

```

/*
 * Copyright 2008 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/

```

```

#ifndef CMOCKA_H_
#define CMOCKA_H_

```

```

#include <stdarg.h>
#include <stdlib.h>
#include <setjmp.h>

```

```

#ifdef _WIN32
# ifdef _MSC_VER

```

```

#define __func__ __FUNCTION__

```

```

# ifndef inline
#define inline __inline
# endif /* inline */

```

```

# if _MSC_VER < 1500
#  ifdef __cplusplus
extern "C" {
#   endif /* __cplusplus */
int __stdcall IsDebuggerPresent();
#   ifdef __cplusplus
} /* extern "C" */
#   endif /* __cplusplus */
#  endif /* _MSC_VER < 1500 */
# endif /* _MSC_VER */
#endif /* _WIN32 */

```

```

/**
 * @defgroup cmocka The CMocka API
 *
 * These headers or their equivalents should be included prior to including
 * this header file.
 * @code
 * #include <stdarg.h>
 * #include <stddef.h>
 * #include <setjmp.h>

```

```

* @endcode
*
* This allows test applications to use custom definitions of C standard
* library functions and types.
*
* @{
*/

/* If __WORDSIZE is not set, try to figure it out and default to 32 bit. */
#ifndef __WORDSIZE
# if defined(__x86_64__) && !defined(__ILP32__)
#  define __WORDSIZE 64
# else
#  define __WORDSIZE 32
# endif
#endif

#ifndef DOXYGEN
/**
 * Largest integral type. This type should be large enough to hold any
 * pointer or integer supported by the compiler.
 */
typedef uintmax_t LargestIntegralType;
#else /* DOXYGEN */
#ifndef LargestIntegralType
# if __WORDSIZE == 64
#  define LargestIntegralType unsigned long int
# else
#  define LargestIntegralType unsigned long long int
# endif
#endif /* LargestIntegralType */
#endif /* DOXYGEN */

/* Printf format used to display LargestIntegralType as a hexadecimal. */
#ifndef LargestIntegralTypePrintfFormat
# ifdef _WIN32
#  define LargestIntegralTypePrintfFormat "0x%I64x"
# else
#  if __WORDSIZE == 64
#   define LargestIntegralTypePrintfFormat "%#lx"
#  else
#   define LargestIntegralTypePrintfFormat "%#llx"
#  endif
# endif /* _WIN32 */
#endif /* LargestIntegralTypePrintfFormat */

/* Printf format used to display LargestIntegralType as a decimal. */
#ifndef LargestIntegralTypePrintfFormatDecimal
# ifdef _WIN32
#  define LargestIntegralTypePrintfFormatDecimal "%I64u"
# else
#  if __WORDSIZE == 64

```

```

# define LargestIntegralTypePrintfFormatDecimal "%lu"
# else
# define LargestIntegralTypePrintfFormatDecimal "%llu"
# endif
# endif /* _WIN32 */
#endif /* LargestIntegralTypePrintfFormat */

/* Perform an unsigned cast to LargestIntegralType. */
#define cast_to_largest_integral_type(value) \
    ((LargestIntegralType)(value))

/* Smallest integral type capable of holding a pointer. */
#if !defined(_UINTPTR_T) && !defined(_UINTPTR_T_DEFINED)
# if defined(_WIN32)
    /* WIN32 is an ILP32 platform */
    typedef unsigned int uintptr_t;
# elif defined(_WIN64)
    typedef unsigned long int uintptr_t
# else /* _WIN32 */

/* ILP32 and LP64 platforms */
# ifdef __WORDSIZE /* glibc */
# if __WORDSIZE == 64
    typedef unsigned long int uintptr_t;
# else
    typedef unsigned int uintptr_t;
# endif /* __WORDSIZE == 64 */
# else /* __WORDSIZE */
# if defined(_LP64) || defined(_I32LPx)
    typedef unsigned long int uintptr_t;
# else
    typedef unsigned int uintptr_t;
# endif
# endif /* __WORDSIZE */
# endif /* _WIN32 */

# define _UINTPTR_T
# define _UINTPTR_T_DEFINED
#endif /* !defined(_UINTPTR_T) || !defined(_UINTPTR_T_DEFINED) */

/* Perform an unsigned cast to uintptr_t. */
#define cast_to_pointer_integral_type(value) \
    ((uintptr_t)((size_t)(value)))

/* Perform a cast of a pointer to LargestIntegralType */
#define cast_ptr_to_largest_integral_type(value) \
    cast_to_largest_integral_type(cast_to_pointer_integral_type(value))

/* GCC have printf type attribute check. */
#ifdef __GNUC__
#define CMOCKA_PRINTF_ATTRIBUTE(a,b) \
    __attribute__((__format__(__printf__, a, b)))

```

```

#else
#define CMOCKA_PRINTF_ATTRIBUTE(a,b)
#endif /* __GNUC__ */

#if defined(__GNUC__)
#define CMOCKA_DEPRECATED __attribute__((deprecated))
#elif defined(_MSC_VER)
#define CMOCKA_DEPRECATED __declspec(deprecated)
#else
#define CMOCKA_DEPRECATED
#endif

#define WILL_RETURN_ALWAYS -1
#define WILL_RETURN_ONCE -2

/**
 * @defgroup cmocka_mock Mock Objects
 * @ingroup cmocka
 *
 * Mock objects mock objects are simulated objects that mimic the behavior of
 * real objects. Instead of calling the real objects, the tested object calls a
 * mock object that merely asserts that the correct methods were called, with
 * the expected parameters, in the correct order.
 *
 * <ul>
 * <li><strong>will_return(function, value)</strong> - The will_return() macro
 * pushes a value onto a stack of mock values. This macro is intended to be
 * used by the unit test itself, while programming the behaviour of the mocked
 * object.</li>
 *
 * <li><strong>mock()</strong> - the mock macro pops a value from a stack of
 * test values. The user of the mock() macro is the mocked object that uses it
 * to learn how it should behave.</li>
 *
 * </ul>
 *
 * Because the will_return() and mock() are intended to be used in pairs, the
 * cmocka library would fail the test if there are more values pushed onto the
 * stack using will_return() than consumed with mock() and vice-versa.
 *
 * The following unit test stub illustrates how would a unit test instruct the
 * mock object to return a particular value:
 *
 * @code
 * will_return(chef_cook, "hotdog");
 * will_return(chef_cook, 0);
 * @endcode
 *
 * Now the mock object can check if the parameter it received is the parameter
 * which is expected by the test driver. This can be done the following way:
 *
 * @code
 * int chef_cook(const char *order, char **dish_out)

```

```

* {
*   check_expected(order);
* }
* @endcode
*
* For a complete example please at a look
* <a href="http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef_wrap/waiter_test_wrap.c">here</a>.
*
* @{
*/

```

```

#ifdef DOXYGEN
/**
* @brief Retrieve a return value of the current function.
*
* @return The value which was stored to return by this function.
*
* @see will_return()
*/

```

```

LargestIntegralType mock(void);
#else
#define mock() _mock(__func__, __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
* @brief Retrieve a typed return value of the current function.
*
* The value would be casted to type internally to avoid having the
* caller to do the cast manually.
*
* @param[in] #type The expected type of the return value
*
* @return The value which was stored to return by this function.
*
* @code
* int param;
*
* param = mock_type(int);
* @endcode
*
* @see will_return()
* @see mock()
* @see mock_ptr_type()
*/
#type mock_type(#type);
#else
#define mock_type(type) ((type) mock())
#endif

```

```

#ifdef DOXYGEN
/**

```

```

* @brief Retrieve a typed return value of the current function.
*
* The value would be casted to type internally to avoid having the
* caller to do the cast manually but also casted to uintptr_t to make
* sure the result has a valid size to be used as a pointer.
*
* @param[in] #type The expected type of the return value
*
* @return The value which was stored to return by this function.
*
* @code
* char *param;
*
* param = mock_ptr_type(char *);
* @endcode
*
* @see will_return()
* @see mock()
* @see mock_type()
*/
type mock_ptr_type(#type);
#else
#define mock_ptr_type(type) ((type) (uintptr_t) mock())
#endif

```

```

#ifdef DOXYGEN

```

```

/**
* @brief Store a value to be returned by mock() later.
*
* @param[in] #function The function which should return the given value.
*
* @param[in] value The value to be returned by mock().
*
* @code
* int return_integer(void)
* {
*     return (int)mock();
* }
*
* static void test_integer_return(void **state)
* {
*     will_return(return_integer, 42);
*
*     assert_int_equal(my_function_calling_return_integer(), 42);
* }
* @endcode
*
* @see mock()
* @see will_return_count()
*/
void will_return(#function, LargestIntegralType value);

```

```

#else
#define will_return(function, value) \
    _will_return(#function, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), 1)
#endif

#ifdef DOXYGEN
/**
 * @brief Store a value to be returned by mock() later.
 *
 * @param[in] #function The function which should return the given value.
 *
 * @param[in] value The value to be returned by mock().
 *
 * @param[in] count The parameter indicates the number of times the value should
 *                  be returned by mock(). If count is set to -1, the value
 *                  will always be returned but must be returned at least once.
 *                  If count is set to -2, the value will always be returned
 *                  by mock(), but is not required to be returned.
 *
 * @see mock()
 */
void will_return_count(#function, LargestIntegralType value, int count);
#else
#define will_return_count(function, value, count) \
    _will_return(#function, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)
#endif

#ifdef DOXYGEN
/**
 * @brief Store a value that will be always returned by mock().
 *
 * @param[in] #function The function which should return the given value.
 *
 * @param[in] #value The value to be returned by mock().
 *
 * This is equivalent to:
 * @code
 * will_return_count(function, value, -1);
 * @endcode
 *
 * @see will_return_count()
 * @see mock()
 */
void will_return_always(#function, LargestIntegralType value);
#else
#define will_return_always(function, value) \
    will_return_count(function, (value), WILL_RETURN_ALWAYS)
#endif

#ifdef DOXYGEN

```

```

/**
 * @brief Store a value that may be always returned by mock().
 *
 * This stores a value which will always be returned by mock() but is not
 * required to be returned by at least one call to mock(). Therefore,
 * in contrast to will_return_always() which causes a test failure if it
 * is not returned at least once, will_return_maybe() will never cause a test
 * to fail if its value is not returned.
 *
 * @param[in] #function The function which should return the given value.
 *
 * @param[in] #value The value to be returned by mock().
 *
 * This is equivalent to:
 * @code
 * will_return_count(function, value, -2);
 * @endcode
 *
 * @see will_return_count()
 * @see mock()
 */
void will_return_maybe(#function, LargestIntegralType value);
#else
#define will_return_maybe(function, value) \
    will_return_count(function, (value), WILL_RETURN_ONCE)
#endif
/** @} */

/**
 * @defgroup cmocka_param Checking Parameters
 * @ingroup cmocka
 *
 * Functionality to store expected values for mock function parameters.
 *
 * In addition to storing the return values of mock functions, cmocka provides
 * functionality to store expected values for mock function parameters using
 * the expect_*() functions provided. A mock function parameter can then be
 * validated using the check_expected() macro.
 *
 * Successive calls to expect_*() macros for a parameter queues values to check
 * the specified parameter. check_expected() checks a function parameter
 * against the next value queued using expect_*, if the parameter check fails
 * a test failure is signalled. In addition if check_expected() is called and
 * no more parameter values are queued a test failure occurs.
 *
 * The following test stub illustrates how to do this. First is the the function
 * we call in the test driver:
 *
 * @code
 * static void test_driver(void **state)
 * {
 *     expect_string(chef_cook, order, "hotdog");

```



```

* }
* @endcode
*
* Now the chef_cook function can check if the parameter we got passed is the
* parameter which is expected by the test driver. This can be done the
* following way:
*
* @code
* int chef_cook(const char *order, char **dish_out)
* {
*     check_expected(order);
* }
* @endcode
*
* For a complete example please take a look at
* <a href="http://git.cryptomilk.org/projects/cmocka.git/tree/example/chef_wrap/waiter_test_wrap.c">here</a>
*
* @ {
*/

/*
* Add a custom parameter checking function. If the event parameter is NULL
* the event structure is allocated internally by this function. If event
* parameter is provided it must be allocated on the heap and doesn't need to
* be deallocated by the caller.
*/
#ifdef DOXYGEN
/**
* @brief Add a custom parameter checking function.
*
* If the event parameter is NULL the event structure is allocated internally
* by this function. If the parameter is provided it must be allocated on the
* heap and doesn't need to be deallocated by the caller.
*
* @param[in] #function The function to add a custom parameter checking
*             function for.
*
* @param[in] #parameter The parameters passed to the function.
*
* @param[in] #check_function The check function to call.
*
* @param[in] check_data The data to pass to the check function.
*/
void expect_check(#function, #parameter, #check_function, const void *check_data);
#else
#define expect_check(function, parameter, check_function, check_data) \
    _expect_check(#function, #parameter, __FILE__, __LINE__, check_function, \
        cast_to_largest_integral_type(check_data), NULL, 1)
#endif

#ifdef DOXYGEN
/**

```

```

* @brief Add an event to check if the parameter value is part of the provided
*      array.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @see check_expected().
*/

```

```

void expect_in_set(#function, #parameter, LargestIntegralType value_array[]);
#else
#define expect_in_set(function, parameter, value_array) \
    expect_in_set_count(function, parameter, value_array, 1)
#endif

```

```

#ifdef DOXYGEN

```

```

/**
* @brief Add an event to check if the parameter value is part of the provided
*      array.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/

```

```

void expect_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);
#else
#define expect_in_set_count(function, parameter, value_array, count) \
    _expect_in_set(#function, #parameter, __FILE__, __LINE__, value_array, \
        sizeof(value_array) / sizeof((value_array)[0]), count)
#endif

```

```

#ifdef DOXYGEN

```

```

/**
* @brief Add an event to check if the parameter value is not part of the
*      provided array.
*
* The event is triggered by calling check_expected() in the mocked function.
*

```

```

* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @see check_expected().
*/
void expect_not_in_set(#function, #parameter, LargestIntegralType value_array[]);
#else
#define expect_not_in_set(function, parameter, value_array) \
    expect_not_in_set_count(function, parameter, value_array, 1)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to check if the parameter value is not part of the
*        provided array.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value_array[] The array to check for the value.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_in_set_count(#function, #parameter, LargestIntegralType value_array[], size_t count);
#else
#define expect_not_in_set_count(function, parameter, value_array, count) \
    _expect_not_in_set( \
        #function, #parameter, __FILE__, __LINE__, value_array, \
        sizeof(value_array) / sizeof((value_array)[0]), count)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to check a parameter is inside a numerical range.
* The check would succeed if minimum <= value <= maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.

```

```

*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @see check_expected().
*/
void expect_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType maximum);
#else
#define expect_in_range(function, parameter, minimum, maximum) \
    expect_in_range_count(function, parameter, minimum, maximum, 1)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to repeatedly check a parameter is inside a
* numerical range. The check would succeed if minimum <= value <= maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @param[in] count The count parameter returns the number of times the value
* should be returned by check_expected(). If count is set
* to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_in_range_count(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum, size_t count);
#else
#define expect_in_range_count(function, parameter, minimum, maximum, count) \
    _expect_in_range(#function, #parameter, __FILE__, __LINE__, minimum, \
        maximum, count)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to check a parameter is outside a numerical range.
* The check would succeed if minimum > value > maximum.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.

```

```

*
* @param[in] minimum The lower boundary of the interval to check against.
*
* @param[in] maximum The upper boundary of the interval to check against.
*
* @see check_expected().
*/
void expect_not_in_range(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum);
#else
#define expect_not_in_range(function, parameter, minimum, maximum) \
    expect_not_in_range_count(function, parameter, minimum, maximum, 1)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to repeatedly check a parameter is outside a
 * numerical range. The check would succeed if minimum > value > maximum.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] minimum The lower boundary of the interval to check against.
 *
 * @param[in] maximum The upper boundary of the interval to check against.
 *
 * @param[in] count The count parameter returns the number of times the value
 * should be returned by check_expected(). If count is set
 * to -1 the value will always be returned.
 *
 * @see check_expected().
*/
void expect_not_in_range_count(#function, #parameter, LargestIntegralType minimum, LargestIntegralType
maximum, size_t count);
#else
#define expect_not_in_range_count(function, parameter, minimum, maximum, \
    count) \
    _expect_not_in_range(#function, #parameter, __FILE__, __LINE__, \
        minimum, maximum, count)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to check if a parameter is the given value.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *

```

```

* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value The value to check.
*
* @see check_expected().
*/
void expect_value(#function, #parameter, LargestIntegralType value);
#else
#define expect_value(function, parameter, value) \
    expect_value_count(function, parameter, value, 1)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to repeatedly check if a parameter is the given value.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value The value to check.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_value_count(#function, #parameter, LargestIntegralType value, size_t count);
#else
#define expect_value_count(function, parameter, value, count) \
    _expect_value(#function, #parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to check if a parameter isn't the given value.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] value The value to check.
*
* @see check_expected().
*/
void expect_not_value(#function, #parameter, LargestIntegralType value);

```

```

#else
#define expect_not_value(function, parameter, value) \
    expect_not_value_count(function, parameter, value, 1)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to repeatedly check if a parameter isn't the given value.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] value The value to check.
 *
 * @param[in] count The count parameter returns the number of times the value
 *                  should be returned by check_expected(). If count is set
 *                  to -1 the value will always be returned.
 *
 * @see check_expected().
 */
void expect_not_value_count(#function, #parameter, LargestIntegralType value, size_t count);
#else
#define expect_not_value_count(function, parameter, value, count) \
    _expect_not_value(#function, #parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(value), count)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to check if the parameter value is equal to the
 *        provided string.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] string The string value to compare.
 *
 * @see check_expected().
 */
void expect_string(#function, #parameter, const char *string);
#else
#define expect_string(function, parameter, string) \
    expect_string_count(function, parameter, string, 1)
#endif

#ifdef DOXYGEN

```

```

/**
 * @brief Add an event to check if the parameter value is equal to the
 *        provided string.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] string The string value to compare.
 *
 * @param[in] count The count parameter returns the number of times the value
 *                  should be returned by check_expected(). If count is set
 *                  to -1 the value will always be returned.
 *
 * @see check_expected().
 */
void expect_string_count(#function, #parameter, const char *string, size_t count);
#else
#define expect_string_count(function, parameter, string, count) \
    _expect_string(#function, #parameter, __FILE__, __LINE__, \
        (const char*)(string), count)
#endif

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Add an event to check if the parameter value isn't equal to the
 *        provided string.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] string The string value to compare.
 *
 * @see check_expected().
 */

```

```

void expect_not_string(#function, #parameter, const char *string);
#else
#define expect_not_string(function, parameter, string) \
    expect_not_string_count(function, parameter, string, 1)
#endif

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Add an event to check if the parameter value isn't equal to the
 *        provided string.
 *
 * The event is triggered by calling check_expected() in the mocked function.

```



```

*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] string The string value to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_string_count(#function, #parameter, const char *string, size_t count);
#else
#define expect_not_string_count(function, parameter, string, count) \
    _expect_not_string(function, #parameter, __FILE__, __LINE__, \
        (const char*)(string), count)
#endif

```

```

#ifdef DOXYGEN
/**
* @brief Add an event to check if the parameter does match an area of memory.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] memory The memory to compare.
*
* @param[in] size The size of the memory to compare.
*
* @see check_expected().
*/
void expect_memory(#function, #parameter, void *memory, size_t size);
#else
#define expect_memory(function, parameter, memory, size) \
    expect_memory_count(function, parameter, memory, size, 1)
#endif

```

```

#ifdef DOXYGEN
/**
* @brief Add an event to repeatedly check if the parameter does match an area
*        of memory.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.

```

```

*
* @param[in] memory The memory to compare.
*
* @param[in] size The size of the memory to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_memory_count(#function, #parameter, void *memory, size_t size, size_t count);
#else
#define expect_memory_count(function, parameter, memory, size, count) \
    _expect_memory(#function, #parameter, __FILE__, __LINE__, \
        (const void*)(memory), size, count)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to check if the parameter doesn't match an area of
 *        memory.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *
 * @param[in] memory The memory to compare.
 *
 * @param[in] size The size of the memory to compare.
 *
 * @see check_expected().
*/
void expect_not_memory(#function, #parameter, void *memory, size_t size);
#else
#define expect_not_memory(function, parameter, memory, size) \
    expect_not_memory_count(function, parameter, memory, size, 1)
#endif

#ifdef DOXYGEN
/**
 * @brief Add an event to repeatedly check if the parameter doesn't match an
 *        area of memory.
 *
 * The event is triggered by calling check_expected() in the mocked function.
 *
 * @param[in] #function The function to add the check for.
 *
 * @param[in] #parameter The name of the parameter passed to the function.
 *

```

```

* @param[in] memory The memory to compare.
*
* @param[in] size The size of the memory to compare.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/
void expect_not_memory_count(#function, #parameter, void *memory, size_t size, size_t count);
#else
#define expect_not_memory_count(function, parameter, memory, size, count) \
    _expect_not_memory(#function, #parameter, __FILE__, __LINE__, \
        (const void*)(memory), size, count)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to check if a parameter (of any value) has been passed.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @see check_expected().
*/
void expect_any(#function, #parameter);
#else
#define expect_any(function, parameter) \
    expect_any_count(function, parameter, 1)
#endif

#ifdef DOXYGEN
/**
* @brief Add an event to repeatedly check if a parameter (of any value) has
*        been passed.
*
* The event is triggered by calling check_expected() in the mocked function.
*
* @param[in] #function The function to add the check for.
*
* @param[in] #parameter The name of the parameter passed to the function.
*
* @param[in] count The count parameter returns the number of times the value
*                  should be returned by check_expected(). If count is set
*                  to -1 the value will always be returned.
*
* @see check_expected().
*/

```

```

*/
void expect_any_count(#function, #parameter, size_t count);
#else
#define expect_any_count(function, parameter, count) \
    _expect_any(#function, #parameter, __FILE__, __LINE__, count)
#endif

#ifdef DOXYGEN
/**
 * @brief Determine whether a function parameter is correct.
 *
 * This ensures the next value queued by one of the expect_*( ) macros matches
 * the specified variable.
 *
 * This function needs to be called in the mock object.
 *
 * @param[in] #parameter The parameter to check.
 */
void check_expected(#parameter);
#else
#define check_expected(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
        cast_to_largest_integral_type(parameter))
#endif

#ifdef DOXYGEN
/**
 * @brief Determine whether a function parameter is correct.
 *
 * This ensures the next value queued by one of the expect_*( ) macros matches
 * the specified variable.
 *
 * This function needs to be called in the mock object.
 *
 * @param[in] #parameter The pointer to check.
 */
void check_expected_ptr(#parameter);
#else
#define check_expected_ptr(parameter) \
    _check_expected(__func__, #parameter, __FILE__, __LINE__, \
        cast_ptr_to_largest_integral_type(parameter))
#endif

/** @} */

/**
 * @defgroup cmocka_asserts Assert Macros
 * @ingroup cmocka
 *
 * This is a set of useful assert macros like the standard C library's
 * assert(3) macro.
 */

```

```

* On an assertion failure a cmocka assert macro will write the failure to the
* standard error stream and signal a test failure. Due to limitations of the C
* language the general C standard library assert() and cmocka's assert_true()
* and assert_false() macros can only display the expression that caused the
* assert failure. cmocka's type specific assert macros, assert_{type}_equal()
* and assert_{type}_not_equal(), display the data that caused the assertion
* failure which increases data visibility aiding debugging of failing test
* cases.
*
* @{\
*/

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Assert that the given expression is true.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if expression is false (i.e., compares equal to
 * zero).
 *
 * @param[in] expression The expression to evaluate.
 *
 * @see assert_int_equal()
 * @see assert_string_equal()
 */

```

```

void assert_true(scalar expression);

```

```

#else

```

```

#define assert_true(c) _assert_true(cast_to_largest_integral_type(c), #c, \
    __FILE__, __LINE__)

```

```

#endif

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Assert that the given expression is false.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if expression is true.
 *
 * @param[in] expression The expression to evaluate.
 *
 * @see assert_int_equal()
 * @see assert_string_equal()
 */

```

```

void assert_false(scalar expression);

```

```

#else

```

```

#define assert_false(c) _assert_true(!(cast_to_largest_integral_type(c)), #c, \
    __FILE__, __LINE__)

```

```

#endif

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Assert that the return_code is greater than or equal to 0.

```

```

*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the return code is smaller than 0. If the function
* you check sets an errno if it fails you can pass it to the function and
* it will be printed as part of the error message.
*
* @param[in] rc    The return code to evaluate.
*
* @param[in] error  Pass errno here or 0.
*/
void assert_return_code(int rc, int error);
#else
#define assert_return_code(rc, error) \
    _assert_return_code(cast_to_largest_integral_type(rc), \
        sizeof(rc), \
        cast_to_largest_integral_type(error), \
        #rc, __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the given pointer is non-NULL.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the pointer is non-NULL.
 *
 * @param[in] pointer The pointer to evaluate.
 *
 * @see assert_null()
 */
void assert_non_null(void *pointer);
#else
#define assert_non_null(c) _assert_true(cast_ptr_to_largest_integral_type(c), #c, \
    __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the given pointer is NULL.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the pointer is non-NULL.
 *
 * @param[in] pointer The pointer to evaluate.
 *
 * @see assert_non_null()
 */
void assert_null(void *pointer);
#else
#define assert_null(c) _assert_true(!(cast_ptr_to_largest_integral_type(c)), #c, \
    __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
 * @brief Assert that the two given pointers are equal.
 *
 * The function prints an error message and terminates the test by calling
 * fail() if the pointers are not equal.
 *
 * @param[in] a    The first pointer to compare.
 *
 * @param[in] b    The pointer to compare against the first one.
 */
void assert_ptr_equal(void *a, void *b);
#else
#define assert_ptr_equal(a, b) \
    _assert_int_equal(cast_ptr_to_largest_integral_type(a), \
        cast_ptr_to_largest_integral_type(b), \
        __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
 * @brief Assert that the two given pointers are not equal.
 *
 * The function prints an error message and terminates the test by calling
 * fail() if the pointers are equal.
 *
 * @param[in] a    The first pointer to compare.
 *
 * @param[in] b    The pointer to compare against the first one.
 */
void assert_ptr_not_equal(void *a, void *b);
#else
#define assert_ptr_not_equal(a, b) \
    _assert_int_not_equal(cast_ptr_to_largest_integral_type(a), \
        cast_ptr_to_largest_integral_type(b), \
        __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
 * @brief Assert that the two given integers are equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the integers are not equal.
 *
 * @param[in] a    The first integer to compare.
 *
 * @param[in] b    The integer to compare against the first one.
 */
void assert_int_equal(int a, int b);
#else

```

```

#define assert_int_equal(a, b) \
    _assert_int_equal(cast_to_largest_integral_type(a), \
        cast_to_largest_integral_type(b), \
        __FILE__, __LINE__)

#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the two given integers are not equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the integers are equal.
 *
 * @param[in] a The first integer to compare.
 *
 * @param[in] b The integer to compare against the first one.
 *
 * @see assert_int_equal()
 */
void assert_int_not_equal(int a, int b);
#else
#define assert_int_not_equal(a, b) \
    _assert_int_not_equal(cast_to_largest_integral_type(a), \
        cast_to_largest_integral_type(b), \
        __FILE__, __LINE__)

#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the two given strings are equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the strings are not equal.
 *
 * @param[in] a The string to check.
 *
 * @param[in] b The other string to compare.
 */
void assert_string_equal(const char *a, const char *b);
#else
#define assert_string_equal(a, b) \
    _assert_string_equal((const char*)(a), (const char*)(b), __FILE__, \
        __LINE__)

#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the two given strings are not equal.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if the strings are equal.
 *

```



```

* @param[in] a The string to check.
*
* @param[in] b The other string to compare.
*/
void assert_string_not_equal(const char *a, const char *b);
#else
#define assert_string_not_equal(a, b) \
    _assert_string_not_equal((const char*)(a), (const char*)(b), __FILE__, \
        __LINE__)
#endif

#ifdef DOXYGEN
/**
* @brief Assert that the two given areas of memory are equal, otherwise fail.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the memory is not equal.
*
* @param[in] a The first memory area to compare
*              (interpreted as unsigned char).
*
* @param[in] b The second memory area to compare
*              (interpreted as unsigned char).
*
* @param[in] size The first n bytes of the memory areas to compare.
*/
void assert_memory_equal(const void *a, const void *b, size_t size);
#else
#define assert_memory_equal(a, b, size) \
    _assert_memory_equal((const void*)(a), (const void*)(b), size, __FILE__, \
        __LINE__)
#endif

#ifdef DOXYGEN
/**
* @brief Assert that the two given areas of memory are not equal.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if the memory is equal.
*
* @param[in] a The first memory area to compare
*              (interpreted as unsigned char).
*
* @param[in] b The second memory area to compare
*              (interpreted as unsigned char).
*
* @param[in] size The first n bytes of the memory areas to compare.
*/
void assert_memory_not_equal(const void *a, const void *b, size_t size);
#else
#define assert_memory_not_equal(a, b, size) \
    _assert_memory_not_equal((const void*)(a), (const void*)(b), size, \

```

```

        __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the specified value is not smaller than the minimum
 * and and not greater than the maximum.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if value is not in range.
 *
 * @param[in] value The value to check.
 *
 * @param[in] minimum The minimum value allowed.
 *
 * @param[in] maximum The maximum value allowed.
 */
void assert_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType maximum);
#else
#define assert_in_range(value, minimum, maximum) \
    _assert_in_range( \
        cast_to_largest_integral_type(value), \
        cast_to_largest_integral_type(minimum), \
        cast_to_largest_integral_type(maximum), __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
 * @brief Assert that the specified value is smaller than the minimum or
 * greater than the maximum.
 *
 * The function prints an error message to standard error and terminates the
 * test by calling fail() if value is in range.
 *
 * @param[in] value The value to check.
 *
 * @param[in] minimum The minimum value to compare.
 *
 * @param[in] maximum The maximum value to compare.
 */
void assert_not_in_range(LargestIntegralType value, LargestIntegralType minimum, LargestIntegralType
maximum);
#else
#define assert_not_in_range(value, minimum, maximum) \
    _assert_not_in_range( \
        cast_to_largest_integral_type(value), \
        cast_to_largest_integral_type(minimum), \
        cast_to_largest_integral_type(maximum), __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**

```

```

* @brief Assert that the specified value is within a set.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is not within a set.
*
* @param[in] value The value to look up
*
* @param[in] values[] The array to check for the value.
*
* @param[in] count The size of the values array.
*/
void assert_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);
#else
#define assert_in_set(value, values, number_of_values) \
    _assert_in_set(value, values, number_of_values, __FILE__, __LINE__)
#endif

#ifndef DOXYGEN
/**
* @brief Assert that the specified value is not within a set.
*
* The function prints an error message to standard error and terminates the
* test by calling fail() if value is within a set.
*
* @param[in] value The value to look up
*
* @param[in] values[] The array to check for the value.
*
* @param[in] count The size of the values array.
*/
void assert_not_in_set(LargestIntegralType value, LargestIntegralType values[], size_t count);
#else
#define assert_not_in_set(value, values, number_of_values) \
    _assert_not_in_set(value, values, number_of_values, __FILE__, __LINE__)
#endif

/** @} */

/**
* @defgroup cmocka_call_order Call Ordering
* @ingroup cmocka
*
* It is often beneficial to make sure that functions are called in an
* order. This is independent of mock returns and parameter checking as both
* of the aforementioned do not check the order in which they are called from
* different functions.
*
* <ul>
* <li><strong>expect_function_call(function)</strong> - The
* expect_function_call() macro pushes an expectation onto the stack of
* expected calls.</li>
*

```

- \* `function_called()` - pops a value from the stack of expected calls. `function_called()` is invoked within the mock object that uses it.

