKINT_Msk |                         /*
                    SysTick_CTRL_ENABLE_Msk;                           /*
Enable SysTick IRQ and SysTick Timer */
    return (0UL);                                                       /*
Function successful */
}

#endif

/*@} end of CMSIS_Core_SysTickFunctions */

```

```

{
    while (ITM->PORT[0].u32 == 0UL) { __NOP(); }
    ITM->PORT[0].u8 = (uint8_t)ch;
}
return (ch);
}

/** \brief  ITM Receive Character

The function inputs a character via the external variable \ref
ITM_RxBuffer.

\return Received character.
\return -1 No character pending.
*/
STATIC_INLINE int32_t ITM_ReceiveChar (void) {
int32_t ch = -1; /* no character available */

if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
    ch = ITM_RxBuffer;
    ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character
*/
}

return (ch);
}

/** \brief  ITM Check Character

The function checks whether a character is pending for reading in the
variable \ref ITM_RxBuffer.

\return 0 No character available.
\return 1 Character available.
*/
STATIC_INLINE int32_t ITM_CheckChar (void) {

if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) { /* no character available
*/
}
else {
    return (1); /* character available
*/
}
}

/*@} end of CMSIS_core_DebugFunctions */

```

```
#ifdef __cplusplus
```

```

}

#endif

#endif /* __CORE_CM4_H_DEPENDANT */

#endif /* __CMSIS_GENERIC */
*****/*
** /**
* @file core_cmSimd.h
* @brief CMSIS Cortex-M SIMD Header File
* @version V4.10
* @date 18. March 2015
*
* @note
*

*****
*/ Copyright (c) 2009 - 2014 ARM LIMITED

```

All rights reserved.  
 Redistribution and use in source and binary forms, with or without  
 modification, are permitted provided that the following conditions are  
 met:

- Redistributions of source code must retain the above copyright  
 notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright  
 notice, this list of conditions and the following disclaimer in the  
 documentation and/or other materials provided with the distribution.
- Neither the name of ARM nor the names of its contributors may be  
 used  
 to endorse or promote products derived from this software without  
 specific prior written permission.

\*

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS  
 "AS IS"  
 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,  
 THE  
 IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR  
 PURPOSE  
 ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS  
 BE  
 LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR  
 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF  
 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR  
 BUSINESS  
 INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER  
 IN  
 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR  
 OTHERWISE)  
 ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF  
 THE  
 POSSIBILITY OF SUCH DAMAGE.

```

-----*/
-----*/



#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA
check */
#endif

#ifndef __CORE_CMSIMD_H
#define __CORE_CMSIMD_H

#ifdef __cplusplus
extern "C" {
#endif


/********************* Hardware Abstraction Layer ******************/


/* ##### Compiler specific Intrinsics
#####
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
   Access to dedicated SIMD instructions
   @{
*/
/*if defined ( __CC_ARM ) /*-----RealView Compiler -----
-----*/
/* ARM armcc specific functions */
#define __SADD8           __sadd8
#define __QADD8           __qadd8
#define __SHADD8          __shadd8
#define __UADD8           __uadd8
#define __UQADD8          __uqadd8
#define __UHADD8          __uhadd8
#define __SSUB8           __ssub8
#define __QSUB8           __qsub8
#define __SHSUB8          __shsub8
#define __USUB8           __usub8
#define __UQSUB8          __uqsub8
#define __UHSUB8          __uhsub8
#define __SADD16          __sadd16
#define __QADD16          __qadd16
#define __SHADD16         __shadd16
#define __UADD16          __uadd16
#define __UQADD16         __uqadd16
#define __UHADD16         __uhadd16
#define __SSUB16          __ssub16

```

```

#define __QSUB16          __qsub16
#define __SHSUB16         __shsub16
#define __USUB16          __usub16
#define __UQSUB16         __uqsub16
#define __UHSUB16         __uhsub16
#define __SASX            __sasx
#define __QASX            __qasx
#define __SHASX           __shasx
#define __UASX            __uasx
#define __UQASX           __uqasx
#define __UHASX           __uhasx
#define __SSAX             __ssax
#define __QSAX             __qsax
#define __SHSAX            __shsax
#define __USAX             __usax
#define __UQSAX            __uqsax
#define __UHSAX            __uhsax
#define __USAD8            __usad8
#define __USADA8           __usada8
#define __SSAT16           __ssat16
#define __USAT16           __usat16
#define __UXTB16           __uxtb16
#define __UXTAB16          __uxtab16
#define __SXTB16           __sxtb16
#define __SXTAB16          __sxtab16
#define __SMUAD            __smuad
#define __SMUADX           __smuadx
#define __SMLAD            __smlad
#define __SMLADX           __smladx
#define __SMLALD           __smlald
#define __SMLALDX          __smlalidx
#define __SMUSD             __smusd
#define __SMUSDX           __smusdx
#define __SMLSD             __smlsd
#define __SMLSIDX          __smlsdx
#define __SMLSLD            __smlsld
#define __SMLSLIDX          __smlsldidx
#define __SEL               __sel
#define __QADD              __qadd
#define __QSUB              __qsub