\* `expect_function_call()` and `function_called()` are intended to be used in pairs. Cmocka will fail a test if there are more or less expected calls created (e.g. `expect_function_call()`) than consumed with `function_called()`. There are provisions such as `ignore_function_calls()` which allow this restriction to be circumvented in tests where mock calls for the code under test are not the focus of the test.

The following example illustrates how a unit test instructs cmocka to expect a `function_called()` from a particular mock, `chef_sing()`:

```
@code
void chef_sing(void);

void code_under_test()
{
    chef_sing();
}

void some_test(void **state)
{
    expect_function_call(chef_sing);
    code_under_test();
}
@endcode
```

The implementation of the mock then must check whether it was meant to be called by invoking `function_called()`:

```
@code
void chef_sing()
{
    function_called();
}
@endcode

@{
/
```

```
#ifndef DOXYGEN
```

```
/**
 * @brief Check that current mocked function is being called in the expected
 *        order
 *
 * @see expect_function_call()
 */
void function_called(void);
```

```
#else
#define function_called() _function_called(__func__, __FILE__, __LINE__)
#endif
```

```
#ifndef DOXYGEN
```

```
/**
 * @brief Store expected call(s) to a mock to be checked by function_called()
 *       later.
 *
 * @param[in] #function The function which should should be called
 *
 * @param[in] times number of times this mock must be called
 *
 * @see function_called()
 */
```

```
void expect_function_calls(#function, const int times);
```

```
#else
#define expect_function_calls(function, times) \
    _expect_function_call(#function, __FILE__, __LINE__, times)
#endif
```

```
#ifndef DOXYGEN
```

```
/**
 * @brief Store expected single call to a mock to be checked by
 *       function_called() later.
 *
 * @param[in] #function The function which should should be called
 *
 * @see function_called()
 */
```

```
void expect_function_call(#function);
```

```
#else
#define expect_function_call(function) \
    _expect_function_call(#function, __FILE__, __LINE__, 1)
#endif
```

```
#ifndef DOXYGEN
```

```
/**
 * @brief Expects function_called() from given mock at least once
 *
 * @param[in] #function The function which should should be called
 *
 * @see function_called()
 */
```

```
void expect_function_call_any(#function);
```

```
#else
#define expect_function_call_any(function) \
    _expect_function_call(#function, __FILE__, __LINE__, -1)
#endif
```

```
#ifndef DOXYGEN
```

```
/**
```

```

* @brief Ignores function_called() invocations from given mock function.
*
* @param[in] #function The function which should be called
*
* @see function_called()
*/
void ignore_function_calls(#function);
#else
#define ignore_function_calls(function) \
    _expect_function_call(#function, __FILE__, __LINE__, -2)
#endif

/** @} */

/**
* @defgroup cmocka_exec Running Tests
* @ingroup cmocka
*
* This is the way tests are executed with CMocka.
*
* The following example illustrates this macro's use with the unit_test macro.
*
* @code
* void Test0(void **state);
* void Test1(void **state);
*
* int main(void)
* {
*     const struct CMUnitTest tests[] = {
*         cmocka_unit_test(Test0),
*         cmocka_unit_test(Test1),
*     };
*
*     return cmocka_run_group_tests(tests, NULL, NULL);
* }
* @endcode
*
* @{
*/

#ifdef DOXYGEN
/**
* @brief Forces the test to fail immediately and quit.
*/
void fail(void);
#else
#define fail() _fail(__FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
* @brief Forces the test to not be executed, but marked as skipped

```

```

*/
void skip(void);
#else
#define skip() _skip(__FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
 * @brief Forces the test to fail immediately and quit, printing the reason.
 *
 * @code
 * fail_msg("This is some error message for test");
 * @endcode
 *
 * or
 *
 * @code
 * char *error_msg = "This is some error message for test";
 * fail_msg("%s", error_msg);
 * @endcode
 */
void fail_msg(const char *msg, ...);
#else
#define fail_msg(msg, ...) do { \
    print_error("ERROR: " msg "\n", ##__VA_ARGS__); \
    fail(); \
} while (0)
#endif

#ifdef DOXYGEN
/**
 * @brief Generic method to run a single test.
 *
 * @deprecated This function was deprecated in favor of cmocka_run_group_tests
 *
 * @param[in] #function The function to test.
 *
 * @return 0 on success, 1 if an error occurred.
 *
 * @code
 * // A test case that does nothing and succeeds.
 * void null_test_success(void **state) {
 * }
 *
 * int main(void) {
 *     return run_test(null_test_success);
 * }
 * @endcode
 */
int run_test(#function);
#else
#define run_test(f) _run_test(#f, f, NULL, UNIT_TEST_FUNCTION_TYPE_TEST, NULL)

```

```
#endif
```

```
static inline void _unit_test_dummy(void **state) {  
    (void)state;  
}
```

```
/** Initializes a UnitTest structure.
```

```
*
```

```
* @deprecated This function was deprecated in favor of cmocka_unit_test
```

```
*/
```

```
#define unit_test(f) { #f, f, UNIT_TEST_FUNCTION_TYPE_TEST }
```

```
#define _unit_test_setup(test, setup) \  
    { #test "_" #setup, setup, UNIT_TEST_FUNCTION_TYPE_SETUP }
```

```
/** Initializes a UnitTest structure with a setup function.
```

```
*
```

```
* @deprecated This function was deprecated in favor of cmocka_unit_test_setup
```

```
*/
```

```
#define unit_test_setup(test, setup) \  
    _unit_test_setup(test, setup), \  
    unit_test(test), \  
    _unit_test_teardown(test, _unit_test_dummy)
```

```
#define _unit_test_teardown(test, teardown) \  
    { #test "_" #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_TEARDOWN }
```

```
/** Initializes a UnitTest structure with a teardown function.
```

```
*
```

```
* @deprecated This function was deprecated in favor of cmocka_unit_test_teardown
```

```
*/
```

```
#define unit_test_teardown(test, teardown) \  
    _unit_test_setup(test, _unit_test_dummy), \  
    unit_test(test), \  
    _unit_test_teardown(test, teardown)
```

```
/** Initializes a UnitTest structure for a group setup function.
```

```
*
```

```
* @deprecated This function was deprecated in favor of cmocka_run_group_tests
```

```
*/
```

```
#define group_test_setup(setup) \  
    { "group_" #setup, setup, UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP }
```

```
/** Initializes a UnitTest structure for a group teardown function.
```

```
*
```

```
* @deprecated This function was deprecated in favor of cmocka_run_group_tests
```

```
*/
```

```
#define group_test_teardown(teardown) \  
    { "group_" #teardown, teardown, UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN }
```

```
/**
```

```
* Initialize an array of UnitTest structures with a setup function for a test
```



```

* and a teardown function. Either setup or teardown can be NULL.
*
* @deprecated This function was deprecated in favor of
* cmocka_unit_test_setup_teardown
*/
#define unit_test_setup_teardown(test, setup, teardown) \
    _unit_test_setup(test, setup), \
    unit_test(test), \
    _unit_test_teardown(test, teardown)

/** Initializes a CMUnitTest structure. */
#define cmocka_unit_test(f) { #f, f, NULL, NULL, NULL }

/** Initializes a CMUnitTest structure with a setup function. */
#define cmocka_unit_test_setup(f, setup) { #f, f, setup, NULL, NULL }

/** Initializes a CMUnitTest structure with a teardown function. */
#define cmocka_unit_test_teardown(f, teardown) { #f, f, NULL, teardown, NULL }

/**
 * Initialize an array of CMUnitTest structures with a setup function for a test
 * and a teardown function. Either setup or teardown can be NULL.
 */
#define cmocka_unit_test_setup_teardown(f, setup, teardown) { #f, f, setup, teardown, NULL }

/**
 * Initialize a CMUnitTest structure with given initial state. It will be passed
 * to test function as an argument later. It can be used when test state does
 * not need special initialization or was initialized already.
 * @note If the group setup function initialized the state already, it won't be
 * overridden by the initial state defined here.
 */
#define cmocka_unit_test_prestate(f, state) { #f, f, NULL, NULL, state }

/**
 * Initialize a CMUnitTest structure with given initial state, setup and
 * teardown function. Any of these values can be NULL. Initial state is passed
 * later to setup function, or directly to test if none was given.
 * @note If the group setup function initialized the state already, it won't be
 * overridden by the initial state defined here.
 */
#define cmocka_unit_test_prestate_setup_teardown(f, setup, teardown, state) { #f, f, setup, teardown, state }

#define run_tests(tests) _run_tests(tests, sizeof(tests) / sizeof(tests)[0])
#define run_group_tests(tests) _run_group_tests(tests, sizeof(tests) / sizeof(tests)[0])

#ifndef DOXYGEN
/**
 * @brief Run tests specified by an array of CMUnitTest structures.
 *
 * @param[in] group_tests[] The array of unit tests to execute.

```

```

*
* @param[in] group_setup The setup function which should be called before
*                       all unit tests are executed.
*
* @param[in] group_teardown The teardown function to be called after all
*                       tests have finished.
*
* @return 0 on success, or the number of failed tests.
*
* @code
* static int setup(void **state) {
*     int *answer = malloc(sizeof(int));
*     if (*answer == NULL) {
*         return -1;
*     }
*     *answer = 42;
*
*     *state = answer;
*
*     return 0;
* }
*
* static int teardown(void **state) {
*     free(*state);
*
*     return 0;
* }
*
* static void null_test_success(void **state) {
*     (void) state;
* }
*
* static void int_test_success(void **state) {
*     int *answer = *state;
*     assert_int_equal(*answer, 42);
* }
*
* int main(void) {
*     const struct CMUnitTest tests[] = {
*         cmocka_unit_test(null_test_success),
*         cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),
*     };
*
*     return cmocka_run_group_tests(tests, NULL, NULL);
* }
* @endcode
*
* @see cmocka_unit_test
* @see cmocka_unit_test_setup
* @see cmocka_unit_test_teardown
* @see cmocka_unit_test_setup_teardown
*/

```

```

int cmocka_run_group_tests(const struct CMUnitTest group_tests[],
                           CMFixtureFunction group_setup,
                           CMFixtureFunction group_teardown);

#else
#define cmocka_run_group_tests(group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(#group_tests, group_tests, sizeof(group_tests) / sizeof(group_tests)[0], \
    group_setup, group_teardown)
#endif

#ifdef DOXYGEN
/**
 * @brief Run tests specified by an array of CMUnitTest structures and specify
 *        a name.
 *
 * @param[in] group_name    The name of the group test.
 *
 * @param[in] group_tests[] The array of unit tests to execute.
 *
 * @param[in] group_setup   The setup function which should be called before
 *                          all unit tests are executed.
 *
 * @param[in] group_teardown The teardown function to be called after all
 *                          tests have finished.
 *
 * @return 0 on success, or the number of failed tests.
 *
 * @code
 * static int setup(void **state) {
 *     int *answer = malloc(sizeof(int));
 *     if (*answer == NULL) {
 *         return -1;
 *     }
 *     *answer = 42;
 *
 *     *state = answer;
 *
 *     return 0;
 * }
 *
 * static int teardown(void **state) {
 *     free(*state);
 *
 *     return 0;
 * }
 *
 * static void null_test_success(void **state) {
 *     (void) state;
 * }
 *
 * static void int_test_success(void **state) {
 *     int *answer = *state;
 *     assert_int_equal(*answer, 42);

```

```

* }
*
* int main(void) {
*     const struct CMUnitTest tests[] = {
*         cmocka_unit_test(null_test_success),
*         cmocka_unit_test_setup_teardown(int_test_success, setup, teardown),
*     };
*
*     return cmocka_run_group_tests_name("success_test", tests, NULL, NULL);
* }
* @endcode
*
* @see cmocka_unit_test
* @see cmocka_unit_test_setup
* @see cmocka_unit_test_teardown
* @see cmocka_unit_test_setup_teardown
*/
int cmocka_run_group_tests_name(const char *group_name,
                                const struct CMUnitTest group_tests[],
                                CMFixtureFunction group_setup,
                                CMFixtureFunction group_teardown);
#else
# define cmocka_run_group_tests_name(group_name, group_tests, group_setup, group_teardown) \
    _cmocka_run_group_tests(group_name, group_tests, sizeof(group_tests) / sizeof(group_tests)[0],
group_setup, group_teardown)
#endif

/** @} */

/**
* @defgroup cmocka_alloc Dynamic Memory Allocation
* @ingroup cmocka
*
* Memory leaks, buffer overflows and underflows can be checked using cmocka.
*
* To test for memory leaks, buffer overflows and underflows a module being
* tested by cmocka should replace calls to malloc(), calloc() and free() to
* test_malloc(), test_calloc() and test_free() respectively. Each time a block
* is deallocated using test_free() it is checked for corruption, if a corrupt
* block is found a test failure is signalled. All blocks allocated using the
* test_*() allocation functions are tracked by the cmocka library. When a test
* completes if any allocated blocks (memory leaks) remain they are reported
* and a test failure is signalled.
*
* For simplicity cmocka currently executes all tests in one process. Therefore
* all test cases in a test application share a single address space which
* means memory corruption from a single test case could potentially cause the
* test application to exit prematurely.
*
* @{
*/

```

```

#ifdef DOXYGEN
/**
 * @brief Test function overriding malloc.
 *
 * @param[in] size The bytes which should be allocated.
 *
 * @return A pointer to the allocated memory or NULL on error.
 *
 * @code
 * #ifdef UNIT_TESTING
 * extern void* _test_malloc(const size_t size, const char* file, const int line);
 *
 * #define malloc(size) _test_malloc(size, __FILE__, __LINE__)
 * #endif
 *
 * void leak_memory() {
 *     int * const temporary = (int*)malloc(sizeof(int));
 *     *temporary = 0;
 * }
 * @endcode
 *
 * @see malloc(3)
 */
void *test_malloc(size_t size);
#else
#define test_malloc(size) _test_malloc(size, __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
 * @brief Test function overriding calloc.
 *
 * The memory is set to zero.
 *
 * @param[in] nmemb The number of elements for an array to be allocated.
 *
 * @param[in] size The size in bytes of each array element to allocate.
 *
 * @return A pointer to the allocated memory, NULL on error.
 *
 * @see calloc(3)
 */
void *test_calloc(size_t nmemb, size_t size);
#else
#define test_calloc(num, size) _test_calloc(num, size, __FILE__, __LINE__)
#endif

```

```

#ifdef DOXYGEN
/**
 * @brief Test function overriding realloc which detects buffer overruns
 * and memory leaks.
 *

```

```

* @param[in] ptr The memory block which should be changed.
*
* @param[in] size The bytes which should be allocated.
*
* @return The newly allocated memory block, NULL on error.
*/
void *test_realloc(void *ptr, size_t size);
#else
#define test_realloc(ptr, size) _test_realloc(ptr, size, __FILE__, __LINE__)
#endif

#ifdef DOXYGEN
/**
* @brief Test function overriding free(3).
*
* @param[in] ptr The pointer to the memory space to free.
*
* @see free(3).
*/
void test_free(void *ptr);
#else
#define test_free(ptr) _test_free(ptr, __FILE__, __LINE__)
#endif

/* Redirect malloc, calloc and free to the unit test allocators. */
#ifdef UNIT_TESTING
#define malloc test_malloc
#define realloc test_realloc
#define calloc test_calloc
#define free test_free
#endif /* UNIT_TESTING */

/** @} */

/**
* @defgroup cmocka_mock_assert Standard Assertions
* @ingroup cmocka
*
* How to handle assert(3) of the standard C library.
*
* Runtime assert macros like the standard C library's assert() should be
* redefined in modules being tested to use cmocka's mock_assert() function.
* Normally mock_assert() signals a test failure. If a function is called using
* the expect_assert_failure() macro, any calls to mock_assert() within the
* function will result in the execution of the test. If no calls to
* mock_assert() occur during the function called via expect_assert_failure() a
* test failure is signalled.
*
* @{
*/

```

```

/**
 * @brief Function to replace assert(3) in tested code.
 *
 * In conjunction with check_assert() it's possible to determine whether an
 * assert condition has failed without stopping a test.
 *
 * @param[in] result The expression to assert.
 *
 * @param[in] expression The expression as string.
 *
 * @param[in] file The file mock_assert() is called.
 *
 * @param[in] line The line mock_assert() is called.
 *
 * @code
 * #ifdef UNIT_TESTING
 * extern void mock_assert(const int result, const char* const expression,
 *                        const char * const file, const int line);
 *
 * #undef assert
 * #define assert(expression) \
 *     mock_assert((int)(expression), #expression, __FILE__, __LINE__);
 * #endif
 *
 * void increment_value(int * const value) {
 *     assert(value);
 *     (*value)++;
 * }
 * @endcode
 *
 * @see assert(3)
 * @see expect_assert_failure
 */

```

```

void mock_assert(const int result, const char* const expression,
                const char * const file, const int line);

```

```

#ifdef DOXYGEN

```

```

/**
 * @brief Ensure that mock_assert() is called.
 *
 * If mock_assert() is called the assert expression string is returned.
 *
 * @param[in] fn_call The function will call mock_assert().
 *
 * @code
 * #define assert mock_assert
 *
 * void showmessage(const char *message) {
 *     assert(message);
 * }
 *
 * int main(int argc, const char* argv[]) {

```

```

* expect_assert_failure(show_message(NULL));
* printf("succeeded\n");
* return 0;
* }
* @endcode
*
*/

void expect_assert_failure(function fn_call);
#else
#define expect_assert_failure(function_call) \
{ \
    const int result = setjmp(global_expect_assert_env); \
    global_expecting_assert = 1; \
    if (result) { \
        print_message("Expected assertion %s occurred\n", \
            global_last_failed_assert); \
        global_expecting_assert = 0; \
    } else { \
        function_call ; \
        global_expecting_assert = 0; \
        print_error("Expected assert in %s\n", #function_call); \
        _fail(__FILE__, __LINE__); \
    } \
}
#endif

/** @} */

/* Function prototype for setup, test and teardown functions. */
typedef void (*UnitTestFixture)(void **state);

/* Function that determines whether a function parameter value is correct. */
typedef int (*CheckParameterValue)(const LargestIntegralType value,
    const LargestIntegralType check_value_data);

/* Type of the unit test function. */
typedef enum UnitTestFixtureType {
    UNIT_TEST_FUNCTION_TYPE_TEST = 0,
    UNIT_TEST_FUNCTION_TYPE_SETUP,
    UNIT_TEST_FUNCTION_TYPE_TEARDOWN,
    UNIT_TEST_FUNCTION_TYPE_GROUP_SETUP,
    UNIT_TEST_FUNCTION_TYPE_GROUP_TEARDOWN,
} UnitTestFixtureType;

/*
* Stores a unit test function with its name and type.
* NOTE: Every setup function must be paired with a teardown function. It's
* possible to specify NULL function pointers.
*/
typedef struct UnitTest {
    const char* name;
    UnitTestFixture function;

```



```

    UnitTestFunctionType function_type;
} UnitTest;

typedef struct GroupTest {
    UnitTestFunction setup;
    UnitTestFunction teardown;
    const UnitTest *tests;
    const size_t number_of_tests;
} GroupTest;

/* Function prototype for test functions. */
typedef void (*CMUnitTestFunction)(void **state);

/* Function prototype for setup and teardown functions. */
typedef int (*CMFixtureFunction)(void **state);

struct CMUnitTest {
    const char *name;
    CMUnitTestFunction test_func;
    CMFixtureFunction setup_func;
    CMFixtureFunction teardown_func;
    void *initial_state;
};

/* Location within some source code. */
typedef struct SourceLocation {
    const char* file;
    int line;
} SourceLocation;

/* Event that's called to check a parameter value. */
typedef struct CheckParameterEvent {
    SourceLocation location;
    const char *parameter_name;
    CheckParameterValue check_value;
    LargestIntegralType check_value_data;
} CheckParameterEvent;

/* Used by expect_assert_failure() and mock_assert(). */
extern int global_expecting_assert;
extern jmp_buf global_expect_assert_env;
extern const char * global_last_failed_assert;

/* Retrieves a value for the given function, as set by "will_return". */
LargestIntegralType _mock(const char * const function, const char* const file,
                          const int line);

void _expect_function_call(
    const char * const function_name,
    const char * const file,
    const int line,
    const int count);

```

```

void _function_called(const char * const function, const char* const file,
                     const int line);

void _expect_check(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const CheckParameterValue check_function,
    const LargestIntegralType check_data, CheckParameterEvent * const event,
    const int count);

void _expect_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType values[],
    const size_t number_of_values, const int count);

void _expect_not_in_set(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType values[],
    const size_t number_of_values, const int count);

void _expect_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType minimum,
    const LargestIntegralType maximum, const int count);

void _expect_not_in_range(
    const char* const function, const char* const parameter,
    const char* const file, const int line,
    const LargestIntegralType minimum,
    const LargestIntegralType maximum, const int count);

void _expect_value(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType value,
    const int count);

void _expect_not_value(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const LargestIntegralType value,
    const int count);

void _expect_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count);

void _expect_not_string(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const char* string,
    const int count);

void _expect_memory(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const void* const memory,

```

```

    const size_t size, const int count);
void _expect_not_memory(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const void* const memory,
    const size_t size, const int count);

void _expect_any(
    const char* const function, const char* const parameter,
    const char* const file, const int line, const int count);

void _check_expected(
    const char * const function_name, const char * const parameter_name,
    const char* file, const int line, const LargestIntegralType value);

void _will_return(const char * const function_name, const char * const file,
    const int line, const LargestIntegralType value,
    const int count);
void _assert_true(const LargestIntegralType result,
    const char* const expression,
    const char * const file, const int line);
void _assert_return_code(const LargestIntegralType result,
    size_t rlen,
    const LargestIntegralType error,
    const char * const expression,
    const char * const file,
    const int line);
void _assert_int_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);
void _assert_int_not_equal(
    const LargestIntegralType a, const LargestIntegralType b,
    const char * const file, const int line);
void _assert_string_equal(const char * const a, const char * const b,
    const char * const file, const int line);
void _assert_string_not_equal(const char * const a, const char * const b,
    const char *file, const int line);
void _assert_memory_equal(const void * const a, const void * const b,
    const size_t size, const char* const file,
    const int line);
void _assert_memory_not_equal(const void * const a, const void * const b,
    const size_t size, const char* const file,
    const int line);
void _assert_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);
void _assert_not_in_range(
    const LargestIntegralType value, const LargestIntegralType minimum,
    const LargestIntegralType maximum, const char* const file, const int line);
void _assert_in_set(
    const LargestIntegralType value, const LargestIntegralType values[],
    const size_t number_of_values, const char* const file, const int line);
void _assert_not_in_set(

```

```

const LargestIntegralType value, const LargestIntegralType values[],
const size_t number_of_values, const char* const file, const int line);

void* _test_malloc(const size_t size, const char* file, const int line);
void* _test_realloc(void *ptr, const size_t size, const char* file, const int line);
void* _test_calloc(const size_t number_of_elements, const size_t size,
    const char* file, const int line);
void _test_free(void* const ptr, const char* file, const int line);

void _fail(const char * const file, const int line);

void _skip(const char * const file, const int line);

int _run_test(
    const char * const function_name, const UnitTestFunction Function,
    void ** const volatile state, const UnitTestFunctionType function_type,
    const void* const heap_check_point);
CMOCKA_DEPRECATED int _run_tests(const UnitTest * const tests,
    const size_t number_of_tests);
CMOCKA_DEPRECATED int _run_group_tests(const UnitTest * const tests,
    const size_t number_of_tests);

/* Test runner */
int _cmocka_run_group_tests(const char *group_name,
    const struct CMockaUnit * const tests,
    const size_t num_tests,
    CMFixtureFunction group_setup,
    CMFixtureFunction group_teardown);

/* Standard output and error print methods. */
void print_message(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void print_error(const char* const format, ...) CMOCKA_PRINTF_ATTRIBUTE(1, 2);
void vprint_message(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);
void vprint_error(const char* const format, va_list args) CMOCKA_PRINTF_ATTRIBUTE(1, 0);

enum cm_message_output {
    CM_OUTPUT_STDOUT,
    CM_OUTPUT_SUBUNIT,
    CM_OUTPUT_TAP,
    CM_OUTPUT_XML,
};

/**
 * @brief Function to set the output format for a test.
 *
 * The output format for the test can either be set globally using this
 * function or overridden with environment variable CMOCKA_MESSAGE_OUTPUT.
 *
 * The environment variable can be set to either STDOUT, SUBUNIT, TAP or XML.
 *
 * @param[in] output The output format to use for the test.
 */

```

```

*/
void cmocka_set_message_output(enum cm_message_output output);

/** @} */

#endif /* CMOCKA_H_ */
/*
 * Copyright 2014 Luis Pabon, Jr.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
 * Programming by Contract is a programming methodology
 * which binds the caller and the function called to a
 * contract. The contract is represented using Hoare Triple:
 *   {P} C {Q}
 * where {P} is the precondition before executing command C,
 * and {Q} is the postcondition.
 *
 * See also:
 * http://en.wikipedia.org/wiki/Design_by_contract
 * http://en.wikipedia.org/wiki/Hoare_logic
 * http://dlang.org/dbc.html
 */
#ifndef CMOCKA_PBC_H_
#define CMOCKA_PBC_H_

#if defined(UNIT_TESTING) || defined(DEBUG)

#include <assert.h>

/*
 * Checks caller responsibility against contract
 */
#define REQUIRE(cond) assert(cond)

/*
 * Checks function responsibility against contract.
 */
#define ENSURE(cond) assert(cond)

```

```

/*
 * While REQUIRE and ENSURE apply to functions, INVARIANT
 * applies to classes/structs. It ensures that instances
 * of the class/struct are consistent. In other words,
 * that the instance has not been corrupted.
 */
#define INVARIANT(invariant_fnc) do{ (invariant_fnc) } while (0);

#else
#define REQUIRE(cond) do { } while (0);
#define ENSURE(cond) do { } while (0);
#define INVARIANT(invariant_fnc) do{ } while (0);

#endif /* defined(UNIT_TESTING) || defined (DEBUG) */
#endif /* CMOCKA_PBC_H_ */

/**
 * @file arch_arm32.h
 * @brief defines important arm macros and provides hardware interface functions
 *
 * this file contains an endianness lookup function for ARM, as well
 * as relevant memory locations in ARM to allow this
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 */
#ifndef __ARCH_ARM32_H__
#define __ARCH_ARM32_H__

/* Type definitions needed for function prototypes */
#include <stdint.h>

#define __SCB_ADDRESS (0xE000E000)

#define __AIRCR_ADDRESS_OFFSET (0xD0C)
#define __AIRCR (__SCB_ADDRESS & __AIRCR_ADDRESS_OFFSET)
#define __AIRCR_ENDIANNESSESS_OFFSET (15)
#define __AIRCR_ENDIANNESSESS_MASK (0x8000)

#define __CPUID_ADDRESS_OFFSET (0xD00)
#define __CPUID (__SCB_ADDRESS & __CPUID_ADDRESS_OFFSET)
#define __CPUID_PART_NO_OFFSET (4)
#define __CPUID_PART_NO_MASK (0xFFF0)

#define __CCR_ADDRESS_OFFSET (0xD14)
#define __CCR (__SCB_ADDRESS & __CCR_ADDRESS_OFFSET)
#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)

#define __SYSTICK_ADDRESS_OFFSET (0x10)
#define __SYSTICK_ENABLE_MASK (0x1)
#define __SYSTICK_CLKSOURCE_MASK (0x4)
#define __SYSTICK_TICKINT_MASK (0x2)
#define __SYSTICK (__SCB_ADDRESS & __SYSTICK_ADDRESS_OFFSET)
#define __CORE_CLOCK (48000000) /* running at 48MHz */

#define SysTick_Base_Ptr ((SysTick_Ptr)0xE000E010)
/*
 * A struct to access the SysTick registers
 */
typedef struct {
    volatile uint32_t CSR;      /* SysTick Control and Status Register: 0x0 */
    volatile uint32_t RVR;      /* SysTick Reload Value Register: 0x4 */
    volatile uint32_t CVR;      /* SysTick Current Value Register: 0x8 */
    const uint32_t CALIB;       /* SysTick Calibration Value Register: 0xC */
} volatile *SysTick_Ptr;
/*
 * @brief setup and enable the system tick
 *
 * @return none
 */
void InitSysTick();

/*
 * @brief interrupt handler for the sys tick
 *
 * @return none
 */
void SysTick_Handler();

/*
 * @brief function to get the endianness of a processor
 *
 * This function uses the defines in this file
 * to directly dereference the processors memory
 * in order to figure out the endianness of a processor
 *
 * TODO change uint32_t to a smaller usable type
 *
 * @return uint32_t the endianness of the processor
 *     0 for little endian
 *     1 for big endian
 */
uint32_t inline ARM32_AIRCR_get_endianness_setting();

/*
 * @brief function to return the current stack alignment
 *
 * this function uses a direct memory dereference of the CCR register

```

```

* in order to calculate the stack alignment value
*
* TODO change uint32_t to a smaller usable type
*
* @return uint32_t the stack alignment value
*     0 for 4-byte aligned
*     1 for 8 byte aligned
*/
uint32_t inline ARM32_CCR_get_stack_alignment();

/*
* @brief function to return the CPUID (part number)
*
* This function uses direct memory dereferencing of the
* CPUID register to return the contents.
*
* @return uint32_t the cpuid
*/
uint32_t inline ARM32_CPUID_get_part_number();

/*
* @brief function to write to enable to divide by zero trap
*
* this function uses direct memory dereferencing of the CCR
* register to enable the divide by zero trap
*
* @return void
*/
void inline ARM32_CCR_enable_divide_by_zero_trap();

/*
* @brief function to write to unaligned access trap
*
* this function uses direct memory dereferencing of the CCR
* register to enable the unaligned access trap
*
* @return void
*/
void inline ARM32_CCR_enable_unaligned_access_trap();

/*
* @brief create a trap that gets triggered when an unaligned address is accessed
*
* this function never returns and uses a usage fault exception.
*
* @return void
*/
void inline ARM32_create_unaligned_access_trap();

/*

```



```

* @brief performs a divide by zero in order to trap the microcontroller
*
* this function never returns and uses a usage fault exception.
*
* @return void
*/
void inline ARM32_create_divide_by_zero_trap();

#endif /* __ARCH_ARM32_H__ */
/*****
* @file   core_cm0plus.h
* @brief  CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
* @version V3.30
* @date   24. 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_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 (0x03)                /*!< [31:16] CMSIS HAL main
version */
#define __CM0PLUS_CMSIS_VERSION_SUB (0x20)                /*!< [15:0] CMSIS HAL sub
version */
#define __CM0PLUS_CMSIS_VERSION ((__CM0PLUS_CMSIS_VERSION_MAIN << 16) | \
__CM0PLUS_CMSIS_VERSION_SUB) /*!< 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

```

```

#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__ ) /* Cosmic */
#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. 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 /* __CORE_CM0PLUS_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM0PLUS_H_DEPENDANT
#define __CORE_CM0PLUS_H_DEPENDANT

/* 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

#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
#endif

```

```

#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

```

```

    {

```

```

#if (__CORTEX_M != 0x04)

```

```

    uint32_t reserved0:27; /*!< bit: 0..26 Reserved */

```

```

#else

```

```

    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 */

```

```

#endif

```

```

    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; /*!< Structure used for bit access */

```

```

uint32_t w; /*!< Type used for word access */

```

```

} APSR_Type;

```

```

/** \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;                          /*!< Structure used for bit access    */
    uint32_t w;                  /*!< Type    used for word access     */
} IPSR_Type;

/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
*/
typedef union
{
    struct
    {
        uint32_t ISR:9;          /*!< bit: 0.. 8 Exception number      */
#ifdef __CORTEX_M != 0x04
        uint32_t _reserved0:15;   /*!< bit: 9..23 Reserved              */
#else
        uint32_t _reserved0:7;    /*!< bit: 9..15 Reserved              */
        uint32_t GE:4;           /*!< bit: 16..19 Greater than or Equal flags */
        uint32_t _reserved1:4;    /*!< bit: 20..23 Reserved              */
#endif
        uint32_t T:1;            /*!< bit: 24 Thumb bit (read 0)      */
        uint32_t IT:2;           /*!< bit: 25..26 saved IT state (read 0) */
        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;                          /*!< Structure used for bit access    */
    uint32_t w;                  /*!< Type    used for word access     */
} xPSR_Type;

/** \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;                          /*!< Structure used for bit access    */
    uint32_t w;                  /*!< Type    used for word access     */
} CONTROL_Type;

```

```
/*@} 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[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 RESERVED1[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
    @{
    */
```

```
/** \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 */
    #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 */
    RESERVED /*!< 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          (0xFFFUL << 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_AICR_VECTCLRACTIVE_Pos    1                      /*!< SCB AICR:
VECTCLRACTIVE Position */
#define SCB_AICR_VECTCLRACTIVE_Msk    (1UL << SCB_AICR_VECTCLRACTIVE_Pos)
/*!< SCB AICR: 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 */
__IO 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 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          (0xFFUL << 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          (0xFFFFUL << 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_MPU */
#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)(IRQn) ) & 0x03) * 8 )
```

```
#define _SHP_IDX(IRQn) ( (((uint32_t)(IRQn) & 0x0F)-8) >> 2 )
```

```
#define _IP_IDX(IRQn) ( ((uint32_t)(IRQn) >> 2) )
```

```
/** \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] = (1 << ((uint32_t)(IRQn) & 0x1F));
```

```
}
```

```
/** \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] = (1 << ((uint32_t)(IRQn) & 0x1F));
```

```
}
```

```
/** \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] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0));
```

```
}
```

```
/** \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] = (1 << ((uint32_t)(IRQn) & 0x1F));
```

```
}
```

```
/** \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] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* Clear pending interrupt */
```

```
}
```

```
/** \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(IRQn < 0) {
        SCB->SHP[_SHP_IDX(IRQn)] = (SCB->SHP[_SHP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _BIT_SHIFT(IRQn)); }
    else {
        NVIC->IP[_IP_IDX(IRQn)] = (NVIC->IP[_IP_IDX(IRQn)] & ~(0xFF << _BIT_SHIFT(IRQn))) |
            (((priority << (8 - __NVIC_PRIO_BITS)) & 0xFF) << _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(IRQn < 0) {
        return((uint32_t)(((SCB->SHP[_SHP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & 0xFF) >> (8 -
__NVIC_PRIO_BITS))); } /* get priority for Cortex-M0 system interrupts */
    else {
        return((uint32_t)(((NVIC->IP[_IP_IDX(IRQn)] >> _BIT_SHIFT(IRQn) ) & 0xFF) >> (8 -
__NVIC_PRIO_BITS))); } /* get priority for device specific interrupts */
}

```

/\*\* \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 = ((0x5FA << SCB_AIRCR_VECTKEY_Pos) |
        SCB_AIRCR_SYSRESETREQ_Msk);
    __DSB();                                     /* Ensure completion of memory access */
    while(1);                                    /* 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.
    @{
*/

```

```

#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 - 1) > SysTick_LOAD_RELOAD_Msk) return (1);    /* Reload value impossible */

    SysTick->LOAD = ticks - 1;                                  /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); /* set Priority for SysTick Interrupt */
    SysTick->VAL = 0;                                           /* 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 (0);                                                /* Function successful */
}

#endif

/*@} end of CMSIS_Core_SysTickFunctions */

```

```

#endif /* __CORE_CM0PLUS_H_DEPENDANT */

```

```

#ifdef __cplusplus
}
#endif

```

```

#endif /* __CMSIS_GENERIC */

```

```

/*****
* @file   core_cm4.h
* @brief  CMSIS Cortex-M4 Core Peripheral Access Layer Header File
* @version V3.30
* @date   24. 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_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 (0x03)

/\*!< [31:16] CMSIS HAL main

version \*/

#define \_\_CM4\_CMSIS\_VERSION\_SUB (0x20)

/\*!< [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__ ) /* Cosmic */
    #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
#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

#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

#include <stdint.h>           /* standard types definitions          */
#include <core_cmInstr.h>      /* Core Instruction Access             */
#include <core_cmFunc.h>       /* Core Function Access                */
#include <core_cm4_simd.h>     /* Compiler specific SIMD Intrinsics   */

#endif /* __CORE_CM4_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM4_H_DEPENDANT
#define __CORE_CM4_H_DEPENDANT

/* 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
#endif

#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
    {
#ifdef __CORTEX_M != 0x04
        uint32_t reserved0:27;          /*!< bit: 0..26 Reserved          */
#else
        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          */
#endif
        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;                                /*!< Structure used for bit access */
    uint32_t w;                         /*!< Type used for word access */
} APSR_Type;
```

```
/** \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;                                /*!< Structure used for bit access */
    uint32_t w;                         /*!< Type used for word access */
} IPSR_Type;
```

```
/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
*/
```

```
typedef union
```

```
{
    struct
```



```

{
    uint32_t ISR:9;          /*!< bit: 0.. 8 Exception number          */
#if (__CORTEX_M != 0x04)
    uint32_t _reserved0:15;  /*!< bit: 9..23 Reserved                    */
#else
    uint32_t _reserved0:7;   /*!< bit: 9..15 Reserved                    */
    uint32_t GE:4;          /*!< bit: 16..19 Greater than or Equal flags */
    uint32_t _reserved1:4;   /*!< bit: 20..23 Reserved                    */
#endif
    uint32_t T:1;           /*!< bit: 24 Thumb bit (read 0)             */
    uint32_t IT:2;          /*!< bit: 25..26 saved IT state (read 0)     */
    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;                       /*!< Structure used for bit access          */
uint32_t w;                /*!< Type used for word access              */
} xPSR_Type;

/** \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;                     /*!< Structure used for bit access              */
    uint32_t w;              /*!< Type used for word access                  */
} CONTROL_Type;

/*@} 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];
__IO 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) Vector 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) System Handlers Priority Registers (4-7, 8-11,
12-15) */
    __IO uint32_t SHCSR;         /*!< Offset: 0x024 (R/W) System 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) Debug 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/ ) Debug 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         (0xFFFUL << 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     (0x1FFUL << 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         (0x1FFFFFFFUL << 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       (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_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) /*!< 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_BFHFNMIGN_Pos         8                        /*!< SCB CCR: BFHFNMIGN
Position */
#define SCB_CCR_BFHFNMIGN_Msk         (1UL << SCB_CCR_BFHFNMIGN_Pos) /*!< SCB
CCR: BFHFNMIGN Mask */

#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) /*!< SCB
CCR: DIV_0_TRP 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 */

#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_BUSFAULTPENDEDED_Pos 14                  /*!< SCB SHCSR:
BUSFAULTPENDEDED Position */
#define SCB_SHCSR_BUSFAULTPENDEDED_Msk (1UL << SCB_SHCSR_BUSFAULTPENDEDED_Pos) /*!< SCB SHCSR: BUSFAULTPENDEDED Mask */

#define SCB_SHCSR_MEMFAULTPENDEDED_Pos 13                  /*!< SCB SHCSR:
MEMFAULTPENDEDED Position */
#define SCB_SHCSR_MEMFAULTPENDEDED_Msk (1UL << SCB_SHCSR_MEMFAULTPENDEDED_Pos) /*!< SCB SHCSR: MEMFAULTPENDEDED Mask */

#define SCB_SHCSR_USGFAULTPENDEDED_Pos 12                  /*!< SCB SHCSR:
USGFAULTPENDEDED Position */
#define SCB_SHCSR_USGFAULTPENDEDED_Msk (1UL << SCB_SHCSR_USGFAULTPENDEDED_Pos) /*!< SCB SHCSR: USGFAULTPENDEDED Mask */

#define SCB_SHCSR_SYSTICKACT_Pos      11                    /*!< SCB SHCSR:

```



```

Position */
#define SCB_HFSR_DEBUG EVT_Msk          (1UL << SCB_HFSR_DEBUG EVT_Pos)          /*!< SCB
HFSR: DEBUG EVT Mask */

#define SCB_HFSR_FORCED_Pos          30          /*!< SCB HFSR: FORCED Position
*/
#define SCB_HFSR_FORCED_Msk          (1UL << SCB_HFSR_FORCED_Pos)          /*!< SCB
HFSR: FORCED Mask */

#define SCB_HFSR_VECTTBL_Pos          1          /*!< SCB HFSR: VECTTBL
Position */
#define SCB_HFSR_VECTTBL_Msk          (1UL << SCB_HFSR_VECTTBL_Pos)          /*!< SCB
HFSR: VECTTBL Mask */

/* 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)          /*!< SCB
DFSR: EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos          3          /*!< SCB DFSR: VCATCH Position
*/
#define SCB_DFSR_VCATCH_Msk          (1UL << SCB_DFSR_VCATCH_Pos)          /*!< SCB
DFSR: VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos          2          /*!< SCB DFSR: DWTTRAP
Position */
#define SCB_DFSR_DWTTRAP_Msk          (1UL << SCB_DFSR_DWTTRAP_Pos)          /*!< SCB
DFSR: DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos          1          /*!< SCB DFSR: BKPT Position */
#define SCB_DFSR_BKPT_Msk          (1UL << SCB_DFSR_BKPT_Pos)          /*!< SCB DFSR:
BKPT Mask */

#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];
__IO 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_DISOOF_Pos        9          /*!< ACTLR: DISOOF Position */
#define SCnSCB_ACTLR_DISOOF_Msk        (1UL << SCnSCB_ACTLR_DISOOF_Pos) /*!<
ACTLR: DISOOF 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 */
    __IO 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_VAL_CURRENT_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;           /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit */
        __O uint16_t u16;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit */
        __O uint32_t u32;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit */
    } PORT [32];                 /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers */
    uint32_t RESERVED0[864];
    __IO uint32_t TER;            /*!< Offset: 0xE00 (R/W) ITM Trace Enable Register */
    uint32_t RESERVED1[15];
    __IO uint32_t TPR;            /*!< Offset: 0xE40 (R/W) ITM Trace Privilege Register */
    uint32_t RESERVED2[15];
    __IO uint32_t TCR;            /*!< Offset: 0xE80 (R/W) ITM Trace Control Register */
    uint32_t RESERVED3[29];
    __O uint32_t IWR;             /*!< Offset: 0xEF8 ( /W) ITM Integration Write Register */
    __I uint32_t IRR;             /*!< Offset: 0xEFC (R/ ) ITM Integration Read Register */
    __IO uint32_t IMCR;           /*!< Offset: 0xF00 (R/W) ITM Integration Mode Control Register */
    uint32_t RESERVED4[43];
    __O uint32_t LAR;             /*!< Offset: 0xFB0 ( /W) ITM Lock Access Register */
    __I uint32_t LSR;             /*!< Offset: 0xFB4 (R/ ) ITM Lock Status Register */
    uint32_t RESERVED5[6];
    __I uint32_t PID4;            /*!< Offset: 0xFD0 (R/ ) ITM Peripheral Identification Register #4 */
    __I uint32_t PID5;            /*!< Offset: 0xFD4 (R/ ) ITM Peripheral Identification Register #5 */
    __I uint32_t PID6;            /*!< Offset: 0xFD8 (R/ ) ITM Peripheral Identification Register #6 */
    __I uint32_t PID7;            /*!< Offset: 0xFDC (R/ ) ITM Peripheral Identification Register #7 */
    __I uint32_t PID0;            /*!< Offset: 0xFE0 (R/ ) ITM Peripheral Identification Register #0 */
    __I uint32_t PID1;            /*!< Offset: 0xFE4 (R/ ) ITM Peripheral Identification Register #1 */
    __I uint32_t PID2;            /*!< Offset: 0xFE8 (R/ ) ITM Peripheral Identification Register #2 */
    __I uint32_t PID3;            /*!< Offset: 0xFEC (R/ ) ITM Peripheral Identification Register #3 */
    __I uint32_t CID0;            /*!< Offset: 0xFF0 (R/ ) ITM Component Identification Register #0 */
    __I uint32_t CID1;            /*!< Offset: 0xFF4 (R/ ) ITM Component Identification Register #1 */
    __I uint32_t CID2;            /*!< Offset: 0xFF8 (R/ ) ITM Component Identification Register #2 */
    __I uint32_t CID3;            /*!< Offset: 0xFFC (R/ ) ITM 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_ATVALIDM_Pos       0                      /*!< ITM IWR: ATVALIDM
Position */
#define ITM_IWR_ATVALIDM_Msk       (1UL << ITM_IWR_ATVALIDM_Pos) /*!< ITM
IWR: ATVALIDM 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	*/
__IO uint32_t PCSR;	/*!< Offset: 0x01C (R/W) Program Counter Sample Register	*/
__IO uint32_t COMP0;	/*!< 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_SLEPCNT_SLEPCNT_Pos    0                      /*!< DWT SLEPCNT:
SLEPCNT Position */
#define DWT_SLEPCNT_SLEPCNT_Msk    (0xFFUL << DWT_SLEPCNT_SLEPCNT_Pos) /*!< DWT SLEPCNT: SLEPCNT 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) */
```

```

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_PTINVALID_Pos         9                                     /*!< TPI DEVID: PTINVALID
Position */
#define TPI_DEVID_PTINVALID_Msk        (0x1UL << TPI_DEVID_PTINVALID_Pos)        /*!< TPI
DEVID: PTINVALID 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_SubType_Pos      0                /*!< TPI DEVTYPE: SubType
Position */
#define TPI_DEVTYPE_SubType_Msk      (0xFUL << TPI_DEVTYPE_SubType_Pos)    /*!< TPI
DEVTYPE: SubType Mask */

#define TPI_DEVTYPE_MajorType_Pos     4                /*!< TPI DEVTYPE: MajorType
Position */
#define TPI_DEVTYPE_MajorType_Msk     (0xFUL << TPI_DEVTYPE_MajorType_Pos)    /*!< TPI
DEVTYPE: MajorType 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
{
    __IO 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        (0xFFUL << 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         (0x7FFFFFFFUL << 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          (0xFFFFUL << 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_MPU */
#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_ASPEN_Pos      31          /*!< FPCCR: ASPEN bit Position */
#define FPU_FPCCR_ASPEN_Msk      (1UL << FPU_FPCCR_ASPEN_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 FPU_FPCAR_ADDRESS_Pos   3                      /*!< FPCAR: ADDRESS bit
Position */
#define FPU_FPCAR_ADDRESS_Msk   (0x1FFFFFFFFUL << FPU_FPCAR_ADDRESS_Pos)      /*!<
FPCAR: ADDRESS bit Mask */

/* Floating-Point Default Status Control Register */
#define FPU_FPDSCR_AHP_Pos      26                     /*!< FPDSCR: AHP bit Position */
#define FPU_FPDSCR_AHP_Msk      (1UL << FPU_FPDSCR_AHP_Pos)      /*!< FPDSCR:
AHP bit Mask */

#define FPU_FPDSCR_DN_Pos       25                     /*!< FPDSCR: DN bit Position */
#define FPU_FPDSCR_DN_Msk       (1UL << FPU_FPDSCR_DN_Pos)      /*!< FPDSCR: DN
bit Mask */

#define FPU_FPDSCR_FZ_Pos       24                     /*!< FPDSCR: FZ bit Position */
#define FPU_FPDSCR_FZ_Msk       (1UL << FPU_FPDSCR_FZ_Pos)      /*!< FPDSCR: FZ bit
Mask */

#define FPU_FPDSCR_RMode_Pos    22                     /*!< FPDSCR: RMode bit Position
*/
#define FPU_FPDSCR_RMode_Msk    (3UL << FPU_FPDSCR_RMode_Pos)      /*!< FPDSCR:
RMode bit Mask */

/* Media and FP Feature Register 0 */
#define FPU_MVFR0_FP_rounding_modes_Pos  28             /*!< MVFR0: FP rounding
modes bits Position */
#define FPU_MVFR0_FP_rounding_modes_Msk   (0xFUL << FPU_MVFR0_FP_rounding_modes_Pos)      /*!< MVFR0: FP rounding modes bits Mask */

#define FPU_MVFR0_Short_vectors_Pos  24                 /*!< MVFR0: Short vectors bits
Position */
#define FPU_MVFR0_Short_vectors_Msk      (0xFUL << FPU_MVFR0_Short_vectors_Pos)      /*!<
MVFR0: Short vectors bits Mask */

#define FPU_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 */
    __IO 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_NOCERR_Pos   5                               /*!< CoreDebug DEMCR:
```

```
VC_NOCERR Position */
```

```
#define CoreDebug_DEMCR_VC_NOCERR_Msk   (1UL << CoreDebug_DEMCR_VC_NOCERR_Pos)
```

```
/*!< CoreDebug DEMCR: VC_NOCERR 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)0x07);          /* only values 0..7 are used */

    reg_value = SCB->AIRCR;          /* read old register configuration */
    reg_value &= ~(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk);      /* clear bits to
change */
    reg_value = (reg_value |
                ((uint32_t)0x5FA << 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 ((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos); /* read
priority grouping field */
}

```

/\*\* \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)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); enable interrupt */
NVIC->ISER[(uint32_t)((int32_t)IRQn) >> 5] = (uint32_t)(1 << ((uint32_t)((int32_t)IRQn) &
(uint32_t)0x1F)); /* enable interrupt */
}

```

/\*\* \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)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* disable interrupt */
}

```

/\*\* \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)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0)); /* Return
1 if pending else 0 */
}

```

/\*\* \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)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* set interrupt pending */
}

```

/\*\* \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)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* Clear pending interrupt */
}

```

/\*\* \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)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) & 0x1F)))?1:0)); /* Return
1 if active else 0 */
}

```

/\*\* \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(IRQn < 0) {
        SCB->SHPR[((uint32_t)(IRQn) & 0xF)-4] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); } /* set Priority
for Cortex-M System Interrupts */
    else {
        NVIC->IP[(uint32_t)(IRQn)] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff); } /* set Priority for
device specific Interrupts */
}

```

/\*\* \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(IRQn < 0) {
        return((uint32_t)(SCB->SHP[(((uint32_t)(IRQn) & 0xF)-4] >> (8 - __NVIC_PRIO_BITS)))); } /* get priority
for Cortex-M system interrupts */
    else {
        return((uint32_t)(NVIC->IP[(((uint32_t)(IRQn) >> (8 - __NVIC_PRIO_BITS)))); } /* get priority for
device specific interrupts */
    }
}
```

/\*\* \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 & 0x07); /* only values 0..7 are used */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS : 7 -
PriorityGroupTmp;
    SubPriorityBits = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 : PriorityGroupTmp - 7 +
__NVIC_PRIO_BITS;

    return (
        ((PreemptPriority & ((1 << (PreemptPriorityBits)) - 1)) << SubPriorityBits) |
        ((SubPriority & ((1 << (SubPriorityBits)) - 1)))
    );
}
```

/\*\* \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 & 0x07);          /* only values 0..7 are used          */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS : 7 -
PriorityGroupTmp;
    SubPriorityBits    = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 : PriorityGroupTmp - 7 +
__NVIC_PRIO_BITS;

    *pPreemptPriority = (Priority >> SubPriorityBits) & ((1 << (PreemptPriorityBits)) - 1);
    *pSubPriority     = (Priority
                        ) & ((1 << (SubPriorityBits  )) - 1);
}

/** \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 = ((0x5FA << 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);                      /* 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.

    @{
*/

#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 - 1) > SysTick_LOAD_RELOAD_Msk) return (1);    /* Reload value impossible */

    SysTick->LOAD = ticks - 1;                                /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); /* set Priority for SysTick Interrupt */
    SysTick->VAL = 0;                                          /* 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 (0);                                              /* Function successful */
}

#endif

/*@} end of CMSIS_Core_SysTickFunctions */


/* ##### Debug In/Output function ##### */
/* ##### */
/** \ingroup CMSIS_Core_FunctionInterface
    \defgroup CMSIS_core_DebugFunctions ITM Functions
    \brief Functions that access the ITM debug interface.
    @{
    */

extern volatile int32_t ITM_RxBuffer;    /*!< External variable to receive characters. */
#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< Value identifying \ref ITM_RxBuffer is
ready for next character. */

/** \brief ITM Send Character

    The function transmits a character via the ITM channel 0, and
    \li Just returns when no debugger is connected that has booked the output.
    \li Is blocking when a debugger is connected, but the previous character sent has not been transmitted.
```

\param [in] ch Character to transmit.

\returns Character to transmit.

\*/

```
__STATIC_INLINE uint32_t ITM_SendChar (uint32_t ch)
{
    if ((ITM->TCR & ITM_TCR_ITMENA_Msk) && /* ITM enabled */
        (ITM->TER & (1UL << 0)) ) /* ITM Port #0 enabled */
    {
        while (ITM->PORT[0].u32 == 0);
        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) {
        return (0); /* no character available */
    } else {
        return (1); /* character available */
    }
}
```

/\*@} end of CMSIS\_core\_DebugFunctions \*/

```

#endif /* __CORE_CM4_H_DEPENDANT */

#ifdef __cplusplus
}
#endif

#endif /* __CMSIS_GENERIC */
/*****
 * @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.

-----\*/

```

#ifdef ( __ICCARM__ )
#pragma system_include /* treat file as system include file for MISRA check */
#endif

```

```

#ifndef __CORE_CM4_SIMD_H
#define __CORE_CM4_SIMD_H

```

```
#ifndef __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        __smlaldx
#define __SMUSD          __smusd
#define __SMUSDX         __smusdx
#define __SMLSD          __smlsd
#define __SMLSDX         __smlsdx
#define __SMLSLD         __smlsld
#define __SMLSLDX        __smlsldx
#define __SEL            __sel
#define __QADD           __qadd
#define __QSUB           __qsub

```

```

#define __PKHBT(ARG1,ARG2,ARG3)      ( (((uint32_t)(ARG1))          ) & 0x0000FFFFUL) | \
                                         (((uint32_t)(ARG2)) << (ARG3)) & 0xFFFF0000UL )

```

```

#define __PKHTB(ARG1,ARG2,ARG3)      ( (((uint32_t)(ARG1))          ) & 0xFFFF0000UL) | \
                                         (((uint32_t)(ARG2)) >> (ARG3)) & 0x0000FFFFUL )

```

```

#define __SMMLA(ARG1,ARG2,ARG3)      ( (int32_t)((((int64_t)(ARG1) * (ARG2)) + \
                                         ((int64_t)(ARG3) << 32)          ) >> 32))

```

```

#ifdef __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;

```

```

#ifdef __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;

```

```

#ifdef __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 __SMUSD (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;

#ifdef __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;

#ifdef __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 */

#ifdef __cplusplus
}
#endif

#endif /* __CORE_CM4_SIMD_H */
/*****
 * @file    core_cmFunc.h
 * @brief   CMSIS Cortex-M Core Function Access 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.
```

```
-----*/
```

```
#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)

/\*\* \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 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) */

```

```

#if    (__CORTEX_M == 0x04)

```

```

/** \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_FPSR(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) */

```

```

#elif defined ( __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_max" : "=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 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)

/\*\* \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_FPSR(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 ) */

#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 */
/*
 * 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

/*@} end of CMSIS_Core_RegAccFunctions */

#endif /* __CORE_CMFUNC_H */
/*****
 * @file    core_cmInstr.h
 * @brief   CMSIS Cortex-M Core Instruction Access 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.

-----\*/

```

#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()              __isb(0xF)
```

```
/** \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()          __dsb(0xF)
```

/\*\* \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()          __dmb(0xF)
```

/\*\* \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)
```

```
#if    (__CORTEX_M >= 0x03)
```

```
/** \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
*/
#define __RBIT                                __rbit
```

```
/** \brief LDR Exclusive (8 bit)
```

This function performs a exclusive LDR command 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 performs a exclusive LDR command for 16 bit values.

```

\param [in] ptr Pointer to data
\return value of type uint16_t at (*ptr)
*/
#define __LDREXH(ptr) ((uint16_t) __ldrex(ptr))

```

/\*\* \brief LDR Exclusive (32 bit)

This function performs a exclusive LDR command for 32 bit values.