#define __PKHBT(ARG1,ARG2,ARG3)          (((uint32_t)(ARG1)) \
) & 0x0000FFFFUL) | \
(ARG3) & 0xFFFFF0000UL)    )

#define __PKHTB(ARG1,ARG2,ARG3)          (((uint32_t)(ARG1)) \
) & 0xFFFFF0000UL) | \
(ARG3) & 0x0000FFFFUL)    )

#define __SMMLA(ARG1,ARG2,ARG3)          ( (int32_t) (((int64_t)(ARG1) * \
(ARG2)) + \

```

```

((int64_t) (ARG3) <<
32)      ) >> 32))

#endif /* !defined __ASSEMBLY__ */

#ifndef __ASSEMBLY__
#elif defined ( __GNUC__ ) /*----- GNU Compiler -----
-----*/
/* GNU gcc specific functions */
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
}

```

```

        return(result);
    }

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__SSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__QSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__SHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__UQSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__QADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__UADD16(uint32_t op1, uint32_t op2)
{

```

```

    uint32_t result;

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
}

```

```

    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UQSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
SHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhaxs %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

```

```

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__USAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__USAD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__USADA8(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3));
    return(result);
}

#define __SSAT16(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)) \
); \
    __RES; \
})

#define __USAT16(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1)) \
); \
    __RES; \
})

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2));
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__SXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

```

```

}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t
__SXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SMUAD
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( __always_inline ) __STATIC_INLINE uint32_t __SMUADX
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

attribute( __always_inline ) __STATIC_INLINE uint32_t __SMLAD
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r"
(op2), "r" (op3) );
    return(result);
}

attribute( __always_inline ) __STATIC_INLINE uint32_t __SMLADX
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3) );
    return(result);
}

attribute( __always_inline ) __STATIC_INLINE uint64_t __SMLALD
(uint32_t op1, uint32_t op2, uint64_t acc)

```

```

{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1]))
);
#else // Big endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0]))
);
#endif

    return(llr.w64);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint64_t __SMLALDX
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1]))
);
#else // Big endian
    __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0]))
);
#endif

    return(llr.w64);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUSD
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r"
    (op2));
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUSDX
(uint32_t op1, uint32_t op2)

```

```

{
    uint32_t result;

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t __SMLSD
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r"
(op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t __SMLSDX
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint64_t __SMLSLD
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__      // Little endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
(llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else                  // Big endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
(llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

    return(llr.w64);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint64_t __SMLSLDX
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{

```

```

        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__      // Little endian
    __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else                  // Big endian
    __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

        return(llr.w64);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SEL
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2)
);
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__QADD(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( always_inline )) __STATIC_INLINE uint32_t
__QSUB(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

#define __PKHBT(ARG1,ARG2,ARG3) \
({ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    __ASM ("pkhbt %0,%1,%2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r" \
(__ARG2), "I" (ARG3) ); \
    __RES; \
}

```

```

    })

#define __PKHTB(ARG1,ARG2,ARG3) \
({ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" \
(__ARG2)); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" \
(__ARG2), "I" (ARG3)); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMMLA
(int32_t op1, int32_t op2, int32_t op3)
{
    int32_t result;

    __ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" \
(op2), "r" (op3));
    return(result);
}

#if defined ( __ICCARM__ ) /*----- ICC Compiler -----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#if defined ( __TMS470__ ) /*----- TI CCS Compiler -----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#if defined ( __TASKING__ ) /*----- TASKING Compiler -----*/
/* TASKING carm specific functions */
/* not yet supported */

#if defined ( __CSMC__ ) /*----- COSMIC Compiler -----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD intrinsics */

#endif

```

```

}

#endif

#endif /* __CORE_CMSIMD_H */
//*********************************************************************
**/ /**
 * @file      core_cm4_simd.h
 * @brief     CMSIS Cortex-M4 SIMD Header File
 * @version   V3.30
 * @date      17. February 2014
 *
 * @note
 *

*****/
/* Copyright (c) 2009 - 2014 ARM LIMITED

   All rights reserved.
   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
met:
   - Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.
   - Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
   - Neither the name of ARM nor the names of its contributors may be
used
     to endorse or promote products derived from this software without
     specific prior written permission.
   *
   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS"
   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
   ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS
BE
   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN
   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE)
   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE
   POSSIBILITY OF SUCH DAMAGE.
-----*/

```