```

\param [in] ptr Pointer to data
\return value of type uint32_t at (*ptr)
*/
#define __LDREXW(ptr) ((uint32_t) __ldrex(ptr))

```

/\*\* \brief STR Exclusive (8 bit)

This function performs a exclusive STR command 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 performs a exclusive STR command 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 __STREXH(value, ptr) __strex(value, ptr)

```

/\*\* \brief STR Exclusive (32 bit)

This function performs a exclusive STR command 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 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  
  
#endif /* (__CORTEX_M >= 0x03) */
```

```
#elif defined ( __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");
}
```

/\*\* \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");
}
```

/\*\* \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");
}
```

/\*\* \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)
```

```
#if (__CORTEX_M >= 0x03)
```

```
/** \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;  
  
    __ASM volatile ("rbit %0, %1" : "=r" (result) : "r" (value) );  
    return(result);  
}
```

```
/** \brief LDR Exclusive (8 bit)
```

This function performs a exclusive LDR command 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 ("ldrex %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 ("ldrex %0, [%1]" : "=r" (result) : "r" (addr) : "memory" );  
    #endif  
    return ((uint8_t) result); /* Add explicit type cast here */  
}
```

```
/** \brief LDR Exclusive (16 bit)
```

This function performs a exclusive LDR command 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 performs a exclusive LDR command 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 performs a exclusive STR command 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 ("strexh %0, %2, %1" : "=&r" (result), "=Q" (*addr) : "r" ((uint32_t)value) );
    return(result);
}

```

/\*\* \brief STR Exclusive (16 bit)

This function performs a exclusive STR command 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 ("strexh %0, %2, %1" : "=&r" (result), "=Q" (*addr) : "r" ((uint32_t)value) );
    return(result);
}
```

/\*\* \brief STR Exclusive (32 bit)

This function performs a exclusive STR command 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 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
*/
__attribute__((always_inline)) __STATIC_INLINE uint8_t __CLZ(uint32_t value)
{
    uint32_t result;

    __ASM volatile ("clz %0, %1" : "=r" (result) : "r" (value) );
    return ((uint8_t) result); /* Add explicit type cast here */
}

#endif /* (__CORTEX_M >= 0x03) */

```

```

#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 carm 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

/*@} */ /* end of group CMSIS_Core_InstructionInterface */

#endif /* __CORE_CMINSTR_H */
/**
* @file cirbuf.h
* @brief function and structure definitions for circular buffer
*
* this file contains the structure of a circular buffer, along with access
* abstractions. This circular buffer is meant to be utilized for a UART
* communication system over GPIO pins
*
* Should be interrupt safe by utilizing the volatile keyword, and other
* safeguard techniques
*
* @author Seth Miers and Jake Cazden
* @date March 04, 2018
*
* TODO should all size_ts be ints?
*
*/

#ifndef __CIRCBUF_H__
#define __CIRCBUF_H__

#include <stdint.h> /* for std ints: uint8_t, uint32_t */
#include <stddef.h> /* for size_t */

#ifdef KL25Z
#include "MKL25Z4.h"
#endif

/*

```

```

* quickly change what the type of the buffer is
*/
#define BUFFER_TYPE uint8_t

/*
* The KL25Z needs to specifically disable interrupts
*/
#ifndef KL25Z
#define START_CRITICAL() __enable_irq()
#define END_CRITICAL() __disable_irq()
#else
#define START_CRITICAL()
#define END_CRITICAL()
#endif

/*
* A structure to implement a FIFO buffer
* TODO need to understand volatile should it be used here?
*/
typedef struct {
    volatile BUFFER_TYPE* base;           // The base address of the buffer
    volatile size_t buff_size;           // the total size of the buffer
    volatile BUFFER_TYPE* head;          // Where next byte should be put in the buff
    volatile BUFFER_TYPE* tail;          // Oldest byte in buff
    volatile size_t num_in;              // Number of bytes in buff
    volatile uint8_t buff_empty_flag:1;  // Flag to show if the buffer is empty
    volatile uint8_t buff_full_flag:1;   // Flag to show if the buffer is full
    volatile uint8_t buff_ovf_flag:1;    // Flag to show that the buffer has overflowed
    volatile uint8_t buff_destroyed_flag:1; // Flag to show that the buffer has been freed
} CB_t;

/*
* An enumerator to allow for readability of errors
*/
typedef enum {
    SUCCESS,
    BAD_POINTER,
    NO_LENGTH,
    POSITION_TOO_LARGE,
    DOUBLE_FREE,
    NO_BUFFER_IN_MEMORY,
    FULL,
    EMPTY,
    CRITICAL_ERROR
} CB_e;

/*
* An enumerator to set and unset flags
*/
typedef enum {
    UNSET = 0,

```

```
    SET = 1
} CB_f;
```

```
/* @brief initializes the circular buffer with default values
 *
 * the circularbuffer must already be allowcated when it is
 * passed in. The function will then create dynamic memory
 * to allocate the entire buffer. It will then set the
 * head, tail, and other values to be what they should be.
 * destroy must be called before free-ing the CB that
 * is passed in, otherwise a stack overflow may occur.
 *
 *
 * @param[in] CB_t* the CB_t object to initialize
 * @param[in] size_t the size of the buffer to initialize
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
```

```
CB_e CB_init(CB_t* circbuff, size_t buffer_size);
```

```
/* @brief frees the entire circular buffer
 *
 * this function simply frees the dynamic memory that it allowcated
 * when initializing the function.
 *
 * @param[in] CB_t* the CB_t object to destroy
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
```

```
CB_e CB_destroy(CB_t* circbuff);
```

```
/* @brief adds an item to the circular buffer
 *
 * add an item in memory tot he circular buffer
 * increment the head, and update all the flags
 *
 *
 * @param[in] CB_t* the CB_t object to operate on
 * @param[in] BUFFER_TYPE the object to add into the buffer
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
```

```
CB_e CB_buffer_add_item(CB_t* circbuff, BUFFER_TYPE data);
```

```
/* @brief removes an item from the circular buffer
 *
 * removes an item from the circular buffer by
 * incrementing the tail an decrementing the num_in
 *
 *
 * @param[in] CB_t* the CB_t object to operate on
 * @param[out] BUFFER_TYPE* put the data removed into this pointer
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
```

```
CB_e CB_buffer_remove_item(CB_t* circbuff, BUFFER_TYPE* data);
```

```

/* @brief checks to see if the buffer is full
 *
 * simply checks the full flag of the buffer to see
 * if the head ever passed the tail
 *
 * @param[in] CB_t# the CB_t object to operate on
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
CB_e CB_is_full(CB_t* circbuff);

/* @brief checks to see if the buffer is empty
 *
 * simply checks the empty flag of the buffer
 * to see if the head and tail are equal and
 * there is no data in the buffer
 *
 *
 * @param[in] CB_t* the CB_t object to operate on
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
CB_e CB_is_empty(CB_t* circbuff);

/* @brief returns the value in the buffer at a position back from the head
 *
 * peeking at the 0th value will just return the last value
 * that was put into the buffer.
 *
 * @param[in] CB_t* the CB_t object to operate on
 * @param[in] size_t the index away from the head
 * @param[out] BUFFER_TYPE* put the data peeked at into this pointer
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */
CB_e CB_peek(CB_t* circbuff, size_t position, BUFFER_TYPE* data);

#endif /* __CIRCBUF_H__ */

/*
 * @file conversion.h
 * @brief this file contains implementations of integer to character conversion
 *
 * This contins functions for integer to character array conversion in mulitple
 * bases, as well as an exponentiation function (not optimized), primarily
 * for use with these functions. it also defines offsets into the ASCII table
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 *
 * */
#ifndef __CONVERSION_H__
#define __CONVERSION_H__

/* needed for function prototypes */
#include <stdint.h>

```



```

#define BASE_2 (2)
#define BASE_3 (3)
#define BASE_4 (4)
#define BASE_5 (5)
#define BASE_6 (6)
#define BASE_7 (7)
#define BASE_8 (8)
#define BASE_9 (9)
#define BASE_10 (10)
#define BASE_11 (11)
#define BASE_12 (12)
#define BASE_13 (13)
#define BASE_14 (14)
#define BASE_15 (15)
#define BASE_16 (16)

#define BASE_2_MAXDIGITS (32)

#ifndef __ASCII_OFFSETS__
#define __ASCII_OFFSETS__
#define ASCII_OFFSET_0 (48)
#define ASCII_OFFSET_9 (57)
#define ASCII_OFFSET_A (65)
#define ASCII_OFFSET_Z (90)
#define ASCII_OFFSET_A_ADDITION (55)*10+55= 'A' for hex interpreting*/
#define ASCII_OFFSET_F (70)
#define ASCII_OFFSET_LA (97)
#define ASCII_OFFSET_LZ (122)
#define ASCII_OFFSET_LA_ADDITION (87)*10+87= 'a' for hex interpreting*/
#define ASCII_OFFSET_LF (102)
#define ASCII_OFFSET_EOF (4)
#endif

/**
 * @brief take the exponent of a number
 *
 * This function takes in a number (base)
 * and raises it to an exponent (power)
 *
 * @param base is equal to X in the expression X^Y (where ^ is exponent)
 * @param power is equal to Y in the expression X^Y (where ^ is exponent)
 * @return int8_t the result of the exponent
 */
int32_t exponent(int32_t base,int32_t power);

/**
 * @brief function to convert an integer to ascii equivalent
 *
 * This function takes in an integer and converts it
 * to an array of bytes that represent the ascii
 * value of that number in a particular base. A null

```

```

* terminator '\0' is placed at the end of the string.
* A negative sign is included at the beginning of the
* string if needed.
*
* The entire function is done using only
* pointer arithmetic.
*
* @param data the 32 bit signed integer that we wish to convert
* @param ptr is a pointer to an array of bytes that will be written to
*      this pointer must be at least 32 bytes long
* @param base is the base we want to convert to, bases 2-16 are supported
* @return uint8_t the length of the string (including the minus sign)
*/
uint8_t my_itoa(int32_t data, uint8_t * ptr, uint32_t base);

/**
* @brief function to convert an ascii string into an integer
*
* This function takes in an array of data (ptr) and
* converts it to the 4byte equivalent. It is assumed
* that the array of ascii formatted bytes (string) that
* is passed in is null '\0' terminated. However,
* the number of digits in this string is also passed in.
* The function can convert anything of base 2 to 16.
* The first character of the string may be a '-' to
* indicate that it is a negative number.
*
* The entire function is done using only
* pointer arithmetic.
*
* @param ptr a pointer to the array of bytes that store ascii characters
* @param digits the number of characters in the string
* @param base is the base we want to convert from, bases 2-16 are supported
* @return int8_t the converted number
*/
int32_t my_atoi(uint8_t * ptr, uint8_t digits, uint32_t base);

#endif /* __CONVERSION_H__ */

/**
* @file data.h
* @brief functions to examine platform specific data type behavior
*
* contains functions for printing system specific type sizes, as well as
* manually determining endianness, and swapping endianness on a given piece
* of data, rendered as a byte array
*
* @author Seth Miers and Jake Cazden
* @date February 11, 2018
*
*/

```

```
#ifndef __DATA_H__
#define __DATA_H__
```

```
/* Type definitions needed for function prototypes */
```

```
#include <stdint.h>
```

```
#include <stddef.h>
```

```
#define LITTLE_ENDIAN 0
```

```
#define BIG_ENDIAN 1
```

```
#define SWAP_NO_ERROR 0
```

```
#define SWAP_ERROR -1
```

```
/**
 * @brief function prints the size of c standard types
 *
 * this function calls sizeof, and stores the values in a temporary size_t
 * the size_t is reused for each call to sizeof
 * utilizes platform independent PRINTF macro to print information on variable
 * type and size
 * reports on
 *   char, short, int, long, double, float,
 *   unsigned char, unsigned int, unsigned long,
 *   signed char, signed int, signed long
 *
 * @param this takes in no parameters
 * @return void returns nothing
 */
```

```
void print_cstd_type_sizes();
```

```
/**
 * @brief this function prints the stdint type sizes
 *
 * this function calls sizeof, and stores the values in a temporary size_t
 * the size_t is reused for each call to sizeof
 * utilizes platform independent PRINTF macro to print information on variable
 * type and size
 * reports on
 *   int8_t, uint8_t, uint16_t, int32_t, uint32_t
 *   uint_fast8_t, uint_fast16_t, uint_fast32_t
 *   uint_least8_t, uint_least16_t, uint_least32_t
 *   size_t, ptrdiff_t
 *
 * @param this takes in no parameters
 * @return void returns nothing
 */
```

```
void print_stdint_type_sizes();
```

```
/**
 * @brief this function prints pointer sizes for a variety of types
 *
 * this function calls sizeof, and stores the values in a temporary size_t
 * the size_t is reused for each call to sizeof
 */
```

```

* utilizes platform independent PRINTF macro to print information on variable
* type and size
* reports on
*   char *, short *, int *, double *, float *, void *,
*   int8_t *, int16_t *, int32_t *,
*   char **, int **, void **
*
* @param this takes in no parameters
* @return void returns nothing
*/
void print_pointer_sizes();

/**
* @brief this function reverses the endianness of a datatype
*
* this function uses my_reverse to swap a piece of data's bitwise endianness
* it then returns a success or failure code accordingly
*
* returns either SWAP_NO_ERROR or SWAP_ERROR
*
* @param data is a pointer to the first byte of the data input
* @param type_length holds the length of the data as with sizeof(data)
* @return int32_t a macro return code signifying swap success or failure;
*/
int32_t swap_data_endianness(uint8_t * data, size_t type_length);

/**
* @brief this function determines the endianness of the system
*
* this function manually creates a multibyte variable, then determines
* if it is stored in big or little endian format
*
* @param this function takes in no parameters
* @return uint32_t returns BIG_ENDIAN=1 or LITTLE_ENDIAN=0
*/
uint32_t determine_endianness();
#endif /* __DATA_H__ */

/**
* @file debug.h
* @brief contains functions for debug printing of memory
*
* contains a function to print hex from memory for a specified length
*
* @author Seth Miers and Jake Cazden
* @date February 11, 2018
*
*/
#ifdef __DEBUG_H__

/* needed for function prototypes */

```

```

#include <stdint.h>

/*
 * @brief function to print the bytes in memory starting at an address
 *
 * This function takes in a pointer to the start of some
 * memory region and starts printing out hex bytes
 * for the number of bytes specified.
 *
 * @param start a pointer to the start of the memory region
 * @param length the number of bytes to read back
 * @return void don't return anything
 */
void print_array(void * start, uint32_t length);

#endif /* __DEBUG_H__ */
/**
 * @file memory.h
 * @brief contains functions for allocating, freeing, and copying memory
 *
 * this contains numerous functions for the allocation, movement, and freeing
 * of memory, utilizing software, not hardware specific functionality.
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 */
#ifndef __MEMORY_H__
#define __MEMORY_H__

/* Type definitions needed for function prototypes */
#include <stdint.h>
#include <stddef.h>

#define DMAMUX_CLOCKGATE_ENABLE (1)
#define DMA_CLOCKGATE_ENABLE (1)
#define DMAMUX_CHCFG_SINGLETRIGGER (0)
#define DMAMUX_CHCFG_DISABLE (0)
#define DMAMUX_CHCFG_ENABLE (1)
#define DMAMUX_CHCFG_SOURCE_ALWAYS_ON_60 (0x3C)
#define DMA_DSR_BCR_DONE_WRTETOCLEAR (1)
#define DMA_DSR_BCR_BCRMAXVALUE (0x0FFFFFFF)
#define DMA_DCR_INTERRUPT_ON_COMPLETE (1)
#define DMA_DCR_NO_PERIPHERAL_REQUEST (0)
#define DMA_DCR_CONTINUOUS_OPERATION (0)
#define DMA_DCR_NO_AUTOALIGN (0)
#define DMA_DCR_AUTOALIGN (1)
#define DMA_DCR_NO_ASYNC_REQUESTS (0)
#define DMA_DCR_NO_SOURCE_INCREMENT (0)
#define DMA_DCR_INCREMENT_SOURCE (1)
#define DMA_DCR_TRANSFERSIZE_32BIT (0)
#define DMA_DCR_TRANSFERSIZE_8BIT (1)

```

```

#define DMA_DCR_TRANSFERSIZE_16BIT    (2)
#define DMA_DCR_NO_DEST_INCREMENT    (0)
#define DMA_DCR_INCREMENT_DEST      (1)
#define DMA_DCR_START_DISABLE      (0)
#define DMA_DCR_START_ENABLE      (1)
#define DMA_DCR_NO_SOURCE_MODULO    (0)
#define DMA_DCR_NO_DEST_MODULO      (0)
#define DMA_DCR_DISABLE_REQUEST_OFF (0)
#define DMA_DCR_CHANNEL_LINK_DISABLED (0)

```

```

typedef enum {
    DMA_SUCCESS,
    DMA_BAD_INDEX,
    DMA_BAD_POINTER,
    DMA_NO_LENGTH,
    DMA_BCR_LENGTH_OVERFLOW,
    DMA_BAD_SIZE,
    DMA_BUSY,
    DMA_ERROR
} DMA_e;

```

```

/**
 * @brief function to set an array of bytes all to the same value using DMA.
 *
 * This function takes in a pointer to an array of bytes
 * then for a given number of bytes, writes a value to each
 * of those bytes. This function uses DMA requests
 * to do the writes. This is designed to work on 1, 2, and 4
 * byte transfer sizes. This will trigger an interrupt when completed.
 *
 * This is currently only supported for the KL25Z FRDM Board
 *
 * @param src a pointer to the array of bytes that we will write over
 * @param length the number of bytes to iterate through
 * @param value the value to write to every byte in the array
 * @return uint8_t a pointer to the source (simply returns src)
 */
uint8_t * memset_dma(uint8_t * src, size_t length, uint8_t value, size_t transfer);

```

```

/**
 * @brief function to copy one byte array to another (overlap) using DMA.
 *
 * This function takes in 2 pointers to byte arrays.
 * A source and a destination pointer are provided
 * to the function as well as the length of the source bytes
 * so the function knows how many to move over.
 * This function handles overlap so that the data moved
 * does not become corrupted. this function uses DMA requests
 * to do the memory moves. This is designed to work on 1, 2, and 4
 * byte transfer sizes. This will trigger an interrupt when completed.
 *
 * This is currently only supported for the KL25Z FRDM Board

```

```

*
* @param src a pointer to the array of bytes that we will move from
* @param dst a pointer to the array of bytes that we will move to
* @param length the number of bytes to move
* @return uint8_t a pointer to the destination
*/
uint8_t * memmove_dma(uint8_t * src, uint8_t * dst, size_t length, size_t transfer);

```

```

/**
* @brief function to copy one byte array to another (overlap)
*
* This function takes in 2 pointers to byte arrays.
* A source and a destination pointer are provided
* to the function as well as the length of the source bytes
* so the function knows how many to move over.
* This function handles overlap so that the data moved
* does not become corrupted.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the array of bytes that we will move from
* @param dst a pointer to the array of bytes that we will move to
* @param length the number of bytes to move
* @return uint8_t a pointer to the destination
*/
uint8_t * my_memmove(uint8_t * src, uint8_t * dst, size_t length);

```

```

/**
* @brief function to copy one byte array to another (no overlap)
*
* This function takes in 2 pointers to byte arrays.
* A source and a destination pointer are provided
* to the function as well as the length of the source bytes
* so the function knows how many to copy over.
* This function does not handle overlap, so data can become corrupted.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the array of bytes that we will copy from
* @param dst a pointer to the array of bytes that we will copy to
* @param length the number of bytes to copy
* @return uint8_t a pointer to the destination
*/
uint8_t * my_memcpy(uint8_t * src, uint8_t * dst, size_t length);

```

```

/**
* @brief function to set an array of bytes all to the same value
*
* This function takes in a pointer to an array of bytes
* then for a given number of bytes, writes a value to each

```

```

* of those bytes.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the array of bytes that we will write over
* @param length the number of bytes to iterate through
* @param value the value to write to every byte in the array
* @return uint8_t a pointer to the source (simply returns src)
*/

```

```

uint8_t * my_memset(uint8_t * src, size_t length, uint8_t value);

```

```

/**
* @brief function to write 0 to an array of bytes
*
* This function takes in a pointer to an array of bytes
* then for a given number of bytes, writes zero to each
* of those bytes.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the array of bytes that we will write to 0
* @param length the number of bytes to write
* @return uint8_t a pointer to the source (simply returns src)
*/

```

```

uint8_t * my_memzero(uint8_t * src, size_t length);

```

```

/**
* @brief function to reverse an array of bytes
*
* This function takes in a pointer to an array of bytes
* and the length of the array. It then reverses all of the
* bytes so the first byte becomes the last and the
* last byte becomes the first.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the array of bytes that we will reverse
* @param length the number of bytes to reverse
* @return uint8_t a pointer to the source (simply returns src)
*/

```

```

uint8_t * my_reverse(uint8_t * src, size_t length);

```

```

/**
* @brief function to allocate an array of size_t
*
* create an area in dynamic memory that is
* reserved for future used. Specifically, this
* function creates an array of size_t of length
* length.

```



```

*
* This function is implemented
* using only pointer arithmetic.
*
* @param length the length of the array to create
* @return void a pointer to the memory location in dynamic memory
*/
void * reserve_words(size_t length);

/**
* @brief function to free the dynamic memory created
*
* This function frees the dynamic memory created,
* usually created by reserve_words. This function
* makes sure that the memory can be used or
* reserved in the future. The function then
* returns if it was successful.
*
* This function is implemented
* using only pointer arithmetic.
*
* @param src a pointer to the start of the memory we need to free
* @return uint8_t an error code (0 or 1) iff the operation was successful
*/
uint8_t free_words(void * src);

/*
* @brief function to setup dma for different transfers
*
* this function takes in the DMA to use, address(s), transfer size,
* and asociated DMA information and configures the registers properly to utilize DMA
*
* @param
* @return DMA_e, this function returns an error type related to improper dma configuration
* */
DMA_e setup_memtransfer_dma(uint8_t* src, uint8_t src_len, uint8_t* dst,
                             size_t transfersize, size_t length);/*, uint8_t dma_index)*/

/*
* @brief DMA IRQ handler, for use with the DMA complete trigger
*
* @param none, this function takes no inputs
* @return void, this function has no return value
* */
void DMA0_IRQHandler();
#endif /* __MEMORY_H__ */

/*
* @file nordic.h
* @brief
*
* Abstraction layer for the nordic NRF chip
*

```

\* it's assumed that the NRF is connected to the KL25Z in the following manner

```
*   NRF   |   KL25Z
*-----|-----
*   GND   ->  GND
*   VCC   ->  3.3V
*   CSN   ->  PTD0
*   CE    ->  PTD5
*   SCK   ->  PTD1
*   MOSI  ->  PTD2
*   MISO  ->  PTD3
*   IRQ   ->  NC
*
* @author Seth Miers and Jake Cazden
* @date March 15, 2018
*/
```

```
#ifndef __NORDIC_H_
#define __NORDIC_H_
```

```
#define NRF_STATUS_REG (0x00)
#define NRF_TXADDR_REG (0x10)
#define NRF_POWER_UP (1)
#define NRF_POWER_DOWN (0)
#define NRF_POWER_UP_MASK (0x02)
```

```
/* RF SETUP REGISTER
```

```
* The RF setup register on the NRF device is used
* for setting up how the device outputs.
* The output power, the low noise amplifier, the data rate,
* and the PLL are all configured here.
* For more information see the datasheet by searching for:
* nRF24L01_Product_Specification_v2_0.pdf
*/
```

```
/* CONFIG REGISTER
```

```
* The configuration register on the NRF device is used
* for setting up the device, enabling and disabling it,
* enabling or disabling TX, enabling or disabling RX, then
* powering up or powering down the device. For more information
* see the NRF datasheet by searching for:
* nRF24L01_Product_Specification_v2_0.pdf
*/
```

```
/* RF CHANNEL REGISTER
```

```
* The RF Channel register is used for setting up the transmit
* channel. This channel is used so that multiple devices all
* transmitting over the same frequency can still communicate properly.
* For more information see the NRF datasheet at:
* nRF24L01_Product_Specification_v2_0.pdf
*/
```

```
/* STATUS REGISTER
```

- \* The Status register on the NRF device is to show the current
- \* status of the device. The status includes things such as if the data
- \* has been properly transmitted, how many times data should attempt
- \* to transmit, if there is anything in the RX buffer, if a fifo
- \* should be used for the RX buffer, and if the TX buffer is full.
- \* For more information see the NRF datasheet by searching for:
- \* nRF24L01\_Product\_Specification\_v2\_0.pdf
- \*/

#### /\* FIFO STATUS REGISTER

- \* The FIFO status register on the NRF device shows
- \* detailed information on the status of the RX and TX FIFO.
- \* This register will represent if the TX has had any reuse,
- \* and if the RX and TX FIFOs are full or empty. For more
- \* information see the NRF datasheet by searching for:
- \* nRF24L01\_Product\_Specification\_v2\_0.pdf
- \*/

#include <stdint.h>

/\*\*

- \* @brief reads a register and returns the value
- \*
- \* read a register off of the nrf chip and
- \* return the result
- \*
- \* @param uint8\_t readRegister the register to read
- \* @return uint8\_t the value of the register
- \*/

uint8\_t nrf\_read\_register(uint8\_t readRegister);

/\*\*

- \* @brief writes to a single register
- \*
- \* write to a register on the nrf chip
- \*
- \* @param uint8\_t writeRegister the register to write to
- \* @param uint8\_t value the value to write to the register
- \* @return uint8\_t the value of the register
- \*/

void nrf\_write\_register(uint8\_t writeRegister, uint8\_t value);

/\*\*

- \* @brief reads the status register and returns the result
- \*
- \* The Status register on the NRF device is to show the current
- \* status of the device. The status includes things such as if the data
- \* has been properly transmitted, how many times data should attempt
- \* to transmit, if there is anything in the RX buffer, if a fifo
- \* should be used for the RX buffer, and if the TX buffer is full.
- \* For more information see the NRF datasheet by searching for:
- \* nRF24L01\_Product\_Specification\_v2\_0.pdf

```

*
* @param none
* @return uint8_t the value of the register
*/
uint8_t nrf_read_status();

/**
* @brief write to the nrf config register
*
* The configuration register on the NRF device is used
* for setting up the device, enabling and disabling it,
* enabling or disabling TX, enabling or disabling RX, then
* powering up or powering down the device. For more information
* see the NRF datasheet by searching for:
* nRF24L01_Product_Specification_v2_0.pdf
*
* @param uint8_t config the config to write
* @return void
*/
void nrf_write_config(uint8_t config);

/**
* @brief read the rf_setup register
*
* The RF Channel register is used for setting up the transmit
* channel. This channel is used so that multiple devices all
* transmitting over the same frequency can still communicate properly.
* For more information see the NRF datasheet at:
* nRF24L01_Product_Specification_v2_0.pdf
*
* @param none
* @return uint8_t the value of the register
*/
uint8_t nrf_read_rf_ch();

/**
* @brief read the 5 bytes for the TX addr
*
* @param out uint8_t * address the tx address to be returned
* @return void
*/
void nrf_read_tx_addr(uint8_t * address);

/**
* @brief write the TX addr
*
* @param uint8_t * tx_addr the tx address to write
* @return void
*/
void nrf_write_tx_addr(uint8_t * tx_addr);

/**

```

```
* @brief read the fifo_status register
*
* The FIFO status register on the NRF device shows
* detailed information on the status of the RX and TX FIFO.
* This register will represent if the TX has had any reuse,
* and if the RX and TX FIFOs are full or empty. For more
* information see the NRF datasheet by searching for:
* nRF24L01_Product_Specification_v2_0.pdf
*
```

```
* @param none
* @return uint8_t the value of the register
*/
```

```
uint8_t nrf_read_fifo_status();
```

```
/**
* @brief send the flush tx command
*
* @param none
* @return void
*/
```

```
void nrf_flush_tx_fifo();
```

```
/**
* @brief send the flush rx command
*
* @param none
* @return void
*/
```

```
void nrf_flush_rx_fifo();
```

```
/**
* @brief enables the nrf chip
*
* @param none
* @return void
*/
```

```
extern void inline nrf_chip_enable();
```

```
/**
* @brief disables the nrf chip
*
* @param none
* @return void
*/
```

```
extern void inline nrf_chip_disable();
```

```
/**
* @brief enables the ability to transmit over RF
*
* @param none
* @return void
*/
```

```
void inline nrf_transmit_enable();
```

```
/**
```

```
 * @brief disables the ability to transmit over RF
 *
```

```
 * @param none
```

```
 * @return void
```

```
 */
```

```
void inline nrf_transmit_disable();
```

```
#endif /* __NORDIC_H_ */
```

```
/**
```

```
 * @file port.h
```

```
 * @brief this file contains gpio and led port headers and functions
```

```
 *
```

```
 * this file contains headers for GPIO setup and accessing, primarily to
```

```
 * utilize onboard KL25Z LEDs and enabling UART
```

```
 *
```

```
 * @author Seth Miers and Jake Cazden
```

```
 * @date March 04, 2018
```

```
 *
```

```
 */
```

```
#ifndef __PORT_H__
```

```
#define __PORT_H__
```

```
#include <stdint.h>
```

```
/* which gpio pin is the LED on? */
```

```
/* These values are for the KL25Z */
```

```
#define RGB_RED_PIN (18)
```

```
#define RGB_GREEN_PIN (19)
```

```
#define RGB_BLUE_PIN (1)
```

```
/* Macro functions */
```

```
#define RGB_RED_ON() (PORTB_Set( RGB_RED_PIN ))
```

```
#define RGB_RED_OFF() (PORTB_Clear( RGB_RED_PIN ))
```

```
#define RGB_RED_TOGGLE() (PORTB_Toggle( RGB_RED_PIN ))
```

```
#define RGB_GREEN_ON() (PORTB_Set( RGB_GREEN_PIN ))
```

```
#define RGB_GREEN_OFF() (PORTB_Clear( RGB_GREEN_PIN ))
```

```
#define RGB_GREEN_TOGGLE() (PORTB_Toggle( RGB_GREEN_PIN ))
```

```
#define RGB_BLUE_ON() (PORTB_Set( RGB_BLUE_PIN ))
```

```
#define RGB_BLUE_OFF() (PORTB_Clear( RGB_BLUE_PIN ))
```

```
#define RGB_BLUE_TOGGLE() (PORTB_Toggle( RGB_BLUE_PIN ))
```

```
/* the systick for the KL25z */
```

```
/*
```

```
 * @brief function to initialize the SysTick
```

```
 *
```

```
 * This function sets up the systick
```

```
 * to loop over the maximum value it can
```

```

* and to us the core clock
*
* @return void
*/
void inline InitSysTick();

/*
* @brief get the current time from systick
*
* simply returns the current time of the
* systick counter by returning the CVR register
*
* @return void
*/
uint32_t inline gettimeofday();

/**
* @brief configures the spi/nrf interface
*
* Configures the gpio pins to have the proper
* configuration for spi and nrf
*
* @param none
* @return void returns nothing
*/
void inline GPIO_nrf_init();

/**
* @brief configures the RGB LEDs
*
* configures the gpios to be outputs for the
* rgb leds. It then sets them to their default
* values
*
* @param none
* @return void returns nothing
*/
extern void inline GPIO_Configure();

/**
* @brief toggles the red LED state
*
* turns the red LED on if it's off
* turns the red LED off if it's on
*
* @param none
* @return void returns nothing
*/
void inline Toggle_Red_LED();

/**
* @brief sets an output to 1 (active high)

```

```

*
* @param the bit number to set on portB
* @return void returns nothing
*/
void PORTB_Set(uint8_t bit_num);

/**
* @brief sets an output to 1 (active high)
*
* @param the bit number to set on portD
* @return void returns nothing
*/
void PORTD_Set(uint8_t bit_num);

/**
* @brief sets an output to 0 (active low)
*
* @param the bit number to clear on portB
* @return void returns nothing
*/
void PORTB_Clear(uint8_t bit_num);

/**
* @brief sets an output to 0 (active low)
*
* @param the bit number to clear on portD
* @return void returns nothing
*/
void PORTD_Clear(uint8_t bit_num);

/**
* @brief toggles the output of the pin
*
* @param the bit number to toggle on portB
* @return void returns nothing
*/
void inline PORTB_Toggle(uint8_t bit_num);

/**
* @brief toggles the output of the pin
*
* @param the bit number to toggle on portD
* @return void returns nothing
*/
void inline PORTD_Toggle(uint8_t bit_num);

#endif /* __PORT_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 shouldl 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 projec2.h
 * @brief contains function headers for simple data processing of strings
 *
 * this file contains the function headers for performing data parsing on
 * strings recieved via UART on the kl25z system
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 *
 */

```

```
#ifndef __PROJECT2_H__
```

```
#define __PROJECT2_H__
```

```
#include<stdint.h>
```

```
#include"conversion.h"
```

```
typedef struct{
```

```
    uint32_t alphabetic;
```

```
    uint32_t numeric;
```

```
    uint32_t punctuation;
```

```
    uint32_t miscellaneous;
```

```
} charcounts;
```

```
volatile charcounts statistics;
```

```
#define CIRCBUF_HOST_LENGTH (256)
```

```
#ifndef __ASCII_OFFSETS__
```

```
#define __ASCII_OFFSETS__
```

```
#define ASCII_OFFSET_0 (48)
```

```
#define ASCII_OFFSET_9 (57)
```

```
#define ASCII_OFFSET_A (65)
```

```

#define ASCII_OFFSET_Z (90)
#define ASCII_OFFSET_A_ADDITION (55)*10+55= 'A' for hex interpreting*/
#define ASCII_OFFSET_F (70)
#define ASCII_OFFSET_LA (97)
#define ASCII_OFFSET_LZ (122)
#define ASCII_OFFSET_LA_ADDITION (87)*10+87= 'a' for hex interpreting*/
#define ASCII_OFFSET_LF (102)
#define ASCII_OFFSET_EOF (0x4)
#endif

```

```

/*
 * @brief interprets a datastream recieved over UART, finishes on EOF
 *
 *
 *
 *
 *
 * @param none
 * @return void returns nothing
 * */

```

```

void project2();
/*
 * @brief prints a datastream back to the user over UART or in terminal on HOST
 *
 *
 *
 *
 * @param none
 * @return void returns nothing
 * */

```

```

void dump_statistics();
#endif /* __PROJECT2_H__ */
/*
 * @file project3.h
 * @brief
 *
 *
 *
 * @author Seth Miers and Jake Cazden
 * @date March 15, 2018
 */

```

```

#ifndef __PROJECT3_H_
#define __PROJECT3_H_

```

```

#include "spi.h"

```

```

/*
 * @brief tests SPI functionality
 *
 * @param none

```

```

* @return void returns nothing
*
*/
void project3();

/*
* @brief initializes the stack to 0xAA
*
* @param none
* @return void returns nothing
*
*/
void stack_tracker_init();

/*
* @brief displays the max used space on the stack
*
* counts backwards from the end of the stack until it reaches
* a byte that has been written to, it then uses this value
* to profile how much of the stack has been used.
*
* @param none
* @return void returns nothing
*
*/
void stackusage();

/*
* @brief profiles the memmove and memset functions
*
* outputs results to uart
*
* @param none
* @return void returns nothing
*
*/
void profiler();

/*
* @brief tests a few outputs on the SPI
*
* outputs results to uart
*
* @param none
* @return void returns nothing
*
*/
void spi_setup_and_test();

#endif /* __PROJECT3_H_ */
/*

```

```
* @file spi.h
* @brief
*
*
*
* @author Seth Miers and Jake Cazden
* @date March 15, 2018
*/
```

```
#ifndef __SPI_H_
#define __SPI_H_
```

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

```
/**
 * @brief initializes the spi controller
 *
 * sets up clocks and gpio (using port.h) so that
 * the spi pins are configured properly
 *
 * @param none
 * @return void returns nothing
 */
```

```
void SPI_init();
```

```
/**
 * @brief reads a single byte from the SPI bus
 *
 * TODO ...
 *
 * @param uint8_t* the byte to read to
 * @return void returns nothing
 */
```

```
void SPI_read_byte(uint8_t * byte);
```

```
/**
 * @brief writes a single byte to the SPI bus
 *
 * TODO ...
 *
 * @param uint8_t the byte to write
 * @return void returns nothing
 */
```

```
void SPI_write_byte(uint8_t byte);
```

```
/**
 * @brief send numerous spi bytes
 *
 * repeated calls to SPI_write_byte
 * until all bytes are written
 *
```

```

* @param uint8_t * p a pointer to the byte array to write
* @param size_t length the number of bytes to send
* @return void returns nothing
*/
void SPI_send_packet(uint8_t * p, size_t length);

/**
* @brief flush all data from the bus
*
* block the interface until the transmit
* buffer is empty and has completed transmission
*
* @param none
* @return void returns nothing
*/
void SPI_flush();

#endif /* __SPI_H */

/**
* @file uart.h
* @brief function definitions for UART communication and interrupt system
*
* This file contains the headers necessary to utilize GPIO pins to
* realize a UART communication system, this also contains the functions
* to be called from an interrupt vector to process incoming data
*
* @author Seth Miers and Jake Cazden
* @date March 04, 2018
*
*/

#ifndef __UART_H__
#define __UART_H__

#include <stdint.h> /* for uint8_t */

#include <stddef.h> /* for size_t */
/*#ifndef KL25Z
#include "startup_MKL25Z4.S"
#endif
*/

/* what the circular buffer should initialize to */
#define BUFFER_LENGTH (256)

#define UART0_C4_OSR_SAMPLERATE (0x0f)
#define BAUD_CLOCK (CPU_INT_FAST_CLK_HZ)
#define BAUD_CALCULATION(x) (BAUD_CLOCK/((UART0_C4_OSR_SAMPLERATE+1)*(x)))
#define BAUD_RATE (19200)
#define CALCULATED_BAUD_MASK (BAUD_CALCULATION(BAUD_RATE))
#define CLEAR_PCR_ISF (1)
#define DISABLE_PCR_IRQC (0x00)
#define PCR_MUX_ALT2 (2)

```

```

#define SIM_SOPT2_UART0SRC_CLEAR      (3)
#define SIM_SOPT2_UART0SRC_PLLFLLSRC (1)
#define SIM_SOPT2_UART0SRC_MCGIRCLK   (3)
#define SIM_SOPT2_PLLFLLSEL_CLEAR     (1)
#define SIM_SOPT2_PLLFLLSEL_FLLSRC    (0)

#define MCG_C1_IRCLKEN_ENABLED (1)
#define MCG_C2_IRCS_FAST      (1)
#define MCG_C2_FCRDIV_CLEAR   (7)
#define MCG_C2_FCRDIV_NODIVISION (0)

#define SIM_SOPT5_UART0RXSRC_CLEAR (1)
#define SIM_SOPT5_UART0RXSRC_RXPIN (0)
#define SIM_SOPT5_UART0TXSRC_CLEAR (3)
#define SIM_SOPT5_UART0TXSRC_TXPIN (0)

#define UART0_BDH_LBKDIE_DISABLE (0)
#define UART0_BDH_RXEDGIE_DISABLE (0)
#define UART0_BDH_SBNS_SINGLESTOPBIT (0)

#define SBR_HIGHMASK      (0x1F00u)
#define SBR_LOWMASK      (0x00FFu)

#define UART0_C1_LOOPS_NORMALOPERATION (0)
#define UART0_C1_DOZEEN_ENABLED        (0)
#define UART0_C1_RSRC_DEFAULT          (0)
#define UART0_C1_M_8BIT                 (0)
#define UART0_C1_WAKE_DEFAULT           (0)
#define UART0_C1_ILT_AFTERSTOP          (1)
#define UART0_C1_PE_NOPARITY            (0)
#define UART0_C1_PT_DEFAULTPARTIY      (0)

#define UART0_C2_TIE_DISABLED           (0)
#define UART0_C2_TIE_ENABLED            (1)
#define UART0_C2_TCIE_DISABLED          (0)
#define UART0_C2_TCIE_ENABLED           (1)
#define UART0_C2_RIE_ENABLED            (1)
#define UART0_C2_TLIE_DISABLED          (0)
#define UART0_C2_TE_DISABLED            (0)
#define UART0_C2_TE_ENABLED             (1)
#define UART0_C2_RE_DISABLED            (0)
#define UART0_C2_RE_ENABLED             (1)
#define UART0_C2_RWU_NOWAKEUP           (0)
#define UART0_C2_SBK_NOBREAK            (0)

#define UART0_S1_TDRE_FULL              (0)
#define UART0_S1_TDRE_EMPTY            (1)
#define UART0_S1_RDRF_EMPTY            (0)
#define UART0_S1_RDRF_FULL             (1)

#define UART0_S1_TC_ACTIVE              (0)

```



```
#define UART0_S1_TC_IDLE          (1)
```

```
typedef enum {  
    UART_FAILURE,  
    UART_SUCCESS  
} UART_e;
```

```
/* @brief configures the specified uart with specified settings  
 *  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_configure();
```

```
/* @brief send a single character over the uart  
 *  
 * @param[in] uint8_t a pointer to a single character to send  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_send(uint8_t *data);
```

```
/* @brief send n characters over the uart  
 *  
 * @param[in] uint8_t a pointer to an array of characters to send  
 * @param[in] size_t the number of bytes to send (treated as int)  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_send_n(uint8_t *data, size_t num_bytes);
```

```
/* @brief receive a character from the uart  
 *  
 * @param[out] uint8_t a pointer to the location where the character should be stored  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_receive(uint8_t *data);
```

```
/* @brief receive n characters from the uart  
 *  
 * @param[out] uint8_t a pointer to the start of where characters should be stored  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_receive_n(uint8_t *data, size_t num_bytes);
```

```
/* @brief the interrupt request handler for the UART  
 * /*TODO does this need an extern or static keyword?*/  
void UART0_IRQHandler();
```

```
/* @brief enable transmit and interrupt for buffered interrupt based transmission  
 *  
 * @param none  
 * @return the status of the function defined by the enum UART_e  
 */  
UART_e UART_start_buffered_transmission();
```

```

#endif /* __UART_H__ */
/**
 * @file unittest.h
 * @brief function definitions for cmocka testing
 *
 * this file contains the necessary function definitions for
 * the unit testing using cmocka
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 */

#ifndef __UNITTEST_H__
#define __UNITTEST_H__

/**
 * @file unittest.c
 * @brief implementation of unittest.h
 *
 * this file implements unittest.h,
 * testing an array of different functions used
 * in this project
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 */

/* test different lengths, over 256 could cause errors
 * since we're using uint8_ts and int8_ts
 */
#define TEST_LENGTH (256)

/* @brief test the functions in memory.h
 *
 * @param[in] a double void pointer to the state of the tests
 * @return void
 */
void memory_test(void **state);

/* @brief test the functions in conversion.h
 *
 * memmove tests
 * - Invalid Pointers - Should return fail if pointers are NULL
 * - No Overlap - Should return a pass for a move
 * - SRC in DST region Overlap - Should succeed at this
 * - DST in SRC region Overlap - Should succeed at this
 * - DST == SRC - Should be successful and likely skip the add
 * memset
 * - Invalid Pointers - Should return fail if pointers are NULL
 * - Check Set - Should accurately set region for length Value

```

```

* memzero
* - Invalid Pointers - Should return fail if pointers are NULL
* - Check Set - Should accurately set region to zeroes
* reverse
* - Invalid Pointers - Should return fail if pointers are NULL
* - Check Odd reverse - Should check that reverse succeeded for even length
* - Check Even reverse - Should check that reverse succeeded for even length
* - Check characters - Should be able to reverse any character set (256 byte array of 0-255)
*
* @param[in] a double void pointer to the state of the tests
* @return void
*/

```

```

void conversion_test(void **state);

```

```

/* @brief test the functions in data.h

```

```

*
* atoi
* - Invalid Pointers
* - Zero Integer
* - Max sized Integer
* itoa
* - Invalid Pointers
* - Zero Integer
* - Max sized Integer
*
* @param[in] a double void pointer to the state of the tests
* @return void
*/

```

```

void data_test(void **state);

```

```

/* @brief test the functions in circbuf.h

```

```

*
* Endianness conversion
* - Invalid Pointer
* - Valid Conversion - Test that a big-to-little and little-to-big conversion worked
* - Valid Conversion - Test that a big-to-little and little-to-bit produce the same as the
*   original
*
* @param[in] a double void pointer to the state of the tests
* @return void
*/

```

```

void circbuf_test(void **state);

```

```

/* @brief the caller function for the others

```

```

*
* implements cmocka to call a unittest of
* several other tests
*
* @param[in] a double void pointer to the state of the tests
* @return void
*/

```

```

int unittest();

```

```

#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:            b150220
**
** 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 */
#ifndef (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 {
/* 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 */
    __IO 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;
__IO uint32_t CLPD;
*/
__IO uint32_t CLPS;
__IO uint32_t CLP4;
__IO uint32_t CLP3;
__IO uint32_t CLP2;
__IO uint32_t CLP1;
__IO uint32_t CLP0;
uint8_t RESERVED_0[4];
__IO uint32_t CLMD;
0x54 */
__IO uint32_t CLMS;
0x58 */
__IO uint32_t CLM4;
0x5C */
__IO uint32_t CLM3;
0x60 */
__IO uint32_t CLM2;
0x64 */
__IO uint32_t CLM1;
0x68 */
__IO uint32_t CLM0;
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_ACFG_T_MASK              0x10u
#define ADC_SC2_ACFG_T_SHIFT            4
#define ADC_SC2_ACFG_T_WIDTH            1
#define ADC_SC2_ACFG_T(x)
(((uint32_t)(((uint32_t)(x))<<ADC_SC2_ACFG_T_SHIFT))&ADC_SC2_ACFG_T_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           0x3FFu
#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 }

/* -----
-- 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 DACCR;    /**< 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_BASE_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 }

/* -----
-- CMP - Register accessor macros
----- */

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

/* CMP - Register instance definitions */
/* CMP0 */
#define CMP0_CR0                  CMP_CR0_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;
        __IO uint8_t DATH;
    } DAT[2];
    uint8_t RESERVED_0[28];
    __IO uint8_t SR;
    __IO uint8_t C0;
    __IO uint8_t C1;
    __IO uint8_t C2;
} 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_BASE }
/** Array initializer of DAC peripheral base pointers */
#define DAC_BASE_PTRS                { DAC0 }

/* -----
-- 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 { /* offset: 0x100, array step: 0x10 */
        __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_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 }

```

```

/* -----
-- 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 */

/* -----
-- FGPIO Peripheral Access Layer
----- */

/*!
 * @addtogroup FGPIO_Peripheral_Access_Layer FGPIO Peripheral Access Layer
 * @{
 */

/** FGPIO - 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 */

```

```
} FGPIO_Type, *FGPIO_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 FGPIOTPTTOMASK 0xFFFFFFFFu
#define FGPIOTPTTOSHIFT 0
#define FGPIOTPTTOWIDTH 32
#define FGPIOTPTTO(x)
(((uint32_t)((uint32_t)(x))<<FGPIOTPTTOSHIFT)&FGPIOTPTTOMASK)
/* PDIR Bit Fields */
#define FGPIOPDIRPDIRMASK 0xFFFFFFFFu
#define FGPIOPDIRPDIRSHIFT 0
#define FGPIOPDIRPDIRWIDTH 32
#define FGPIOPDIRPDIR(x)
(((uint32_t)((uint32_t)(x))<<FGPIOPDIRPDIRSHIFT)&FGPIOPDIRPDIRMASK)
/* PDDR Bit Fields */
#define FGPIOPDDR_PDD_MASK 0xFFFFFFFFu
#define FGPIOPDDR_PDD_SHIFT 0
#define FGPIOPDDR_PDD_WIDTH 32
#define FGPIOPDDR_PDD(x)
(((uint32_t)((uint32_t)(x))<<FGPIOPDDR_PDD_SHIFT)&FGPIOPDDR_PDD_MASK)

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

/* FGPIOT - Peripheral instance base addresses */
/** Peripheral FGPIOTA base address */
#define FGPIOTA_BASE (0xF80FF000u)
/** Peripheral FGPIOTA base pointer */
#define FGPIOTA ((FGPIOT_Type *)FGPIOTA_BASE)
#define FGPIOTA_BASE_PTR (FGPIOTA)
/** Peripheral FGPIOTB base address */
#define FGPIOTB_BASE (0xF80FF040u)
/** Peripheral FGPIOTB base pointer */
#define FGPIOTB ((FGPIOT_Type *)FGPIOTB_BASE)
#define FGPIOTB_BASE_PTR (FGPIOTB)
/** Peripheral FGPIOTC base address */
#define FGPIOTC_BASE (0xF80FF080u)
/** Peripheral FGPIOTC base pointer */
#define FGPIOTC ((FGPIOT_Type *)FGPIOTC_BASE)
#define FGPIOTC_BASE_PTR (FGPIOTC)
/** Peripheral FGPIOTD base address */
#define FGPIOTD_BASE (0xF80FF0C0u)
/** Peripheral FGPIOTD base pointer */
#define FGPIOTD ((FGPIOT_Type *)FGPIOTD_BASE)
#define FGPIOTD_BASE_PTR (FGPIOTD)
/** Peripheral FGPIOTE base address */
#define FGPIOTE_BASE (0xF80FF100u)
/** Peripheral FGPIOTE base pointer */
#define FGPIOTE ((FGPIOT_Type *)FGPIOTE_BASE)
#define FGPIOTE_BASE_PTR (FGPIOTE)
/** Array initializer of FGPIOT peripheral base addresses */
#define FGPIOT_BASE_ADDRS { FGPIOTA_BASE, FGPIOTB_BASE, FGPIOTC_BASE,

```

```

FGPIOD_BASE, FGPIOE_BASE }
/** Array initializer of FGPIO peripheral base pointers */
#define FGPIO_BASE_PTRS { FGPIOA, FGPIOB, FGPIOC, FGPIOD, FGPIOE }

/* -----
-- FGPIO - Register accessor macros
----- */

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

/* FGPIO - Register instance definitions */
/* FGPIOA */
#define FGPIOA_PDOR FGPIO_PDOR_REG(FGPIOA)
#define FGPIOA_PSOR FGPIO_PSOR_REG(FGPIOA)
#define FGPIOA_PCOR FGPIO_PCOR_REG(FGPIOA)
#define FGPIOA_PTOR FGPIO_PTOR_REG(FGPIOA)
#define FGPIOA_PDIR FGPIO_PDIR_REG(FGPIOA)
#define FGPIOA_PDDR FGPIO_PDDR_REG(FGPIOA)
/* FGPIOB */
#define FGPIOB_PDOR FGPIO_PDOR_REG(FGPIOB)
#define FGPIOB_PSOR FGPIO_PSOR_REG(FGPIOB)
#define FGPIOB_PCOR FGPIO_PCOR_REG(FGPIOB)
#define FGPIOB_PTOR FGPIO_PTOR_REG(FGPIOB)
#define FGPIOB_PDIR FGPIO_PDIR_REG(FGPIOB)
#define FGPIOB_PDDR FGPIO_PDDR_REG(FGPIOB)
/* 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)
/* FGPIOE */
#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 FGPIO_Register_Accessor_Macros */

/*!
* @}
*/ /* end of group FGPIO_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 */
    __IO uint8_t FSEC;           /**< Flash Security Register, offset: 0x2 */
    __IO 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_FCNFG_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_FCCOBB_REG(base)         ((base)->FCCOBB)
#define FTFA_FCCOBA_REG(base)         ((base)->FCCOBA)
#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_FCCOBB_CCOBn_MASK          0xFFu
#define FTFA_FCCOBB_CCOBn_SHIFT        0
#define FTFA_FCCOBB_CCOBn_WIDTH        8
#define FTFA_FCCOBB_CCOBn(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FCCOBB_CCOBn_SHIFT))&FTFA_FCCOBB_CCOBn_MASK)
/* FCCOBA Bit Fields */
#define FTFA_FCCOBA_CCOBn_MASK          0xFFu
#define FTFA_FCCOBA_CCOBn_SHIFT        0
#define FTFA_FCCOBA_CCOBn_WIDTH        8
#define FTFA_FCCOBA_CCOBn(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FCCOBA_CCOBn_SHIFT))&FTFA_FCCOBA_CCOBn_MASK)
/* FCCOB9 Bit Fields */
#define FTFA_FCCOB9_CCOBn_MASK          0xFFu
#define FTFA_FCCOB9_CCOBn_SHIFT        0
#define FTFA_FCCOB9_CCOBn_WIDTH        8
#define FTFA_FCCOB9_CCOBn(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FCCOB9_CCOBn_SHIFT))&FTFA_FCCOB9_CCOBn_MASK)
/* FCCOB8 Bit Fields */
#define FTFA_FCCOB8_CCOBn_MASK          0xFFu
#define FTFA_FCCOB8_CCOBn_SHIFT        0
#define FTFA_FCCOB8_CCOBn_WIDTH        8
#define FTFA_FCCOB8_CCOBn(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FCCOB8_CCOBn_SHIFT))&FTFA_FCCOB8_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_FPROT2_PROT_MASK           0xFFu
#define FTFA_FPROT2_PROT_SHIFT         0
#define FTFA_FPROT2_PROT_WIDTH         8
#define FTFA_FPROT2_PROT(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FPROT2_PROT_SHIFT))&FTFA_FPROT2_PROT_MASK)
/* FPROT1 Bit Fields */
#define FTFA_FPROT1_PROT_MASK           0xFFu
#define FTFA_FPROT1_PROT_SHIFT         0
#define FTFA_FPROT1_PROT_WIDTH         8
#define FTFA_FPROT1_PROT(x)
(((uint8_t)(((uint8_t)(x))<<FTFA_FPROT1_PROT_SHIFT))&FTFA_FPROT1_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_BASE_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 }

/* -----
-- 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_FCNFG                FTFA_FCNFG_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_FCCOBB               FTFA_FCCOBB_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          0xFFFFFFFFFu
#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_PTTO_MASK         0xFFFFFFFFFu
#define GPIO_PSOR_PTTO_SHIFT        0
#define GPIO_PSOR_PTTO_WIDTH        32
#define GPIO_PSOR_PTTO(x)
(((uint32_t)((uint32_t)(x))<<GPIO_PSOR_PTTO_SHIFT))&GPIO_PSOR_PTTO_MASK)
/* PCOR Bit Fields */
#define GPIO_PCOR_PTCO_MASK         0xFFFFFFFFFu
#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         0xFFFFFFFFFu
#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          0xFFFFFFFFFu
#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          0xFFFFFFFFFu
#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          0xFEu
#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_SSLT_MASK              0xFFu
#define I2C_SLTH_SSLT_SHIFT            0
#define I2C_SLTH_SSLT_WIDTH            8
#define I2C_SLTH_SSLT(x)
(((uint8_t)(((uint8_t)(x))<<I2C_SLTH_SSLT_SHIFT))&I2C_SLTH_SSLT_MASK)
/* SLTL Bit Fields */
#define I2C_SLTL_SSLT_MASK              0xFFu

```

```

#define I2C_SLTL_SSLT_SHIFT          0
#define I2C_SLTL_SSLT_WIDTH          8
#define I2C_SLTL_SSLT(x)
(((uint8_t)(((uint8_t)(x))<<I2C_SLTL_SSLT_SHIFT))&I2C_SLTL_SSLT_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 }

/* -----
-- 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 */
    __IO 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)(((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)(((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)(((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)(((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)(((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)(((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)(((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)(((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)(((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)((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            0xFu
#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_MASK)
#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_MASK)
/* FILT2 Bit Fields */
#define LLWU_FILT2_FILTSEL_MASK            0xFu
#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_MASK)
#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_MASK)

/*!
 * @}

```

```
*/ /* 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 }
```

```
/* -----
```

```
-- 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_FILT1 LLWU_FILT1_REG(LLWU)
```

```
#define LLWU_FILT2 LLWU_FILT2_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)

```

```

(((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_CMR_COMPARE_WIDTH            16
#define LPTMR_CMR_COMPARE(x)
(((uint32_t)(((uint32_t)(x))<<LPTMR_CMR_COMPARE_SHIFT))&LPTMR_CMR_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 }

/* -----
-- 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_CMR LPTMR_CMR_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 */
    __IO uint8_t C7;           /**< MCG Control 7 Register, offset: 0xC */
    __IO uint8_t C8;           /**< MCG Control 8 Register, offset: 0xD */
    __IO uint8_t C9;           /**< MCG Control 9 Register, offset: 0xE */
    __IO 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_DM32_MASK                0x80u
#define MCG_C4_DM32_SHIFT                7
#define MCG_C4_DM32_WIDTH                1
#define MCG_C4_DM32(x)
(((uint8_t)(((uint8_t)(x))<<MCG_C4_DM32_SHIFT))&MCG_C4_DM32_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_PLLSTEN0_MASK            0x20u
#define MCG_C5_PLLSTEN0_SHIFT            5
#define MCG_C5_PLLSTEN0_WIDTH            1

```



```

#define MCG_C5_PLLSTEN0(x)
(((uint8_t)(((uint8_t)(x))<<MCG_C5_PLLSTEN0_SHIFT))&MCG_C5_PLLSTEN0_MASK)
#define MCG_C5_PLLCLKEN0_MASK          0x40u
#define MCG_C5_PLLCLKEN0_SHIFT        6
#define MCG_C5_PLLCLKEN0_WIDTH        1
#define MCG_C5_PLLCLKEN0(x)
(((uint8_t)(((uint8_t)(x))<<MCG_C5_PLLCLKEN0_SHIFT))&MCG_C5_PLLCLKEN0_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 }

/* -----
-- 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 DEVICEARCH;     /**< 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_DEVICEARCH_REG(base)   ((base)->DEVICEARCH)
#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_RAMPRIV_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_WATERMARK_MASK)
/* BASE Bit Fields */
#define MTB_BASE_BASEADDR_MASK               0xFFFFFFFFFu
#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           0xFFFFFFFFFu
#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               0xFFFFFFFFFu
#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           0xFFFFFFFFFu
#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       0xFFFFFFFFFu
#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_LOCK
ACCESS_MASK)
/* LOCKSTAT Bit Fields */
#define MTB_LOCKSTAT_LOCKSTAT_MASK           0xFFFFFFFFFu

```

```

#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_M
ASK)
/* 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)
/* DEVICEARCH Bit Fields */
#define MTB_DEVICEARCH_DEVICEARCH_MASK    0xFFFFFFFFFu
#define MTB_DEVICEARCH_DEVICEARCH_SHIFT    0
#define MTB_DEVICEARCH_DEVICEARCH_WIDTH    32
#define MTB_DEVICEARCH_DEVICEARCH(x)
(((uint32_t)(((uint32_t)(x))<<MTB_DEVICEARCH_DEVICEARCH_SHIFT))&MTB_DEVICEARCH_DEVIC
EARCH_MASK)
/* DEVICECFG Bit Fields */
#define MTB_DEVICECFG_DEVICECFG_MASK       0xFFFFFFFFFu
#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   0xFFFFFFFFFu
#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_DEVI
CETYPID_MASK)
/* PERIPID Bit Fields */
#define MTB_PERIPID_PERIPID_MASK           0xFFFFFFFFFu
#define MTB_PERIPID_PERIPID_SHIFT          0
#define MTB_PERIPID_PERIPID_WIDTH          32
#define MTB_PERIPID_PERIPID(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_DEVICEARCH                 MTB_DEVICEARCH_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             0xFFFFFFFFFu
#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_MASK_WIDTH            5
#define MTBDWT_MASK_MASK(x)
(((uint32_t)(((uint32_t)(x))<<MTBDWT_MASK_MASK_SHIFT))&MTBDWT_MASK_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_ACOMP0_MASK          0x1u
#define MTBDWT_TBCTRL_ACOMP0_SHIFT         0
#define MTBDWT_TBCTRL_ACOMP0_WIDTH         1
#define MTBDWT_TBCTRL_ACOMP0(x)
(((uint32_t)(((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP0_SHIFT))&MTBDWT_TBCTRL_ACOMP0_MASK)
#define MTBDWT_TBCTRL_ACOMP1_MASK          0x2u