```

#if defined ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA
check */
#endif

#ifndef __CORE_CM4 SIMD_H
#define __CORE_CM4 SIMD_H

#ifdef __cplusplus
extern "C" {
#endif


/********************* Hardware Abstraction Layer *****************/
***** Hardware Abstraction Layer *****

***** /



/* ##### Compiler specific Intrinsics
#####
/** \defgroup CMSIS SIMD_intrinsics CMSIS SIMD Intrinsics
Access to dedicated SIMD instructions
@{
*/
#if defined ( __CC_ARM ) /*-----RealView Compiler -----*/
/* ARM armcc specific functions */
#define __SADD8           sadd8
#define __QADD8           qadd8
#define __SHADD8          shadd8
#define __UADD8           uadd8
#define __UQADD8          uqadd8
#define __UHADD8          uhadd8
#define __SSUB8            ssub8
#define __QSUB8            qsub8
#define __SHSUB8           shsub8
#define __USUB8            usub8
#define __UQSUB8           uqsub8
#define __UHSUB8           uhsub8
#define __SADD16          sadd16
#define __QADD16          qadd16
#define __SHADD16          shadd16
#define __UADD16          uadd16
#define __UQADD16          uqadd16
#define __UHADD16          uhadd16
#define __SSUB16           ssub16
#define __QSUB16           qsub16
#define __SHSUB16          shsub16
#define __USUB16           usub16

```

```

#define __UQSUB16          __uqsub16
#define __UHSUB16          __uhsub16
#define __SASX              __sasx
#define __QASX              __qasx
#define __SHASX             __shasx
#define __UASX              __uasx
#define __UQASX             __uqasx
#define __UHASX             __uhasx
#define __SSAX              __ssax
#define __QSAX              __qsax
#define __SHSAX             __shsax
#define __USAX              __usax
#define __UQSAX             __uqsax
#define __UHSAX             __uhsax
#define __USAD8             __usad8
#define __USADA8            __usada8
#define __SSAT16             __ssat16
#define __USAT16             __usat16
#define __UXTB16             __uxtb16
#define __UXTAB16            __uxtab16
#define __SXTB16             __sxtb16
#define __SXTAB16            __sxtab16
#define __SMUAD              __smuad
#define __SMUADX             __smuadx
#define __SMLAD              __smlad
#define __SMLADX             __smladx
#define __SMLALD             __smlald
#define __SMLALDX            __smlalidx
#define __SMUSD              __smusd
#define __SMUSDX             __smusdx
#define __SMLSD              __smlsd
#define __SMLSIDX            __smlsdx
#define __SMLSLD              __smlsld
#define __SMLSIDX             __smlsldidx
#define __SEL                __sel
#define __QADD               __qadd
#define __QSUB               __qsub

#define __PKHBT(ARG1,ARG2,ARG3)          ( (((uint32_t)(ARG1)) \
) & 0x0000FFFFUL) | \
(ARG3) & 0xFFFFF0000UL) )

#define __PKHTB(ARG1,ARG2,ARG3)          ( (((uint32_t)(ARG1)) \
) & 0xFFFFF0000UL) | \
(ARG3) & 0x0000FFFFUL) )

#define __SMMLA(ARG1,ARG2,ARG3)          ( (int32_t) (((int64_t)(ARG1) * \
(ARG2)) + \
32) ) >> 32)

```

```

#elif defined ( __GNUC__ ) /*----- GNU Compiler -----
-----*/
/* GNU gcc specific functions */
__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__UHADD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__SSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__QSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__SHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__USUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) STATIC_INLINE uint32_t
__UQSUB8(uint32_t op1, uint32_t op2)
{

```

```

    uint32_t result;

    __ASM volatile ("uqsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
UHSUB8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
SADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
QADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
SHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) STATIC_INLINE uint32_t
UADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;
}

```

```

    __ASM volatile ("uadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UHADD16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhadd16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__USUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__UQSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__UHSUB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsub16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__SASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__QASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__((always_inline)) __STATIC_INLINE uint32_t
__SHASX(uint32_t op1, uint32_t op2)
{

```

```

    uint32_t result;

    __ASM volatile ("shasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UHASX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhasx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("ssax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
}

```

```

        return(result);
    }

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__SHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("shsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__USAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UQSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uqsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__UHSAX(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uhsax %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__USAD8(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("usad8 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

```

```

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__USADA8(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("usada8 %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3));
    return(result);
}

#define __SSAT16(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("ssat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1) \
); \
    __RES; \
})

#define __USAT16(ARG1,ARG2) \
({ \
    uint32_t __RES, __ARG1 = (ARG1); \
    __ASM ("usat16 %0, %1, %2" : "=r" (__RES) : "I" (ARG2), "r" (__ARG1) \
); \
    __RES; \
})