```

```

#define MTBDWT_TBCTRL_ACOMP1_SHIFT      1
#define MTBDWT_TBCTRL_ACOMP1_WIDTH      1
#define MTBDWT_TBCTRL_ACOMP1(x)
(((uint32_t)(((uint32_t)(x))<<MTBDWT_TBCTRL_ACOMP1_SHIFT))&MTBDWT_TBCTRL_ACOMP1_MA
SK)
#define MTBDWT_TBCTRL_NUMCOMP_MASK      0xF0000000u
#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  0xFFFFFFFFFu
#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_D
EVICECFG_MASK)
/* DEVICETYPID Bit Fields */
#define MTBDWT_DEVICETYPID_DEVICETYPID_MASK  0xFFFFFFFFFu
#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_DEVICETY
PID_DEVICETYPID_MASK)
/* PERIPID Bit Fields */
#define MTBDWT_PERIPID_PERIPID_MASK  0xFFFFFFFFFu
#define MTBDWT_PERIPID_PERIPID_SHIFT  0
#define MTBDWT_PERIPID_PERIPID_WIDTH  32
#define MTBDWT_PERIPID_PERIPID(x)
(((uint32_t)(((uint32_t)(x))<<MTBDWT_PERIPID_PERIPID_SHIFT))&MTBDWT_PERIPID_PERIPID
_MASK)
/* COMPID Bit Fields */
#define MTBDWT_COMPID_COMPID_MASK  0xFFFFFFFFFu
#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_MA
SK)

/*!
 * @}
 */ /* 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_PERIPHID0           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_REG(FTFA_FlashConfig) NV_BACKKEY3_REG(FTFA_FlashConfig)
#define NV_BACKKEY2_REG(FTFA_FlashConfig) NV_BACKKEY2_REG(FTFA_FlashConfig)
#define NV_BACKKEY1_REG(FTFA_FlashConfig) NV_BACKKEY1_REG(FTFA_FlashConfig)
#define NV_BACKKEY0_REG(FTFA_FlashConfig) NV_BACKKEY0_REG(FTFA_FlashConfig)
#define NV_BACKKEY7_REG(FTFA_FlashConfig) NV_BACKKEY7_REG(FTFA_FlashConfig)
#define NV_BACKKEY6_REG(FTFA_FlashConfig) NV_BACKKEY6_REG(FTFA_FlashConfig)
#define NV_BACKKEY5_REG(FTFA_FlashConfig) NV_BACKKEY5_REG(FTFA_FlashConfig)
#define NV_BACKKEY4_REG(FTFA_FlashConfig) NV_BACKKEY4_REG(FTFA_FlashConfig)
#define NV_FPROT3_REG(FTFA_FlashConfig) NV_FPROT3_REG(FTFA_FlashConfig)
#define NV_FPROT2_REG(FTFA_FlashConfig) NV_FPROT2_REG(FTFA_FlashConfig)
#define NV_FPROT1_REG(FTFA_FlashConfig) NV_FPROT1_REG(FTFA_FlashConfig)
#define NV_FPROT0_REG(FTFA_FlashConfig) NV_FPROT0_REG(FTFA_FlashConfig)
#define NV_FSEC_REG(FTFA_FlashConfig) NV_FSEC_REG(FTFA_FlashConfig)
#define NV_FOPT_REG(FTFA_FlashConfig) 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_EREFTEN_MASK 0x20u
#define OSC_CR_EREFTEN_SHIFT 5
#define OSC_CR_EREFTEN_WIDTH 1
#define OSC_CR_EREFTEN(x)
(((uint8_t)((uint8_t)(x))<<OSC_CR_EREFTEN_SHIFT))&OSC_CR_EREFTEN_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
 * @{

```



```

*/

/** PIT - Register Layout Typedef */
typedef struct {
    __IO uint32_t MCR;                /**< PIT Module Control Register, offset: 0x0 */
    uint8_t RESERVED_0[220];
    __I uint32_t LTMR64H;             /**< PIT Upper Lifetime Timer Register, offset: 0xE0 */
    __I uint32_t LTMR64L;             /**< PIT Lower Lifetime Timer Register, offset: 0xE4 */
    uint8_t RESERVED_1[24];
    struct {                          /* offset: 0x100, array step: 0x10 */
        __IO uint32_t LDVAL;          /**< Timer Load Value Register, array offset: 0x100, array step:
0x10 */
        __I uint32_t CVAL;            /**< Current Timer Value Register, array offset: 0x104, array step:
0x10 */
        __IO uint32_t TCTRL;          /**< Timer Control Register, array offset: 0x108, array step: 0x10 */
        __IO uint32_t TFLG;          /**< Timer Flag Register, array offset: 0x10C, array step: 0x10 */
    } CHANNEL[2];
} PIT_Type, *PIT_MemMapPtr;

/* -----
-- PIT - Register accessor macros
----- */

/*!
* @addtogroup PIT_Register_Accessor_Macros PIT - Register accessor macros
* @{
*/

/* PIT - Register accessors */
#define PIT_MCR_REG(base)             ((base)->MCR)
#define PIT_LTMR64H_REG(base)         ((base)->LTMR64H)
#define PIT_LTMR64L_REG(base)         ((base)->LTMR64L)
#define PIT_LDVAL_REG(base,index)     ((base)->CHANNEL[index].LDVAL)
#define PIT_LDVAL_COUNT                2
#define PIT_CVAL_REG(base,index)      ((base)->CHANNEL[index].CVAL)
#define PIT_CVAL_COUNT                 2
#define PIT_TCTRL_REG(base,index)     ((base)->CHANNEL[index].TCTRL)
#define PIT_TCTRL_COUNT                2
#define PIT_TFLG_REG(base,index)      ((base)->CHANNEL[index].TFLG)
#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 0xFFFFFFFFFu

#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 0xFFFFFFFFFu

#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 0xFFFFFFFFFu

#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 0xFFFFFFFFFu

#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 }

/* -----
-- 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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_BASE_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 }

```

```

/* -----
-- 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          0xFFFFFu
#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)
#define PORT_GPCHR_GPWE_MASK          0xFFFF0000u
#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            0xFFFFFFFFFu
#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 }

/* -----
-- 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 PORT_PCR_REG(PORTA,24)
#define PORTA_PCR25 PORT_PCR_REG(PORTA,25)

```

```

#define PORTA_PCR26      PORT_PCR_REG(PORTA,26)
#define PORTA_PCR27      PORT_PCR_REG(PORTA,27)
#define PORTA_PCR28      PORT_PCR_REG(PORTA,28)
#define PORTA_PCR29      PORT_PCR_REG(PORTA,29)
#define PORTA_PCR30      PORT_PCR_REG(PORTA,30)
#define PORTA_PCR31      PORT_PCR_REG(PORTA,31)
#define PORTA_GPCLR      PORT_GPCLR_REG(PORTA)
#define PORTA_GPCHR      PORT_GPCHR_REG(PORTA)
#define PORTA_ISFR       PORT_ISFR_REG(PORTA)
/* PORTB */
#define PORTB_PCR0        PORT_PCR_REG(PORTB,0)
#define PORTB_PCR1        PORT_PCR_REG(PORTB,1)
#define PORTB_PCR2        PORT_PCR_REG(PORTB,2)
#define PORTB_PCR3        PORT_PCR_REG(PORTB,3)
#define PORTB_PCR4        PORT_PCR_REG(PORTB,4)
#define PORTB_PCR5        PORT_PCR_REG(PORTB,5)
#define PORTB_PCR6        PORT_PCR_REG(PORTB,6)
#define PORTB_PCR7        PORT_PCR_REG(PORTB,7)
#define PORTB_PCR8        PORT_PCR_REG(PORTB,8)
#define PORTB_PCR9        PORT_PCR_REG(PORTB,9)
#define PORTB_PCR10       PORT_PCR_REG(PORTB,10)
#define PORTB_PCR11       PORT_PCR_REG(PORTB,11)
#define PORTB_PCR12       PORT_PCR_REG(PORTB,12)
#define PORTB_PCR13       PORT_PCR_REG(PORTB,13)
#define PORTB_PCR14       PORT_PCR_REG(PORTB,14)
#define PORTB_PCR15       PORT_PCR_REG(PORTB,15)
#define PORTB_PCR16       PORT_PCR_REG(PORTB,16)
#define PORTB_PCR17       PORT_PCR_REG(PORTB,17)
#define PORTB_PCR18       PORT_PCR_REG(PORTB,18)
#define PORTB_PCR19       PORT_PCR_REG(PORTB,19)
#define PORTB_PCR20       PORT_PCR_REG(PORTB,20)
#define PORTB_PCR21       PORT_PCR_REG(PORTB,21)
#define PORTB_PCR22       PORT_PCR_REG(PORTB,22)
#define PORTB_PCR23       PORT_PCR_REG(PORTB,23)
#define PORTB_PCR24       PORT_PCR_REG(PORTB,24)
#define PORTB_PCR25       PORT_PCR_REG(PORTB,25)
#define PORTB_PCR26       PORT_PCR_REG(PORTB,26)
#define PORTB_PCR27       PORT_PCR_REG(PORTB,27)
#define PORTB_PCR28       PORT_PCR_REG(PORTB,28)
#define PORTB_PCR29       PORT_PCR_REG(PORTB,29)
#define PORTB_PCR30       PORT_PCR_REG(PORTB,30)
#define PORTB_PCR31       PORT_PCR_REG(PORTB,31)
#define PORTB_GPCLR      PORT_GPCLR_REG(PORTB)
#define PORTB_GPCHR      PORT_GPCHR_REG(PORTB)
#define PORTB_ISFR       PORT_ISFR_REG(PORTB)
/* PORTC */
#define PORTC_PCR0        PORT_PCR_REG(PORTC,0)
#define PORTC_PCR1        PORT_PCR_REG(PORTC,1)
#define PORTC_PCR2        PORT_PCR_REG(PORTC,2)
#define PORTC_PCR3        PORT_PCR_REG(PORTC,3)
#define PORTC_PCR4        PORT_PCR_REG(PORTC,4)
#define PORTC_PCR5        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 {
    __IO uint8_t SRS0;          /**< System Reset Status Register 0, offset: 0x0 */
    __IO 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 PERIPID4;          /**< Peripheral ID Register, offset: 0xFD0 */
    __I uint32_t PERIPID5;          /**< Peripheral ID Register, offset: 0xFD4 */
    __I uint32_t PERIPID6;          /**< Peripheral ID Register, offset: 0xFD8 */
    __I uint32_t PERIPID7;          /**< Peripheral ID Register, offset: 0xFDC */
    __I uint32_t PERIPID0;          /**< Peripheral ID Register, offset: 0xFE0 */
    __I uint32_t PERIPID1;          /**< Peripheral ID Register, offset: 0xFE4 */
    __I uint32_t PERIPID2;          /**< Peripheral ID Register, offset: 0xFE8 */
    __I uint32_t PERIPID3;          /**< 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      0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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        0xFFFFFFFFFu
#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          0xFFFFFFFFFu
#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              0xFFFFFFFFFu
#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_PERIPHID0      ROM_PERIPHID0_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

```

```
* @{\n*/
```

```
/* RTC - Register accessors */
```

```
#define RTC_TSR_REG(base)      ((base)->TSR)\n#define RTC_TPR_REG(base)      ((base)->TPR)\n#define RTC_TAR_REG(base)      ((base)->TAR)\n#define RTC_TCR_REG(base)      ((base)->TCR)\n#define RTC_CR_REG(base)       ((base)->CR)\n#define RTC_SR_REG(base)       ((base)->SR)\n#define RTC_LR_REG(base)       ((base)->LR)\n#define RTC_IER_REG(base)      ((base)->IER)
```

```
/*!
```

```
* @}
```

```
*/ /* end of group RTC_Register_Accessor_Macros */
```

```
/* -----
```

```
-- RTC Register Masks
```

```
----- */
```

```
/*!
```

```
* @addtogroup RTC_Register_Masks RTC Register Masks
```

```
* @{\n*/
```

```
/* TSR Bit Fields */
```

```
#define RTC_TSR_TSR_MASK      0xFFFFFFFFu\n#define RTC_TSR_TSR_SHIFT      0\n#define RTC_TSR_TSR_WIDTH      32\n#define RTC_TSR_TSR(x)\n(((uint32_t)((uint32_t)(x))<<RTC_TSR_TSR_SHIFT))&RTC_TSR_TSR_MASK)
```

```
/* TPR Bit Fields */
```

```
#define RTC_TPR_TPR_MASK      0xFFFFu\n#define RTC_TPR_TPR_SHIFT      0\n#define RTC_TPR_TPR_WIDTH      16\n#define RTC_TPR_TPR(x)\n(((uint32_t)((uint32_t)(x))<<RTC_TPR_TPR_SHIFT))&RTC_TPR_TPR_MASK)
```

```
/* TAR Bit Fields */
```

```
#define RTC_TAR_TAR_MASK      0xFFFFFFFFu\n#define RTC_TAR_TAR_SHIFT      0\n#define RTC_TAR_TAR_WIDTH      32\n#define RTC_TAR_TAR(x)\n(((uint32_t)((uint32_t)(x))<<RTC_TAR_TAR_SHIFT))&RTC_TAR_TAR_MASK)
```

```
/* TCR Bit Fields */
```

```
#define RTC_TCR_TCR_MASK      0xFFu\n#define RTC_TCR_TCR_SHIFT      0\n#define RTC_TCR_TCR_WIDTH      8\n#define RTC_TCR_TCR(x)\n(((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                0xFF0000u
#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                0xFF000000u
#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_TIE_MASK          0x1u
#define RTC_IER_TIE_SHIFT         0
#define RTC_IER_TIE_WIDTH         1
#define RTC_IER_TIE(x)
(((uint32_t)(((uint32_t)(x))<<RTC_IER_TIE_SHIFT))&RTC_IER_TIE_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_BASE_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 }

/* -----
-- 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];
    __IO 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_M
ASK)
#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_TPM1CH0SRC_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_TPM2CH0SRC_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_TPM0CLKSEL_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_TPM1CLKSEL_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_TPM2CLKSEL_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_UART0TXSRC_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_UART0RXSRC_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_UART1TXSRC_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_UART1RXSRC_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         0xFu
#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_ADC0ALTTRGEN_MASK       0x80u
#define SIM_SOPT7_ADC0ALTTRGEN_SHIFT     7
#define SIM_SOPT7_ADC0ALTTRGEN_WIDTH     1
#define SIM_SOPT7_ADC0ALTTRGEN(x)
(((uint32_t)((uint32_t)(x))<<SIM_SOPT7_ADC0ALTTRGEN_SHIFT))&SIM_SOPT7_ADC0ALTTRGEN_M
ASK)

```

/\* SDID Bit Fields \*/

```
#define SIM_SDID_PINID_MASK          0xFu
#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_MASK)
```

```

#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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             0x8000000u
#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                0xFFFFFFFFFu
#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                  0xFFFFFFFFFu
#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 */
    __IO 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)

```

```

(((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)
/** 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)((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)((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)((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)(((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)(((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)(((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)(((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)(((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)(((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_BASE_PTR            (SPI0)
/** Peripheral SPI1 base address */
#define SPI1_BASE                (0x40077000u)
/** Peripheral SPI1 base pointer */
#define SPI1                      ((SPI_Type *)SPI1_BASE)
#define SPI1_BASE_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 }

/* -----
-- 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 {                        /* offset: 0xC, array step: 0x8 */
        __IO uint32_t CnSC;          /**< Channel (n) Status and Control, array offset: 0xC, 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_CMODO_MASK              0x18u
#define TPM_SC_CMODO_SHIFT              3
#define TPM_SC_CMODO_WIDTH              2
#define TPM_SC_CMODO(x)
(((uint32_t)((uint32_t)(x))<<TPM_SC_CMODO_SHIFT))&TPM_SC_CMODO_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_CH0F_MASK        0x1u
#define TPM_STATUS_CH0F_SHIFT       0
#define TPM_STATUS_CH0F_WIDTH       1
#define TPM_STATUS_CH0F(x)
(((uint32_t)(((uint32_t)(x))<<TPM_STATUS_CH0F_SHIFT))&TPM_STATUS_CH0F_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_MASK)

```

```

/*!
 * @}
 */ /* 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)
/** Peripheral TPM1 base pointer */
#define TPM1                        ((TPM_Type *)TPM1_BASE)
#define TPM1_BASE_PTR                (TPM1)
/** Peripheral TPM2 base address */
#define TPM2_BASE                    (0x4003A000u)
/** Peripheral TPM2 base pointer */
#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 }

```

```

/* -----
-- 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_C0SC          TPM_CnSC_REG(TPM0,0)
#define TPM0_C0V          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              0xFFFFFu
#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              0xFFFFFu
#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)
#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)

```

```

/*!

```

```

* @}
*// 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_BASE_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 }

/* -----
-- 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 */
__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 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 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_ILT_MASK               0x4u
#define UART_C1_ILT_SHIFT              2
#define UART_C1_ILT_WIDTH              1
#define UART_C1_ILT(x)
(((uint8_t)(((uint8_t)(x))<<UART_C1_ILT_SHIFT))&UART_C1_ILT_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              { UART1_BASE, UART2_BASE }
/** Array initializer of UART peripheral base pointers */
#define UART_BASE_PTRS              { UART1, UART2 }

```

```

/* -----
-- 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)((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)((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)((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)((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_RESYNCDIS_MASK      0x1u
#define UART0_C5_RESYNCDIS_SHIFT      0
#define UART0_C5_RESYNCDIS_WIDTH      1
#define UART0_C5_RESYNCDIS(x)
(((uint8_t)(((uint8_t)(x))<<UART0_C5_RESYNCDIS_SHIFT))&UART0_C5_RESYNCDIS_MASK)
#define UART0_C5_BOTHEGE_MASK      0x2u
#define UART0_C5_BOTHEGE_SHIFT      1
#define UART0_C5_BOTHEGE_WIDTH      1
#define UART0_C5_BOTHEGE(x)
(((uint8_t)(((uint8_t)(x))<<UART0_C5_BOTHEGE_SHIFT))&UART0_C5_BOTHEGE_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 }

/* -----
-- 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 {  
    __IO uint8_t PERID;                /**< Peripheral ID register, offset: 0x0 */  
    __IO uint8_t RESERVED_0[3];  
    __IO uint8_t IDCOMP;               /**< Peripheral ID Complement register, offset: 0x4 */  
    __IO uint8_t RESERVED_1[3];  
    __IO uint8_t REV;                 /**< Peripheral Revision register, offset: 0x8 */  
    __IO uint8_t RESERVED_2[3];  
    __IO uint8_t ADDINFO;             /**< Peripheral Additional Info register, offset: 0xC */  
    __IO uint8_t RESERVED_3[3];  
    __IO uint8_t OTGISTAT;            /**< OTG Interrupt Status register, offset: 0x10 */  
    __IO uint8_t RESERVED_4[3];  
    __IO uint8_t OTGICR;              /**< OTG Interrupt Control Register, offset: 0x14 */  
    __IO uint8_t RESERVED_5[3];  
    __IO uint8_t OTGSTAT;             /**< OTG Status register, offset: 0x18 */  
    __IO uint8_t RESERVED_6[3];  
    __IO uint8_t OTGCTL;              /**< OTG Control register, offset: 0x1C */  
    __IO uint8_t RESERVED_7[99];  
    __IO uint8_t ISTAT;               /**< Interrupt Status register, offset: 0x80 */  
    __IO uint8_t RESERVED_8[3];  
    __IO uint8_t INTEN;               /**< Interrupt Enable register, offset: 0x84 */  
    __IO uint8_t RESERVED_9[3];  
    __IO uint8_t ERRSTAT;             /**< Error Interrupt Status register, offset: 0x88 */  
    __IO uint8_t RESERVED_10[3];  
    __IO uint8_t ERREN;               /**< Error Interrupt Enable register, offset: 0x8C */  
    __IO uint8_t RESERVED_11[3];  
    __IO uint8_t STAT;                /**< Status register, offset: 0x90 */  
    __IO uint8_t RESERVED_12[3];  
    __IO uint8_t CTL;                 /**< Control register, offset: 0x94 */  
    __IO uint8_t RESERVED_13[3];  
    __IO uint8_t ADDR;                /**< Address register, offset: 0x98 */  
    __IO uint8_t RESERVED_14[3];  
    __IO uint8_t BDTPAGE1;            /**< BDT Page Register 1, offset: 0x9C */  
    __IO uint8_t RESERVED_15[3];  
    __IO uint8_t FRMNUML;             /**< Frame Number Register Low, offset: 0xA0 */  
    __IO uint8_t RESERVED_16[3];  
    __IO uint8_t FRMNUMH;             /**< Frame Number Register High, offset: 0xA4 */  
    __IO uint8_t RESERVED_17[3];  
    __IO uint8_t TOKEN;               /**< Token register, offset: 0xA8 */  
    __IO uint8_t RESERVED_18[3];  
    __IO uint8_t SOFTHLD;             /**< SOF Threshold Register, offset: 0xAC */  
    __IO uint8_t RESERVED_19[3];  
    __IO uint8_t BDTPAGE2;            /**< BDT Page Register 2, offset: 0xB0 */  
    __IO 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];
__IO 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_SOFTHLDD_REG(base)         ((base)->SOFTHLDD)
#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_ENDPT_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)

```

```

/* IDCOMP 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_ONEMSEC_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_BSESSEND_MASK          0x4u
#define USB_OTGSTAT_BSESSEND_SHIFT        2
#define USB_OTGSTAT_BSESSEND_WIDTH        1
#define USB_OTGSTAT_BSESSEND(x)
(((uint8_t)(((uint8_t)(x))<<USB_OTGSTAT_BSESSEND_SHIFT))&USB_OTGSTAT_BSESSEND_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_MASK)
#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_MASK)
#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_MASK)
/* 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_CRC5EOF_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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
#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_MASK)
/* 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_MASK)
#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_CRC5EOFEN_MASK)
#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_MASK)
#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_MASK)
#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_BTSEERREN_MASK          0x80u
#define USB_ERREN_BTSEERREN_SHIFT         7
#define USB_ERREN_BTSEERREN_WIDTH         1
#define USB_ERREN_BTSEERREN(x)
(((uint8_t)(((uint8_t)(x))<<USB_ERREN_BTSEERREN_SHIFT))&USB_ERREN_BTSEERREN_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_USBENSOFFEN_MASK           0x1u
#define USB_CTL_USBENSOFFEN_SHIFT          0
#define USB_CTL_USBENSOFFEN_WIDTH          1
#define USB_CTL_USBENSOFFEN(x)
(((uint8_t)(((uint8_t)(x))<<USB_CTL_USBENSOFFEN_SHIFT))&USB_CTL_USBENSOFFEN_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_TXSUSPENDTOKE
NBUSY_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_BDPAGE1_BDTBA_MASK                0xFEu
#define USB_BDPAGE1_BDTBA_SHIFT               1
#define USB_BDPAGE1_BDTBA_WIDTH               7
#define USB_BDPAGE1_BDTBA(x)
(((uint8_t)(((uint8_t)(x))<<USB_BDPAGE1_BDTBA_SHIFT))&USB_BDPAGE1_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              0xFu
#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             0xFFu
#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_BDPAGE2_BDTBA_MASK           0xFFu
#define USB_BDPAGE2_BDTBA_SHIFT          0
#define USB_BDPAGE2_BDTBA_WIDTH          8
#define USB_BDPAGE2_BDTBA(x)
(((uint8_t)(((uint8_t)(x))<<USB_BDPAGE2_BDTBA_SHIFT))&USB_BDPAGE2_BDTBA_MASK)
/* BDTPAGE3 Bit Fields */
#define USB_BDPAGE3_BDTBA_MASK           0xFFu
#define USB_BDPAGE3_BDTBA_SHIFT          0
#define USB_BDPAGE3_BDTBA_WIDTH          8
#define USB_BDPAGE3_BDTBA(x)
(((uint8_t)(((uint8_t)(x))<<USB_BDPAGE3_BDTBA_SHIFT))&USB_BDPAGE3_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_DPPULLUP
NONOTG_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_MAS
K)
#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 }

/* -----
-- 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_FRMNUMML     USB_FRMNUMML_REG(USB0)
#define USB0_FRMNUMMH     USB_FRMNUMMH_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_DMACH_MASK          This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMACH_SHIFT        This_symbol_has_been_deprecated
#define DMA_REQC_ARR_DMACH(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                  MCG_S_LOLS0_MASK
#define MCG_S_LOLS_SHIFT                 MCG_S_LOLS0_SHIFT
#define SIM_FCFG2_MAXADDR_MASK           SIM_FCFG2_MAXADDR0_MASK
#define SIM_FCFG2_MAXADDR_SHIFT         SIM_FCFG2_MAXADDR0_SHIFT
#define SIM_FCFG2_MAXADDR                SIM_FCFG2_MAXADDR0
#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	UART0_Type
#define UARTLP_BDH_REG	UART0_BDH_REG
#define UARTLP_BDL_REG	UART0_BDL_REG
#define UARTLP_C1_REG	UART0_C1_REG
#define UARTLP_C2_REG	UART0_C2_REG
#define UARTLP_S1_REG	UART0_S1_REG
#define UARTLP_S2_REG	UART0_S2_REG
#define UARTLP_C3_REG	UART0_C3_REG
#define UARTLP_D_REG	UART0_D_REG
#define UARTLP_MA1_REG	UART0_MA1_REG
#define UARTLP_MA2_REG	UART0_MA2_REG
#define UARTLP_C4_REG	UART0_C4_REG
#define UARTLP_C5_REG	UART0_C5_REG
#define UARTLP_BDH_SBR_MASK	UART0_BDH_SBR_MASK
#define UARTLP_BDH_SBR_SHIFT	UART0_BDH_SBR_SHIFT
#define UARTLP_BDH_SBR(x)	UART0_BDH_SBR(x)
#define UARTLP_BDH_SBNS_MASK	UART0_BDH_SBNS_MASK
#define UARTLP_BDH_SBNS_SHIFT	UART0_BDH_SBNS_SHIFT
#define UARTLP_BDH_RXEDGIE_MASK	UART0_BDH_RXEDGIE_MASK
#define UARTLP_BDH_RXEDGIE_SHIFT	UART0_BDH_RXEDGIE_SHIFT
#define UARTLP_BDH_LBKDIE_MASK	UART0_BDH_LBKDIE_MASK
#define UARTLP_BDH_LBKDIE_SHIFT	UART0_BDH_LBKDIE_SHIFT
#define UARTLP_BDL_SBR_MASK	UART0_BDL_SBR_MASK
#define UARTLP_BDL_SBR_SHIFT	UART0_BDL_SBR_SHIFT
#define UARTLP_BDL_SBR(x)	UART0_BDL_SBR(x)
#define UARTLP_C1_PT_MASK	UART0_C1_PT_MASK
#define UARTLP_C1_PT_SHIFT	UART0_C1_PT_SHIFT
#define UARTLP_C1_PE_MASK	UART0_C1_PE_MASK
#define UARTLP_C1_PE_SHIFT	UART0_C1_PE_SHIFT
#define UARTLP_C1_ILT_MASK	UART0_C1_ILT_MASK
#define UARTLP_C1_ILT_SHIFT	UART0_C1_ILT_SHIFT
#define UARTLP_C1_WAKE_MASK	UART0_C1_WAKE_MASK
#define UARTLP_C1_WAKE_SHIFT	UART0_C1_WAKE_SHIFT
#define UARTLP_C1_M_MASK	UART0_C1_M_MASK
#define UARTLP_C1_M_SHIFT	UART0_C1_M_SHIFT
#define UARTLP_C1_RSRC_MASK	UART0_C1_RSRC_MASK
#define UARTLP_C1_RSRC_SHIFT	UART0_C1_RSRC_SHIFT
#define UARTLP_C1_DOZEEN_MASK	UART0_C1_DOZEEN_MASK
#define UARTLP_C1_DOZEEN_SHIFT	UART0_C1_DOZEEN_SHIFT
#define UARTLP_C1_LOOPS_MASK	UART0_C1_LOOPS_MASK
#define UARTLP_C1_LOOPS_SHIFT	UART0_C1_LOOPS_SHIFT
#define UARTLP_C2_SBK_MASK	UART0_C2_SBK_MASK
#define UARTLP_C2_SBK_SHIFT	UART0_C2_SBK_SHIFT
#define UARTLP_C2_RWU_MASK	UART0_C2_RWU_MASK
#define UARTLP_C2_RWU_SHIFT	UART0_C2_RWU_SHIFT
#define UARTLP_C2_RE_MASK	UART0_C2_RE_MASK
#define UARTLP_C2_RE_SHIFT	UART0_C2_RE_SHIFT
#define UARTLP_C2_TE_MASK	UART0_C2_TE_MASK
#define UARTLP_C2_TE_SHIFT	UART0_C2_TE_SHIFT
#define UARTLP_C2_ILIE_MASK	UART0_C2_ILIE_MASK
#define UARTLP_C2_ILIE_SHIFT	UART0_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
#define UARTLP_S1_RDRF_SHIFT
#define UARTLP_S1_TC_MASK
#define UARTLP_S1_TC_SHIFT
#define UARTLP_S1_TDRE_MASK
#define UARTLP_S1_TDRE_SHIFT
#define UARTLP_S2_RAF_MASK
#define UARTLP_S2_RAF_SHIFT
#define UARTLP_S2_LBKDE_MASK
#define UARTLP_S2_LBKDE_SHIFT
#define UARTLP_S2_BRK13_MASK
#define UARTLP_S2_BRK13_SHIFT
#define UARTLP_S2_RWUID_MASK
#define UARTLP_S2_RWUID_SHIFT
#define UARTLP_S2_RXINV_MASK
#define UARTLP_S2_RXINV_SHIFT
#define UARTLP_S2_MSBF_MASK
#define UARTLP_S2_MSBF_SHIFT
#define UARTLP_S2_RXEDGIF_MASK
#define UARTLP_S2_RXEDGIF_SHIFT
#define UARTLP_S2_LBKDIF_MASK
#define UARTLP_S2_LBKDIF_SHIFT
#define UARTLP_C3_PEIE_MASK
#define UARTLP_C3_PEIE_SHIFT
#define UARTLP_C3_FEIE_MASK
#define UARTLP_C3_FEIE_SHIFT
#define UARTLP_C3_NEIE_MASK
#define UARTLP_C3_NEIE_SHIFT
#define UARTLP_C3_ORIE_MASK
#define UARTLP_C3_ORIE_SHIFT
#define UARTLP_C3_TXINV_MASK
#define UARTLP_C3_TXINV_SHIFT
#define UARTLP_C3_TXDIR_MASK
#define UARTLP_C3_TXDIR_SHIFT
#define UARTLP_C3_R9T8_MASK
#define UARTLP_C3_R9T8_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
UART0_S1_RDRF_SHIFT
UART0_S1_TC_MASK
UART0_S1_TC_SHIFT
UART0_S1_TDRE_MASK
UART0_S1_TDRE_SHIFT
UART0_S2_RAF_MASK
UART0_S2_RAF_SHIFT
UART0_S2_LBKDE_MASK
UART0_S2_LBKDE_SHIFT
UART0_S2_BRK13_MASK
UART0_S2_BRK13_SHIFT
UART0_S2_RWUID_MASK
UART0_S2_RWUID_SHIFT
UART0_S2_RXINV_MASK
UART0_S2_RXINV_SHIFT
UART0_S2_MSBF_MASK
UART0_S2_MSBF_SHIFT
UART0_S2_RXEDGIF_MASK
UART0_S2_RXEDGIF_SHIFT
UART0_S2_LBKDIF_MASK
UART0_S2_LBKDIF_SHIFT
UART0_C3_PEIE_MASK
UART0_C3_PEIE_SHIFT
UART0_C3_FEIE_MASK
UART0_C3_FEIE_SHIFT
UART0_C3_NEIE_MASK
UART0_C3_NEIE_SHIFT
UART0_C3_ORIE_MASK
UART0_C3_ORIE_SHIFT
UART0_C3_TXINV_MASK
UART0_C3_TXINV_SHIFT
UART0_C3_TXDIR_MASK
UART0_C3_TXDIR_SHIFT
UART0_C3_R9T8_MASK
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	This_symbol_has_been_deprecated
#define NV_FOPT_EZPORT_DIS_SHIFT	This_symbol_has_been_deprecated
#define ADC_BASES	ADC_BASE_PTRS
#define CMP_BASES	CMP_BASE_PTRS
#define DAC_BASES	DAC_BASE_PTRS
#define DMA_BASES	DMA_BASE_PTRS
#define DMAMUX_BASES	DMAMUX_BASE_PTRS
#define FPTA_BASE_PTR	FGPIOA_BASE_PTR
#define FPTA_BASE	FGPIOA_BASE
#define FPTA	FGPIOA

```

#define FPTB_BASE_PTR      FGPIOB_BASE_PTR
#define FPTB_BASE          FGPIOB_BASE
#define FPTB              FGPIOB
#define FPTC_BASE_PTR      FGPIOC_BASE_PTR
#define FPTC_BASE          FGPIOC_BASE
#define FPTC              FGPIOC
#define FPTD_BASE_PTR      FGPIOD_BASE_PTR
#define FPTD_BASE          FGPIOD_BASE
#define FPTD              FGPIOD
#define FPTE_BASE_PTR      FGPIOE_BASE_PTR
#define FPTE_BASE          FGPIOE_BASE
#define FPTE              FGPIOE
#define FGPIIO_BASES        FGPIIO_BASE_PTRS
#define FTFA_BASES          FTFA_BASE_PTRS
#define PTA_BASE_PTR        GPIOA_BASE_PTR
#define PTA_BASE           GPIOA_BASE
#define PTA               GPIOA
#define PTB_BASE_PTR        GPIOB_BASE_PTR
#define PTB_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. */
/*
** #####
**
** 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:            b150220
**
** 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: AVLPE=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. */

#ifdef 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,EREFS0=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=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,EREFS=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,EREFS0=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,EREFS=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_BLP1 /* 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,EREFS0=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,EREFS=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 */
#elif (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,EREFS0=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=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,EREFS=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 */
#elif (CLOCK_SETUP == 4)
#define DEFAULT_SYSTEM_CLOCK    4800000u /* 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,EREFS0=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,EREFS0=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 */

```

```

#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);

#ifdef __cplusplus
}
#endif

#endif /* #if !defined(SYSTEM_MKL25Z4_H_) */
/**
 * @file platform.h
 * @brief contains platform necessary includes and macros
 *
 * This contains the defines for the PRINTF macro and other functionality,
 * as well as ensuring the PRINTF macro doesn't cause errors
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 */
#ifndef __PLATFORM_H__
#define __PLATFORM_H__

/* defines to see if we have a PRINTF macro */
#ifdef (__GNUC__)
#pragma GCC diagnostic ignored "-Wunused-but-set-variable"
#endif

#ifdef KL25Z
#define PRINTF(...)
#else
#include <stdio.h>
#define PRINTF(str, ...) printf(str, ##__VA_ARGS__)
#endif

#endif /* __PLATFORM_H__ */
/**
 * @file arch_arm32.c
 * @brief implementation of arch_arm32.h
 *
 * contains the functional implementation of endianness ARM lookup
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 */
#include "arch_arm32.h"

```



```

void InitSysTick()
{
    /* SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); * disable counting */
    SysTick_Base_Ptr->CSR = 0; /* disable counting */
    SysTick_Base_Ptr->RVR = 0xFFFFF; /* max value */
    SysTick_Base_Ptr->CSR = __SYSTICK_CLKSOURCE_MASK;
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    /* | __SYSTICK_TICKINT_MASK; * enable the interrupt handler */
}

void SysTick_Handler()
{
}

__attribute__((always_inline))
uint32_t inline ARM32_AIRCR_get_endianness_setting()
{
    return ( (__AIRCR & __AIRCR_ENDIANNESSE_MASK) >> __AIRCR_ENDIANNESSE_OFFSET );
}

__attribute__((always_inline))
uint32_t inline ARM32_CCR_get_stack_alignment()
{
    volatile uint32_t *CCR = (uint32_t *)__CCR; /* volatile for instant access */
    return ((*CCR & __CCR_STK_ALIGNMENT_MASK) >> __CCR_STK_ALIGNMENT_OFFSET);
}

__attribute__((always_inline))
uint32_t inline ARM32_CPUID_get_part_number()
{
    volatile uint32_t *CPUIDReg = (uint32_t *)__CPUID; /* volatile for instant access */
    return ((*CPUIDReg & __CPUID_PART_NO_MASK) >> __CPUID_PART_NO_OFFSET);
}

__attribute__((always_inline))
void inline ARM32_CCR_enable_divide_by_zero_trap()
{
    volatile uint32_t *CCR = (uint32_t *)__CCR; /* volatile for instant access */
    *CCR |= (1 << __CCR_DIVIDE_BY_ZERO_TRAP_OFFSET);
    return;
}

__attribute__((always_inline))
void inline ARM32_CCR_enable_unaligned_access_trap()
{
    volatile uint32_t *CCR = (uint32_t *)__CCR; /* volatile for instant access */
    *CCR |= (1 << __CCR_UNALIGNED_ACCESS_TRAP_OFFSET);
    return;
}

void inline ARM32_create_unaligned_access_trap()

```

```

{
    uint8_t unaligned_ptr;
#pragma GCC diagnostic ignored "-Wunused-variable"
    uint32_t *ptr = (uint32_t *)(&unaligned_ptr);
    /* this should never get here */
    while(1);
}

```

```

void inline ARM32_create_divide_by_zero_trap()
{
#pragma GCC diagnostic ignored "-Wdiv-by-zero"
#pragma GCC diagnostic ignored "-Wunused-variable"
    uint8_t trap = (1/0);
    /* this should never get here */
    while(1);
}

```

```

/**
 * @file circbuf.c
 * @brief implementation of circbuf.h
 *
 * implements the circular buffer access functions found in circbuf.h
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 *
 */
#include "circbuf.h"

/* standard library for malloc */
#include <stdlib.h>

/* @brief initializes the circular buffer with default values
 *
 * the circular buffer must already be allocated when it is
 * passed in. The function will then create dynamic memory
 * to allocate the entire buffer. It will then set the
 * head, tail, and other values to be what they should be.
 * destroy must be called before free-ing the CB that
 * is passed in, otherwise a stack overflow may occur.
 *
 *
 * @param[in] CB_t* the CB_t object to initialize
 * @param[in] size_t the size of the buffer to initialize
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */

```

```

CB_e CB_init(CB_t* circbuff, size_t buffer_size)
{
    if ((int)buffer_size == 0)
        return NO_LENGTH;
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
}

```

```

    circbuff->buff_size = buffer_size;
    circbuff->base = (BUFFER_TYPE*)malloc(((int)buffer_size)*sizeof(BUFFER_TYPE));
    circbuff->head = circbuff->base;
    circbuff->tail = circbuff->base;
    circbuff->num_in = 0;
    circbuff->buff_empty_flag = SET;
    circbuff->buff_full_flag = UNSET;
    circbuff->buff_ovf_flag = UNSET;
    circbuff->buff_destroyed_flag = UNSET;
    if ((void*)circbuff->base == NULL)
    {
        circbuff->buff_size = 0;
        circbuff->buff_full_flag = SET;
        circbuff->buff_destroyed_flag = SET;
        circbuff->buff_ovf_flag = SET;
        return NO_BUFFER_IN_MEMORY;
    }
    return SUCCESS;
}

```

```

/* @brief frees the entire circular buffer
 *
 * this function simply frees the dynamic memory that it allocated
 * when initializing the function.
 *
 * @param[in] CB_t* the CB_t object to destroy
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */

```

```

CB_e CB_destroy(CB_t* circbuff)
{
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
    /* we don't want to free something twice */
    if ((CB_f)circbuff->buff_destroyed_flag == SET)
        return DOUBLE_FREE;
    free((void*)circbuff->base);
    circbuff->buff_destroyed_flag = SET;
    return SUCCESS;
}

```

```

/* @brief adds an item to the circular buffer
 *
 * add an item in memory to the circular buffer
 * increment the head, and update all the flags
 *
 *
 * @param[in] CB_t* the CB_t object to operate on
 * @param[in] BUFFER_TYPE the object to add into the buffer
 * @return CB_e This function returns the CB_e typedef to indicate errors
 */

```

```

CB_e CB_buffer_add_item(CB_t* circbuff, BUFFER_TYPE data)
{

```

```

if ((void*)circbuff == NULL)
    return BAD_POINTER;
/* if we have already destroyed the buffer */
if ((CB_f)circbuff->buff_destroyed_flag == SET)
    return NO_BUFFER_IN_MEMORY;
/* the buffer is already full */
if ((CB_f)circbuff->buff_full_flag == SET)
{
    circbuff->buff_ovf_flag = SET;
    /* data is trashed */
    return FULL;
}
END_CRITICAL();
*circbuff->head = data;
circbuff->head = circbuff->head + 1; /* or just circbuff->head++; */
circbuff->num_in++;
circbuff->buff_empty_flag = UNSET;
/* it's a circular buffer, so loop around if we go beyond the max */
if (circbuff->head > (circbuff->base + (circbuff->buff_size - 1)))
{
    circbuff->head = circbuff->base;
}
/* the buffer must be full since we added something and the following is true*/
if (circbuff->head == circbuff->tail)
{
    circbuff->buff_full_flag = SET;
    /* this should never happen */
    if (circbuff->num_in != circbuff->buff_size)
    {
        START_CRITICAL();
        return CRITICAL_ERROR;
    }
}
START_CRITICAL();
return SUCCESS;
}

```

/\* @brief removes an item from the circular buffer

\*

\* removes an item from the circular buffer by

\* incrementing the tail and decrementing the num\_in

\*

\* @param[in] CB\_t\* the CB\_t object to operate on

\* @param[out] BUFFER\_TYPE\* put the data removed into this pointer

\* @return CB\_e This function returns the CB\_e typedef to indicate errors

\*/

CB\_e CB\_buffer\_remove\_item(CB\_t\* circbuff, BUFFER\_TYPE\* data)

```

{
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
    /* if we have already destroyed the buffer */
    if ((CB_f)circbuff->buff_destroyed_flag == SET)

```

```

    return NO_BUFFER_IN_MEMORY;
if ((CB_f)circbuff->buff_empty_flag == SET)
    return EMPTY;
END_CRITICAL();
*data = *circbuff->tail;
circbuff->tail = circbuff->tail + 1; /* or just circbuff->tail++; */
circbuff->num_in--;
circbuff->buff_full_flag = UNSET;
/* it's a circular buffer, so loop around if we go beyond the max */
if (circbuff->tail > (circbuff->base + (circbuff->buff_size - 1)))
{
    circbuff->tail = circbuff->base;
}
/* the buffer must be empty since we removed something and the following is true */
if (circbuff->head == circbuff->tail)
{
    circbuff->buff_empty_flag = SET;
    /* this should never happen */

    if (circbuff->num_in != 0)
    {
        START_CRITICAL();
        return CRITICAL_ERROR;
    }
}
START_CRITICAL();
return SUCCESS;
}

```

```

/* @brief checks to see if the buffer is full
*
* simply checks the full flag of the buffer to see
* if the head ever passed the tail
*
* @param[in] CB_t* the CB_t object to operate on
* @return CB_e This function returns the CB_e typedef to indicate errors
*/

```

```

CB_e CB_is_full(CB_t* circbuff)
{
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
    if ((CB_f)circbuff->buff_full_flag == SET)
        return FULL;
    return SUCCESS;
}

```

```

/* @brief checks to see if the buffer is empty
*
* simply checks the empty flag of the buffer
* to see if the head and tail are equal and
* there is no data in the buffer

```

```

*
*
* @param[in] CB_t* the CB_t object to operate on
* @return CB_e This function returns the CB_e typedef to indicate errors
*/
CB_e CB_is_empty(CB_t* circbuff)
{
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
    if ((CB_f)circbuff->buff_empty_flag == SET)
        return EMPTY;
    return SUCCESS;
}

/* @brief returns the value in the buffer at a position back from the head
*
* peeking at the 0th value will just return the last value
* that was put into the buffer.
*
* @param[in] CB_t* the CB_t object to operate on
* @param[in] size_t the index away from the head
* @param[out] BUFFER_TYPE* put the data peeked at into this pointer
* @return CB_e This function returns the CB_e typedef to indicate errors
*/
CB_e CB_peek(CB_t* circbuff, size_t position, BUFFER_TYPE* data)
{
    BUFFER_TYPE* peekdata = (BUFFER_TYPE*)circbuff->head;
    if ((void*)circbuff == NULL)
        return BAD_POINTER;
    /* since head points at an empty value we increment by one */
    position++;
    if (position > circbuff->num_in)
        return POSITION_TOO_LARGE;
    /* it's a circular buffer, so loop around if we go beyond the max */
    if (peekdata < circbuff->base)
    {
        peekdata = 1*(size_t)(circbuff->buff_size-1) + peekdata;
    }
    *data = *(BUFFER_TYPE*)peekdata;
    return SUCCESS;
}

/*
* @file conversion.c
* @brief this file implements conversion.h
*
* contains the implementation of integer to array and array to integer
* conversions as well as an exponentiation function
*
* @author Seth Miers and Jake Cazden
* @date February 11, 2018
*
* */

```

```
#include "conversion.h"
```

```
/* used for the my_reverse function */
```

```
#include "memory.h"
```

```
int32_t exponent(int32_t base,int32_t power)
```

```
{  
    if(power<0) return 0;  
    uint32_t i=0;  
    int32_t retval=1;  
    for(i=1;i<=power;i++)  
    {  
        retval*=base;  
    }  
    return retval;  
}
```

```
uint8_t my_itoa(int32_t data, uint8_t * ptr, uint32_t base)
```

```
{  
    if(!ptr)  
    {  
        return (uint8_t)0;  
    }  
    uint8_t length=0;  
    uint8_t negative=0;  
    if(data == 0)  
        return 0;  
    if(base<BASE_2||base>BASE_16)  
    {  
        return 0;  
    }  
    if(data<0)  
    {  
        *ptr++='-';  
        data=-data; /*make the number positive*/  
        negative=1;  
    }  
    while(1) /* figure out the magnitude of the number */  
    {  
        *(ptr+length) = data%base;  
        data = data/base;  
        length++;  
        if ((data == 0) || (length > BASE_2_MAXDIGITS ))  
            break;  
    }  
    /* we want to go from smallest to biggest not visaversa */  
    my_reverse(ptr, length);  
    int32_t j=length-1;  
    /* length-1 becausee the while loop puts you one order of magnitude high */  
    uint8_t num;  
    for(j=length-1;j>=0;j--)  
    {
```

```

    num = *(ptr+j);
    if(num>9)
    {
        *(ptr+j)=num+ASCII_OFFSET_A_ADDITION;
    }
    else
    {
        *(ptr+j)=num+ASCII_OFFSET_0;
    }
}
length = length + negative;
return length;
}

```

```

int32_t my_atoi(uint8_t * ptr, uint8_t digits, uint32_t base)
{
    if(!ptr)
    {
        return (uint32_t)0;
    }
    int8_t i = 0;
    uint8_t negative = 0;
    int32_t return_value = 0;
    if(digits == 0)
        return 0;
    /* check to see if the integer is negative*/
    if(*ptr=='-')/* using int is okay here */
    {
        negative = 1;
        ptr++;
        digits--;
    }
    /* base starts at 0 so subtract one more */
    digits--;
    /* loop through each digit and add it to the return_value*/
    for(i=digits;i>=0;i--)
    {
        /* the digit is between a 0 and a 9 */
        if (*ptr >= ASCII_OFFSET_0 && *ptr <= ASCII_OFFSET_9)
        {
            if((*ptr - ASCII_OFFSET_0) > base)
                /* an error has occurred, we've encountered an unsupported character */
                return 0;
            return_value += (*ptr++ - ASCII_OFFSET_0) * exponent(base, i);
        }
        /* the digit is between an uppercase A and a F */
        else if (*ptr >= ASCII_OFFSET_A && *ptr <= ASCII_OFFSET_F)
        {
            if((*ptr - ASCII_OFFSET_A_ADDITION) > base)
                /* an error has occurred, we've encountered an unsupported character */
                return 0;
            return_value += (*ptr++ - ASCII_OFFSET_A_ADDITION) * exponent(base, i);
        }
    }
}

```



```

    }
    /* the digit is between a lowercase a and a f */
    else if (*ptr >= ASCII_OFFSET_LA && *ptr <= ASCII_OFFSET_LF)
    {
        if((*ptr - ASCII_OFFSET_LA_ADDITION) > base)
            /* an error has occurred, we've encountered an unsupported character */
            return 0;
        return_value += (*ptr++ - ASCII_OFFSET_LA_ADDITION) * exponent(base, i);
    }
    else
    {
        /* an error has occurred, we've encountered an unsupported character */
        return 0;
    }
}
if(negative)
{
    return -return_value;
}
return return_value;
}
/**
 * @file data.h
 * @brief implementation of data.h
 *
 * implements functions to print type information, as well as determine
 * endianness and swap data endianness in a platform independent manner
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 *
 */
#include "data.h"

/* needed to check endianness */
#include "memory.h"

/* needed for printf macro */
#include "platform.h"

/*
#ifdef __GNUC__
#pragma GCC diagnostic ignored "-Wunused-but-set-variable"
#endif
*/
void print_cstd_type_sizes()
{
    /*this function is pretty straightforward, it prints a list of types and
    their sizes*/
    size_t temp=sizeof(char);
    PRINTF("char:      %zd\n",temp);
    temp=sizeof(short);
    PRINTF("short:      %zd\n",temp);

```

```

temp=sizeof(int);
PRINTF("int:      %zd\n",temp);
temp=sizeof(long);
PRINTF("long:      %zd\n",temp);
temp=sizeof(double);
PRINTF("double:     %zd\n",temp);
temp=sizeof(float);
PRINTF("float:      %zd\n",temp);
temp=sizeof(unsigned char);
PRINTF("unsigned char: %zd\n",temp);
temp=sizeof(unsigned int);
PRINTF("unsigned int:  %zd\n",temp);
temp=sizeof(unsigned long);
PRINTF("unsigned long: %zd\n",temp);
temp=sizeof(signed char);
PRINTF("signed char:   %zd\n",temp);
temp=sizeof(signed int);
PRINTF("signed int:    %zd\n",temp);
temp=sizeof(signed long);
PRINTF("signed long:   %zd\n",temp);
return;
}

void print_stdint_type_sizes()
{ /* this function is pretty straightforward, it prints a list of stdint types
and their sizes*/
size_t temp=sizeof(int8_t);
PRINTF("int8_t:      %zd\n",temp);
temp=sizeof(uint8_t);
PRINTF("uint8_t:      %zd\n",temp);
temp=sizeof(uint16_t);
PRINTF("uint16_t:     %zd\n",temp);
temp=sizeof(int32_t);
PRINTF("int32_t:      %zd\n",temp);
temp=sizeof(uint32_t);
PRINTF("uint32_t:     %zd\n",temp);
temp=sizeof(uint_fast8_t);
PRINTF("uint_fast8_t:  %zd\n",temp);
temp=sizeof(uint_fast16_t);
PRINTF("uint_fast16_t: %zd\n",temp);
temp=sizeof(uint_fast32_t);
PRINTF("uint_fast32_t: %zd\n",temp);
temp=sizeof(uint_least8_t);
PRINTF("uint_least8_t: %zd\n",temp);
temp=sizeof(uint_least16_t);
PRINTF("uint_least16_t: %zd\n",temp);
temp=sizeof(uint_least32_t);
PRINTF("uint_least32_t: %zd\n",temp);
temp=sizeof(size_t);
PRINTF("size_t:      %zd\n",temp);
temp=sizeof(ptrdiff_t);
PRINTF("ptrdiff_t:    %zd\n",temp);
return;

```

```

}
void print_pointer_sizes()
{
    size_t temp=sizeof(char*);
    PRINTF("char*:      %zd\n",temp);
    temp=sizeof(short*);
    PRINTF("short*:      %zd\n",temp);
    temp=sizeof(int*);
    PRINTF("int*:        %zd\n",temp);
    temp=sizeof(double*);
    PRINTF("double*:      %zd\n",temp);
    temp=sizeof(float*);
    PRINTF("float*:       %zd\n",temp);
    temp=sizeof(void*);
    PRINTF("void*:        %zd\n",temp);
    temp=sizeof(int8_t*);
    PRINTF("int8_t*:      %zd\n",temp);
    temp=sizeof(int16_t*);
    PRINTF("int16_t*:     %zd\n",temp);
    temp=sizeof(int32_t*);
    PRINTF("int32_t*:     %zd\n",temp);
    temp=sizeof(char**);
    PRINTF("char**:       %zd\n",temp);
    temp=sizeof(int**);
    PRINTF("int**:        %zd\n",temp);
    temp=sizeof(void**);
    PRINTF("void**:       %zd\n",temp);
    return;
}

int32_t swap_data_endianness(uint8_t * data, size_t type_length)
{
    /*TODO make this function nondestructive on failure if possible*/
    if(!data) return SWAP_ERROR; /*if null, return error*/
    uint8_t test[type_length]; /*block out a region on the stack*/
    my_memmove(data,test,type_length); /*create a copy of 'data' on the stack*/
    my_reverse(data , type_length); /*reverse data*/
    uint32_t i;
    for(i=0;i<(type_length/2);i++)
    {
        if(*(test+i)!=*(data+type_length-1-i))
        {
            return SWAP_ERROR;
        }
    }
    return SWAP_NO_ERROR;
}

uint32_t determine_endianness()
{
    uint32_t test = 0x00BADA55; /*standard test word*/
    uint8_t* ptr = (uint8_t*)&test;
    if(*(ptr)==5) /*LSB of 00BADA55*/
    {
        return LITTLE_ENDIAN;
    }
}

```

```

    }
    else
    {
        return BIG_ENDIAN;
    }
}
/**
 * @file debug.c
 * @brief implements the prototypes in debug.h
 *
 * implements a printing function to hexdump from a location in memory
 * the function should print in a manner similar to a common unix hexdump
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 *
 */

```

```

#include "debug.h"

```

```

/*included due to use of PRINTF macro*/
#include "platform.h"

```

```

/*
 * @brief function to print the bytes in memory starting at an address
 *
 * This function takes in a pointer to the start of some
 * memory region and starts printing out hex bytes
 * for the number of bytes specified.
 *
 * @param start a pointer to the start of the memory region
 * @param length the number of bytes to read back
 * @return void don't return anything
 */

```

```

void print_array(void * start, uint32_t length)
{
    #ifdef VERBOSE
    uint32_t i=0;
    uint8_t* temp=(uint8_t*)start;
    for(i=0;i<length;i++)
    {
        if((i)%16==0)/*every 16 bytes (zero inclusive), print the address*/
        {
            PRINTF("%p ",(void*)temp);
        }
        PRINTF("%02X ",*(temp++));
        if((i+1)%16==0)/*after every 16 bytes, print a newline.*/
        {
            PRINTF("\n");/*eg, addr 0-15 \n addr 16-31*/
        }
    }
    #endif
}

```

```

    return;
}
/**
 * @file main.c
 * @brief calls test functionality for project 1 on multiple platorms
 *
 * this main file supports calling test functions on the project 1
 * function implementations. This will be expanded later to include
 * more functionality
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2017
 */
#include <stdlib.h>
#ifdef PROJECT1
    #include "project1.h"
    #include "circbuf.h"
#endif
#ifdef PROJECT2
    #include "project2.h"
    #include "circbuf.h"
#endif
#ifdef PROJECT3
    #include "project3.h"
    #include "circbuf.h"
#endif
#ifdef CMOCKA
    #include "unittest.h"
#endif
#ifdef DEBUG
#include <stdio.h>
#endif
/*
#ifdef KL25Z
#define CLOCK_SETUP (0)
#endif
*/

#if defined(PROJECT2) || defined(PROJECT3)
/* static to retain in any scope, const so that the compiler will complain if we touch this from this file */
CB_t* recieve_buffer;
CB_t* transmit_buffer;
uint8_t dma0_done=0;
volatile uint32_t DMA_end_value = 0;
#endif
#ifdef KL25Z
volatile uint8_t dma_first_setup = 0;
volatile uint8_t dma_error_flag = 0;
#endif
volatile uint32_t nooperation=0;
int main(void)

```

```

{
    #ifdef CMOCKA
return unittest();
    #endif

    #ifdef PROJECT1
        project1();
    #endif

    #ifdef PROJECT2
        recieve_buffer = (CB_t*) malloc(sizeof(CB_t));
        transmit_buffer = (CB_t*) malloc(sizeof(CB_t));
        project2();
    #endif

    #ifdef PROJECT3
        recieve_buffer = (CB_t*) malloc(sizeof(CB_t));
        transmit_buffer = (CB_t*) malloc(sizeof(CB_t));
        project3();
    #endif

    return 0;
}
/**
 * @file memory.c
 * @brief implements memory.h
 *
 * implements software defined functions for copying, allocating, and freeing
 * memory, without relying on hardware specifically.
 *
 * @author Seth Miers and Jake Cazden
 * @date February 11, 2018
 *
 */
#include "memory.h"

/* Only need the following if we cant use
 * built in malloc and free functions
 * extern void __HeapBase, __HeapLimit;
 */

/* Needed to return null pointers
 * This is defined in stddef so it's commented out
 *
 * #ifndef NULL
 * #define NULL ((void*)0)
 * #endif
 */

#include <stdlib.h>
#include <stddef.h>
#ifdef KL25Z

```

```

#include "MKL25Z4.h"
#include "arch_arm32.h"
extern uint8_t dma_first_setup;
#endif
extern volatile uint8_t dma_error_flag;

extern uint8_t dma0_done;
extern volatile uint32_t DMA_end_value;
extern volatile uint32_t nooperation;

#pragma GCC push_options
#pragma GCC optimize ("O0")
uint8_t * memset_dma(uint8_t * src, size_t length, uint8_t value, size_t transfer)
{
    /* TODO implement function */
#ifdef KL25Z
    dma0_done=0;
    uint32_t valholder= (value)+(value<<8)+(value<<16)+(value<<24);
    DMA_e retval = setup_memtransfer_dma((uint8_t *)&valholder, 1, src, transfer, length);
    while(dma0_done==0 && retval==DMA_SUCCESS)
    {
        nooperation++;
    }
    if(retval==DMA_ERROR)
    {
        return NULL;
    }
    return src;
#else
    return my_memset(src, length, value);
#endif
}
#pragma GCC pop_options

#pragma GCC push_options
#pragma GCC optimize ("O0")
uint8_t * memmove_dma(uint8_t * src, uint8_t * dst, size_t length, size_t transfer)
{
#ifdef KL25Z
    if(dst>src && dst<src+length)
    {
        /*the case where overlap will occur, but isn't exactly on top of itself*/
        dma0_done=0;
        uint32_t templength1 = ((uint32_t)src+length)-(uint32_t)dst;
        uint32_t templength2 = (uint32_t)dst-(uint32_t)src;
        DMA_e retval = setup_memtransfer_dma(dst, templength1, src+length, transfer, templength1);
        while(dma0_done==0 && retval==DMA_SUCCESS)
        {
            nooperation++;
        }
        if(retval==DMA_ERROR)
        {
            return NULL;
        }
    }

```