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("uxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__UXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("uxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2));
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t
__SXTB16(uint32_t op1)
{
    uint32_t result;

    __ASM volatile ("sxtb16 %0, %1" : "=r" (result) : "r" (op1));
    return(result);
}

```

```

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SXTAB16(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sxtab16 %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUAD
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuad %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUADX
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smuadx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMLAD
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlad %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r"
(op2), "r" (op3) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMLADX
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smladx %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint64_t __SMLALD
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{

```

```

        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__      // Little endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else                  // Big endian
    __ASM volatile ("smlald %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

        return(llr.w64);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint64_t __SMLALDX
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__      // Little endian
    __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else                  // Big endian
    __ASM volatile ("smlaldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

        return(llr.w64);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUSD
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("smusd %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__(( __always_inline )) __STATIC_INLINE uint32_t __SMUSDX
(uint32_t op1, uint32_t op2)
{
    uint32_t result;
}

```

```

    __ASM volatile ("smusdx %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSD
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsd %0, %1, %2, %3" : "=r" (result) : "r" (op1), "r"
(op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SMLSDX
(uint32_t op1, uint32_t op2, uint32_t op3)
{
    uint32_t result;

    __ASM volatile ("smlsdx %0, %1, %2, %3" : "=r" (result) : "r" (op1),
"r" (op2), "r" (op3) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSLD
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__ // Little endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
(llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else // Big endian
    __ASM volatile ("smlsld %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
(llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

    return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint64_t __SMLSLDX
(uint32_t op1, uint32_t op2, uint64_t acc)
{
    union llreg_u{
        uint32_t w32[2];
        uint64_t w64;
    } llr;
    llr.w64 = acc;
}

```

```

    } llr;
    llr.w64 = acc;

#ifndef __ARMEB__      // Little endian
    __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[0]), "=r"
    (llr.w32[1]): "r" (op1), "r" (op2) , "0" (llr.w32[0]), "1" (llr.w32[1])
);
#else                  // Big endian
    __ASM volatile ("smlsldx %0, %1, %2, %3" : "=r" (llr.w32[1]), "=r"
    (llr.w32[0]): "r" (op1), "r" (op2) , "0" (llr.w32[1]), "1" (llr.w32[0])
);
#endif

    return(llr.w64);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t __SEL
(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("sel %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2)
);
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QADD(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qadd %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

__attribute__( ( always_inline ) ) __STATIC_INLINE uint32_t
__QSUB(uint32_t op1, uint32_t op2)
{
    uint32_t result;

    __ASM volatile ("qsub %0, %1, %2" : "=r" (result) : "r" (op1), "r"
(op2) );
    return(result);
}

#define __PKHBT(ARG1,ARG2,ARG3) \
({ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    __ASM ("pkhbt %0, %1, %2, lsl %3" : "=r" (__RES) : "r" (__ARG1), "r"
(__ARG2), "I" (ARG3) ); \
    __RES; \
})

```

```

#define __PKHTB(ARG1,ARG2,ARG3) \
({ \
    uint32_t __RES, __ARG1 = (ARG1), __ARG2 = (ARG2); \
    if (ARG3 == 0) \
        __ASM ("pkhtb %0, %1, %2" : "=r" (__RES) : "r" (__ARG1), "r" \
(__ARG2) ); \
    else \
        __ASM ("pkhtb %0, %1, %2, asr %3" : "=r" (__RES) : "r" (__ARG1), "r" \
(__ARG2), "I" (ARG3) ); \
    __RES; \
})

__attribute__(( always_inline )) __STATIC_INLINE uint32_t __SMMLA
(int32_t op1, int32_t op2, int32_t op3)
{
    int32_t result;

    __ASM volatile ("smmla %0, %1, %2, %3" : "=r" (result): "r" (op1), "r" \
(op2), "r" (op3) );
    return(result);
}

#elif defined ( __ICCARM__ ) /*----- ICC Compiler -----\
-----*/
/* IAR iccarm specific functions */
#include <cmsis_iar.h>

#elif defined ( __TMS470__ ) /*----- TI CCS Compiler -----\
-----*/
/* TI CCS specific functions */
#include <cmsis_ccs.h>

#elif defined ( __TASKING__ ) /*----- TASKING Compiler -----\
-----*/
/* TASKING arm specific functions */
/* not yet supported */

#elif defined ( __CSMC__ ) /*----- COSMIC Compiler -----\
-----*/
/* Cosmic specific functions */
#include <cmsis_csm.h>

#endif

/*@} end of group CMSIS SIMD_intrinsics */

#ifndef __cplusplus
}
#endif

```

```
#endif /* __CORE_CM4 SIMD_H */
```