```

    }
    dma0_done=0;
    retval = setup_memtransfer_dma(src, templength2, dst, transfer, templength2);
    while(dma0_done==0 && retval==DMA_SUCCESS)
    {
        nooperation++;
    }
    if(retval==DMA_ERROR)
    {
        return NULL;
    }
}
else
{
    dma0_done=0;
    DMA_e retval = setup_memtransfer_dma(src, length, dst, transfer, length);
    while(dma0_done==0 && retval==DMA_SUCCESS)
    {
        nooperation++;
    }
    if(retval==DMA_ERROR)
    {
        return NULL;
    }
}
return dst;
#else
return my_memmove(src, dst, length);
#endif
}
#pragma GCC pop_options

```

```

uint8_t * my_memmove(uint8_t * src, uint8_t * dst, size_t length)
{
    /* check for null pointers */
    if(!dst)
    {
        return NULL;
    }
    if(!src)
    {
        return NULL;
    }
    /* create a new array to copy into to prevent corruption */
    uint8_t my_array[length];
    /* copy our current array into the new array */
    my_memcpy(src, my_array, length);
    /* copy the new array into the destination array*/
    return my_memcpy(my_array, dst, length);
}

```

```

uint8_t * my_memcpy(uint8_t * src, uint8_t * dst, size_t length)

```



```

{
    /* check for null pointers */
    if(!dst)
    {
        return NULL;
    }
    if(!src)
    {
        return NULL;
    }
    /* create a variable to iterate through (for loop) */
    size_t i = 0;
    /* save the original destination since we'll be changing its value */
    uint8_t* orig_dst = dst;
    /* loop for how many bytes there are */
    for(i=0;i<length;i++)
    {
        /* copy the source to the destination */
        *dst++ = *src++;
    }
    return orig_dst;
}

```

```

uint8_t * my_memset(uint8_t * src, size_t length, uint8_t value)
{
    /* check for null pointers */
    if(!src)
    {
        return NULL;
    }
    /* create a variable to iterate through (for loop) */
    size_t i = 0;
    /* loop for how many bytes there are */
    for(i=0;i<length;i++)
    {
        /* copy the value into the source */
        *src++ = value;
    }
    return src;
}

```

```

uint8_t * my_memzero(uint8_t * src, size_t length)
{
    /* just call my_memset with a value of 0 */
    return my_memset(src, length, 0);
}

```

```

uint8_t * my_reverse(uint8_t * src, size_t length)
{
    /* check for null pointers */
    if(!src)
    {

```

```

    return NULL;
}
/* create a temporary array */
uint8_t my_array[length];
/* copy the source into the temporary array */
my_memcpy(src, my_array, length);
/* create an address that points to the end of the array */
uint8_t * my_array_ptr = &my_array[length-1];
/* loop for how many bytes there are */
size_t i = 0;
for(i=0;i<length;i++)
{
    /* copy the destination of the pointer
     * of the end of the array into the srouce */
    *src++ = *my_array_ptr--;
}
return src;
}

void * reserve_words(size_t length)
{
    void* src;
    /* malloc will return null if it fails */
    src = (void*)malloc(length*sizeof(void*));
    return src;
}

uint8_t free_words(void * src)
{
    free(src);
    return 0;
}

#ifdef KL25Z
DMA_e setup_memtransfer_dma(uint8_t* src, uint8_t src_len, uint8_t* dst,
                             size_t transfersize, size_t length)/*, uint8_t dma_index)*/
{
    /*if(dma_index>4) return BAD_INDEX;*/
    if(src_len==0 || length==0) return DMA_NO_LENGTH;
    if(src==NULL || dst==NULL) return DMA_BAD_POINTER;
    if(dma_first_setup==0)
    {
        /*turn on clock gates to the mux and dma modules*/
        SIM_SCGC6 |= SIM_SCGC6_DMAMUX(DMAMUX_CLOCKGATE_ENABLE);
        SIM_SCGC7 |= SIM_SCGC7_DMA(DMA_CLOCKGATE_ENABLE);
        dma_first_setup=1;
    }
    /*if((DMA_DSR_BCR0&DMA_DSR_BCR_DONE_MASK)>>DMA_DSR_BCR_DONE_SHIFT!=1)
    {
        if a transfer is active, error out TODO this could optionally
        search for an open channel instead
        return DMA_BUSY;
    }*/
}

```

```

/*enable dma channel 0, set it to non-periodic, and attach it to an always
 * active request source*/
/*clear errors and disable active transfer*/
DMA_DSR_BCR0 |= DMA_DSR_BCR_DONE(DMA_DSR_BCR_DONE_WRITE_TOCLEAR);
DMAMUX0_CHCFG0 = DMAMUX_CHCFG_ENBL(DMAMUX_CHCFG_ENABLE)
                 |DMAMUX_CHCFG_TRIG(DMAMUX_CHCFG_SINGLE_TRIGGER)
                 |DMAMUX_CHCFG_SOURCE(DMAMUX_CHCFG_SOURCE_ALWAYS_ON_60);
uint32_t checkaddr = ((uint32_t)src & (uint32_t)0xffff0000) >> 20;
if(checkaddr == 0x000 || checkaddr == 0x1ff || checkaddr == 0x200 || checkaddr == 0x400)
{
    /*if the source is allowed, put it into the source register*/
    DMA_SAR0 = (uint32_t)src;
}
else return DMA_BAD_POINTER;
checkaddr = ((uint32_t)dst & (uint32_t)0xffff0000) >> 20;
if(checkaddr == 0x000 || checkaddr == 0x1ff || checkaddr == 0x200 || checkaddr == 0x400)
{
    /*if the destination is allowed, put it into the destination register*/
    DMA_DAR0 = (uint32_t)dst;
}
else return DMA_BAD_POINTER;
/*if the transfer is too large, turn it off*/
if(length < DMA_DSR_BCR_BCR_MAXVALUE)
{
    /*put the number of bytes into the byte count register*/
    /*TODO might need to clear the BCR register as well.*/
    DMA_DSR_BCR0 |= DMA_DSR_BCR_BCR(length);
}
else return DMA_BCR_LENGTH_OVERFLOW;
/*enable interrupt on complete or error, keep peripheral requests off,
 * allow for continuous (non-cycle-steal) operation, dont auto-align
 * dont allow asynchronous requests during low power, enable the transfer
 * disable writing into and from a circular buffer, turn off the ability of
 * the dma to disable it's source request (peripheral operation only)
 * and turn off channel linking. set the destination to increment.*/
uint32_t DCRregwrite = DMA_DCR_EINT(DMA_DCR_INTERRUPT_ON_COMPLETE)
                      |DMA_DCR_ERQ(DMA_DCR_NO_PERIPHERAL_REQUEST)
                      |DMA_DCR_CS(DMA_DCR_CONTINUOUS_OPERATION)
                      |DMA_DCR_EADREQ(DMA_DCR_NO_ASYNC_REQUESTS)
                      |DMA_DCR_START(DMA_DCR_START_ENABLE)
                      |DMA_DCR_SMOD(DMA_DCR_NO_SOURCE_MODULO)
                      |DMA_DCR_DMOD(DMA_DCR_NO_DEST_MODULO)
                      |DMA_DCR_D_REQ(DMA_DCR_DISABLE_REQUEST_OFF)
                      |DMA_DCR_LINKCC(DMA_DCR_CHANNEL_LINK_DISABLED)
                      |DMA_DCR_DINC(DMA_DCR_INCREMENT_DEST);
if(src_len == 1) /*set the source to increment iff there is more than 1 byte of source*/
{
    DCRregwrite |= DMA_DCR_SINC(DMA_DCR_NO_SOURCE_INCREMENT);
}
else
{
    DCRregwrite |= DMA_DCR_SINC(DMA_DCR_INCREMENT_SOURCE);
}

```

```

}
if(transfersize==1 || length<=16)/*set transfer size if passed appropriately*/
{
    DCRregwrite|=DMA_DCR_SSIZE(DMA_DCR_TRANSFERSIZE_8BIT)
                |DMA_DCR_DSIZE(DMA_DCR_TRANSFERSIZE_8BIT)
                |DMA_DCR_AA(DMA_DCR_NO_AUTOALIGN);
}
else if(transfersize==2)
{
    DCRregwrite|=DMA_DCR_SSIZE(DMA_DCR_TRANSFERSIZE_16BIT)
                |DMA_DCR_DSIZE(DMA_DCR_TRANSFERSIZE_16BIT)
                |DMA_DCR_AA(DMA_DCR_AUTOALIGN);
}
else if(transfersize==4)
{
    DCRregwrite|=DMA_DCR_SSIZE(DMA_DCR_TRANSFERSIZE_32BIT)
                |DMA_DCR_DSIZE(DMA_DCR_TRANSFERSIZE_32BIT)
                |DMA_DCR_AA(DMA_DCR_AUTOALIGN);
}
else return DMA_BAD_SIZE;

NVIC_ClearPendingIRQ(DMA0_IRQn);/*enable interrupts on DMA0 ocmplete and error*/
NVIC_EnableIRQ(DMA0_IRQn);
__enable_irq();

DMA_DCR0=DCRregwrite;/*write DMA control register to start transfer*/
return DMA_SUCCESS;
}
#endif
#ifdef KL25Z
#pragma GCC push_options
#pragma GCC optimize ("O0")
void DMA0_IRQHandler()
{
    DMA_end_value = SysTick_Base_Ptr->CVR;
    NVIC_ClearPendingIRQ(DMA0_IRQn);
    NVIC_DisableIRQ(DMA0_IRQn);
    DMAMUX0_CHCFG0 = 0;
    DMA_DCR0=0;
    DMA_SAR0=0;
    DMA_DAR0=0;
    if(DMA_DSR_BCR(0)&(DMA_DSR_BCR_CE_MASK))
    {
        dma_error_flag=1;
    }
    DMA_DSR_BCR0 |= DMA_DSR_BCR_DONE(DMA_DSR_BCR_DONE_WRITETOCLEAR);
    dma0_done=1;
}
#pragma GCC pop_options
#endif
/*
 * @file nordic.c

```

```

* @brief
*
* it's assumed that the NRF is connected to the KL25Z in the following manner
*
*   NRF   |   KL25Z
* -----|-----
*   GND   ->  GND
*   VCC   ->  3.3V
*   CSN   ->  PTD0
*   CE    ->  PTD5
*   SCK   ->  PTD1
*   MOSI  ->  PTD2
*   MISO  ->  PTD3
*   IRQ   ->  NC
*
* @author Seth Miers and Jake Cazden
* @date March 15, 2018
*/

```

```

#include "nordic.h"
#include "port.h"
#include "spi.h"

```

```

uint8_t nrf_read_register(uint8_t readRegister)
{
    uint8_t command = 0x1F & readRegister;
    uint8_t readByte = 0x00;
    nrf_chip_enable();
    SPI_write_byte(command);
    SPI_read_byte(&readByte);
    nrf_chip_disable();
    return readByte;
}

```

```

void nrf_write_register(uint8_t writeRegister, uint8_t value)
{
    uint8_t command = 0x20 | (0x3F & writeRegister);
    nrf_chip_enable();
    SPI_write_byte(command);
    SPI_write_byte(value);
    nrf_chip_disable();
}

```

```

__attribute__((always_inline))
uint8_t inline nrf_read_status()
{
    uint8_t command = 0xFF;
    uint8_t readByte = 0x00;
    nrf_chip_enable();
    SPI_write_byte(command);
    SPI_read_byte(&readByte);
    nrf_chip_disable();
    return readByte;
}

```

```
void nrf_write_config(uint8_t config)
{
    nrf_write_register(0x00, config & 0x7F);
}
```

```
uint8_t nrf_read_rf_ch()
{
    return nrf_read_register(0x06);
}
```

```
void nrf_read_tx_addr(uint8_t * address)
{
    uint8_t command = 0x10;
    SPI_write_byte(command);
    SPI_read_byte(address++);
    SPI_read_byte(address++);
    SPI_read_byte(address++);
    SPI_read_byte(address);
}
```

```
void nrf_write_tx_addr(uint8_t * tx_addr)
{
    /* the tx_addr array has the LSB first and the MSB last, assumed to be 4 bytes long */
    uint8_t command = 0x10;
    SPI_write_byte(command);
    SPI_write_byte(*tx_addr++);
    SPI_write_byte(*tx_addr++);
    SPI_write_byte(*tx_addr++);
    SPI_write_byte(*tx_addr);
}
```

```
uint8_t nrf_read_fifo_status()
{
    return nrf_read_register(0x17);
}
```

```
void nrf_flush_tx_fifo()
{
    SPI_write_byte(0xE1);
}
```

```
void nrf_flush_rx_fifo()
{
    SPI_write_byte(0xE2);
}
```

```
__attribute__((always_inline))
void inline nrf_chip_enable()
{
    PORTD_Clear(0);
}
```

```
__attribute__((always_inline))
void inline nrf_chip_disable()
{
    PORTD_Set(0);
}
```

```
__attribute__((always_inline))
void inline nrf_transmit_enable()
{
    uint8_t config_pwr_down = nrf_read_register(0x00) & 0xED;
    nrf_write_config(config_pwr_down);
}
```

```
__attribute__((always_inline))
void inline nrf_transmit_disable()
{
    uint8_t config_pwr_up = nrf_read_register(0x00) | 0x02;
    nrf_write_config(config_pwr_up);
}
/**
```

```
 * @file port.c
 * @brief implementation of port.h
 *
 * this file contains implementations of GPIO setup and accessing code
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 *
 */
```

```
#include "port.h"
#include "MKL25Z4.h"
#include "arch_arm32.h" /* for use of systick */
```

```
#define SysTick_Base_Ptr ((SysTick_Ptr)0xE000E010)
```

```
__attribute__((always_inline))
void inline InitSysTick()
{
    SysTick_Base_Ptr->RVR = 0x00FFFFFF; /* maximum value */
    SysTick_Base_Ptr->CSR = __SYSTICK_CLKSOURCE_MASK | __SYSTICK_ENABLE_MASK;
}
```

```
__attribute__((always_inline))
uint32_t inline gettime()
{
    return SysTick_Base_Ptr->CVR;
}
```

```
__attribute__((always_inline))
void inline GPIO_nrf_init()
```

```

{
/* TODO implement function */
}

__attribute__((always_inline))
void inline GPIO_Configure()
{
/* clock gating */
SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;
SIM_SCGC5 |= SIM_SCGC5_PORTD_MASK;

/* set to use GPIO mode */
PORTB_PCR18 |= PORT_PCR_MUX(1);
PORTB_PCR19 |= PORT_PCR_MUX(1);
PORTD_PCR1 |= PORT_PCR_MUX(1);
PORTD_PCR1 |= PORT_PCR_MUX(1);

/* set to outputs */
GPIOB_PDDR |= ((1 << RGB_RED_PIN) & (1 << RGB_GREEN_PIN));
GPIOD_PDDR |= (1 << RGB_BLUE_PIN);

/* set to logical low */
GPIOB_PCOR |= ((1 << RGB_RED_PIN) & (1 << RGB_GREEN_PIN));
GPIOD_PCOR |= (1 << RGB_BLUE_PIN);

}

__attribute__((always_inline))
void inline Toggle_Red_LED()
{
GPIOB_PTOR |= (1 << RGB_RED_PIN);
}

__attribute__((always_inline))
void inline PORTB_Set(uint8_t bit_num)
{
GPIOB_PSOR |= (1 << bit_num);
}

__attribute__((always_inline))
void inline PORTD_Set(uint8_t bit_num)
{
GPIOD_PSOR |= (1 << bit_num);
}

__attribute__((always_inline))
void inline PORTB_Clear(uint8_t bit_num)
{
GPIOB_PCOR |= (1 << bit_num);
}

__attribute__((always_inline))

```



```

void inline PORTD_Clear(uint8_t bit_num)
{
    GPIOD_PCOR |= (1 << bit_num);
}

__attribute__((always_inline))
void inline PORTB_Toggle(uint8_t bit_num)
{
    GPIOB_PTOR |= (1 << bit_num);
}

__attribute__((always_inline))
void inline PORTD_Toggle(uint8_t bit_num)
{
    GPIOD_PTOR |= (1 << bit_num);
}
/*****
 * 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 "debug.h"
#include "conversion.h"

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);
#ifdef VERBOSE
PRINTF(" Initial number: %d\n", num);
PRINTF(" Final Decimal number: %d\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);
    #ifdef VERBOSE
    PRINTF(" Initial Decimal number: %d\n", num);
    PRINTF(" Final Decimal number: %d\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;  
}
```

```
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_memmove(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_memmove2() {  
    uint8_t i;  
    int8_t ret = TEST_NO_ERROR;  
    uint8_t * set;  
    uint8_t * ptra;  
    uint8_t * ptrb;
```

```
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;  
}  
ptra = &set[0];
```

```
ptrb = &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;
uint8_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;
}
ptrb = &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 projec2.c
 * @brief implementation of projec2.h
 *
 * implements the data parsing code necessary for evaluating the UART
 * communication system for the KL25z
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 */
#include<stdint.h>
#include"project2.h"
#include"circbuf.h"
#include<stdio.h>
#ifdef KL25Z
#include"uart.h"
#endif

extern CB_t* recieve_buffer;
extern CB_t* transmit_buffer;

void project2()
{
#ifdef KL25Z
    UART_configure();/*TODO put in better UART setup control*/
#endif
#ifdef HOST
    if(CB_init(recieve_buffer, CIRCBUF_HOST_LENGTH)!=SUCCESS);
    CB_init(transmit_buffer, CIRCBUF_HOST_LENGTH);
#endif
    if(recieve_buffer==NULL)return;
    if(transmit_buffer==NULL)return;
    uint8_t data=0;
    CB_e retval=SUCCESS;
#ifdef HOST
    uint8_t char_holder=data;
    printf("Type a string to be processed, return to submit\n");
#endif
    while(1)
    {
        data=0;
        while( data!=ASCII_OFFSET_EOF && data!=EOF && data!=0xff)
        {
#ifdef HOST
            do
            {
                if(data==ASCII_OFFSET_EOF||data==EOF||data==0xff)break;
                if(char_holder==ASCII_OFFSET_EOF||char_holder==EOF||char_holder==0xff||data!=char_holder)break;
                char_holder = (uint8_t)getchar();
            } while(1);

```



```

    CB_buffer_add_item(recieve_buffer, char_holder);
}while(char_holder!='\n' && char_holder!='\r' && char_holder!=ASCII_OFFSET_EOF && char_holder!=EOF &&
char_holder!=0xff);
#endif
    retval=CB_buffer_remove_item(recieve_buffer, &data);
    if(retval==SUCCESS)
    {
        if(data>=ASCII_OFFSET_0 && data<=ASCII_OFFSET_9)
        {
            statistics.numeric++;
        }
        else if( (data>=ASCII_OFFSET_A && data<=ASCII_OFFSET_Z)
            |(data>=ASCII_OFFSET_LA && data<=ASCII_OFFSET_LZ))
        {
            statistics.alphabetic++;
        }
        else if( (data==(uint8_t)"\")(data==(uint8_t)['_])(data==(uint8_t)['_'])
            |(data==(uint8_t){'})(data==(uint8_t){'})|(data==(uint8_t)(')
            |(data==(uint8_t)')<')|(data==(uint8_t)')>')
            |(data==(uint8_t)':')|(data==(uint8_t)';')|(data==(uint8_t)'.)
            |(data==(uint8_t)!'')|(data==(uint8_t)')|(data==(uint8_t)')?')
            |(data==(uint8_t)"")|(data==(uint8_t)';')|(data==(uint8_t)'/'))
        {
            statistics.punctuation++;
        }
        else
        {
            statistics.miscellaneous++;
        }
    }
}
dump_statistics();
}
}

```

```

void dump_statistics()
{
    uint32_t i=0;

    uint8_t string1[] = "\r\nStatistics: "; /*14 characters*/
    uint8_t stringnewline[] = "\r\n";
    for(i=0; i<14; i++)
    {
        CB_buffer_add_item(transmit_buffer, *(string1+i));
    }
    for(i=0; i<3; i++)
    {
        CB_buffer_add_item(transmit_buffer, *(stringnewline+i));
    }

    uint8_t string2[] = "\tAlphabetic Characters: "; /*24 characters*/
    uint8_t string_alphabetic[32] = {};
}

```

```

uint32_t length = my_itoa(statistics.alphabetic,string_alphabetic,10);
for(i=0;i<24;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string2+i));
}
for(i=0;i<length;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string_alphabetic+i));
}
for(i=0;i<3;i++)
{
    CB_buffer_add_item(transmit_buffer,*(stringnewline+i));
}

```

```

uint8_t string3[] = "\tNumeric Characters: ";/*21 characters*/
uint8_t string_numeric[32] = {};
length = my_itoa(statistics.numeric,string_numeric,10);
for(i=0;i<24;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string3+i));
}
for(i=0;i<length;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string_numeric+i));
}
for(i=0;i<3;i++)
{
    CB_buffer_add_item(transmit_buffer,*(stringnewline+i));
}

```

```

uint8_t string4[] = "\tPunctuation Characters: ";/*25 characters*/
uint8_t string_punctuation[32] = {};
length = my_itoa(statistics.punctuation,string_punctuation,10);
for(i=0;i<25;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string4+i));
}
for(i=0;i<length;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string_punctuation+i));
}
for(i=0;i<3;i++)
{
    CB_buffer_add_item(transmit_buffer,*(stringnewline+i));
}

```

```

uint8_t string5[] = "\tMiscellaneous Characters: ";/*27 characters*/
uint8_t string_miscellaneous[32] = {};
length = my_itoa(statistics.miscellaneous,string_miscellaneous,10);
for(i=0;i<27;i++)
{
    CB_buffer_add_item(transmit_buffer,*(string5+i));
}

```

```

    }
    for(i=0;i<length;i++)
    {
        CB_buffer_add_item(transmit_buffer,*(string_miscellaneous+i));
    }
    for(i=0;i<3;i++)
    {
        CB_buffer_add_item(transmit_buffer,*(stringnewline+i));
    }

```

```

#ifdef KL25Z

```

```

    UART_start_buffered_transmission();

```

```

#endif

```

```

#ifdef HOST

```

```

    uint8_t char_to_print=0;

```

```

    while(CB_is_empty(transmit_buffer)!=EMPTY)
    {

```

```

        CB_buffer_remove_item(transmit_buffer, &char_to_print);

```

```

        printf("%c",char_to_print);

```

```

    }

```

```

#endif

```

```

}

```

```

/*

```

```

 * @file project3.c

```

```

 * @brief implements project3.h

```

```

 *

```

```

 * Project 3 functionality

```

```

 *

```

```

 * @author Seth Miers and Jake Cazden

```

```

 * @date March 15, 2018

```

```

 */

```

```

#define _POSIX_C_SOURCE 199309L

```

```

#include "project3.h"

```

```

#include "circbuf.h"

```

```

#include "conversion.h"

```

```

#include "memory.h"

```

```

#include <string.h>

```

```

#ifdef BBB

```

```

    #include "nordic.h"

```

```

    #include "spi.h"

```

```

    #include <time.h>

```

```

    #include <stdio.h>

```

```

#endif

```

```

#ifdef KL25Z

```

```

    #include "nordic.h"

```

```

    #include "spi.h"

```

```

    #include "port.h"

```

```

    #include "uart.h"

```

```

    #include "arch_arm32.h"

```

```

#endif

```

```

extern CB_t* recieve_buffer;
extern CB_t* transmit_buffer;

extern volatile uint32_t DMA_end_value;
extern volatile uint8_t dma_error_flag;
extern volatile uint32_t nooperation;

/* start of stack, end of stack */
extern int __StackTop, __StackLimit;

void project3()
{
#ifdef KL25Z
    GPIO_Configure();
    SPI_init();
    UART_configure();
#endif
#if defined(KL25Z) || defined(BBB)
#ifdef KL25Z
    stack_tracker_init();
#endif
    spi_setup_and_test();
    profiler();
#ifdef KL25Z
    stackusage();
#endif
#else
    nooperation++;
#endif
}

#ifdef KL25Z
void stack_tracker_init()
{
    uint8_t dummy_stack_var = 0xAA;
    uint8_t *start_of_unused_stack = &dummy_stack_var;
    uint8_t *stackend = (uint8_t *) (&__StackLimit);
    while(start_of_unused_stack >= stackend)
    {
        *start_of_unused_stack = 0xAA;
        start_of_unused_stack--;
    }
}

void stackusage()
{
    uint8_t *stackend = (uint8_t *) (&__StackLimit);
    uint8_t *stackbegin = (uint8_t *) (&__StackTop);
    uint8_t *stackused = stackend;
    uint8_t *my_string;
    uint8_t num_string[16];

```

```

uint8_t printsize = 0x00;
while(*stackused==0xAA)
{
    stackused++;
}
printsize = my_itoa((int32_t)(stackbegin-stackused), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " bytes of the stack used out of a ";
UART_send_n(my_string, 34);
printsize = my_itoa((int32_t)(stackbegin-stackend), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " total bytes (at max not current)\r\n";
UART_send_n(my_string, 35);
}

```

```

void profiler()

```

```

{
    /* TODO buildtime settings must be adjusted so that the heap can support these regions (at least 1kB) */
    /* TODO refactor this code to be smaller */
    /* TODO run with highest optimizations and lowest optimizations to see difference */

    /* areas to copy and move to */
    uint8_t *area_one = (uint8_t *) malloc(sizeof(uint8_t)*5020);
    uint8_t *area_two = area_one+20;
    uint8_t *retval = NULL;

    /* timer values */
    volatile uint32_t start_value = 0;
    volatile uint32_t end_value = 0;

    /* strings used for printing */
    uint8_t *my_string;
    uint8_t num_string[16];
    uint8_t printsize = 0x00;

    /******
    /* MEMSET PROFILER */
    /******

    /* standard library */
    /* 10 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the standard memset library with 10 bytes took ";
    UART_send_n(my_string, 57);
    start_value = SysTick_Base_Ptr->CVR;
    memset(area_one, '1', 10);
    end_value = SysTick_Base_Ptr->CVR;
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
}

```

```

UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the standard memset library with 100 bytes took ";
UART_send_n(my_string, 58);
start_value = SysTick_Base_Ptr->CVR;
memset(area_one, '2', 100);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the standard memset library with 1000 bytes took ";
UART_send_n(my_string, 59);
start_value = SysTick_Base_Ptr->CVR;
memset(area_one, '3', 1000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the standard memset library with 5000 bytes took ";
UART_send_n(my_string, 59);
start_value = SysTick_Base_Ptr->CVR;
memset(area_one, '4', 5000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* my library */
/* 10 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the custom memset function with 10 bytes took ";
UART_send_n(my_string, 56);
start_value = SysTick_Base_Ptr->CVR;
my_memset(area_one, 10, '1');
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);

```

```

    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* 100 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memset function with 100 bytes took ";
    UART_send_n(my_string, 57);
    start_value = SysTick_Base_Ptr->CVR;
    my_memset(area_one, 100, '2');
    end_value = SysTick_Base_Ptr->CVR;
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* 1000 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memset function with 1000 bytes took ";
    UART_send_n(my_string, 58);
    start_value = SysTick_Base_Ptr->CVR;
    my_memset(area_one, 1000, '3');
    end_value = SysTick_Base_Ptr->CVR;
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* 5000 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memset function with 5000 bytes took ";
    UART_send_n(my_string, 58);
    start_value = SysTick_Base_Ptr->CVR;
    my_memset(area_one, 5000, '4');
    end_value = SysTick_Base_Ptr->CVR;
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* TODO the entire dma profiler */
    /* dma library */
    /* 10 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the DMA memset function with 10 bytes took ";
    UART_send_n(my_string, 53);
    start_value = SysTick_Base_Ptr->CVR;
    retval = memset_dma(area_one, 10, '1', 1);
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);

```

```

if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memset function with 100 bytes took ";
UART_send_n(my_string, 54);
start_value = SysTick_Base_Ptr->CVR;
retval = memset_dma(area_one, 100, '2', 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memset function with 1000 bytes took ";
UART_send_n(my_string, 55);
start_value = SysTick_Base_Ptr->CVR;
retval = memset_dma(area_one, 1000, '3', 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memset function with 5000 bytes took ";
UART_send_n(my_string, 55);

```



```

start_value = SysTick_Base_Ptr->CVR;
retval = memset_dma(area_one, 5000, '4', 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* dma library - 4 byte transfer */
/* 10 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memset function with 10 bytes and 4 byte transfer took ";
UART_send_n(my_string, 73);
start_value = SysTick_Base_Ptr->CVR;
retval = memset_dma(area_one, 10, '1', 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memset function with 100 bytes and 4 byte transfer took ";
UART_send_n(my_string, 74);
start_value = SysTick_Base_Ptr->CVR;
retval = memset_dma(area_one, 100, '2', 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);

```

```

    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* 1000 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the DMA memset function with 1000 bytes and 4 byte transfer took ";
    UART_send_n(my_string, 75);
    start_value = SysTick_Base_Ptr->CVR;
    retval = memset_dma(area_one, 1000, '3', 4);
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
    if(dma_error_flag)
    {
        dma_error_flag=0;
        num_string[0] = "-";
        num_string[1] = "1";
        printsize = 2;
    }
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);
    /* 5000 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the DMA memset function with 5000 bytes and 4 byte transfer took ";
    UART_send_n(my_string, 75);
    start_value = SysTick_Base_Ptr->CVR;
    retval = memset_dma(area_one, 5000, '4', 4);
    SysTick_Base_Ptr->CVR = 0;
    SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
    printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
    if(dma_error_flag)
    {
        dma_error_flag=0;
        num_string[0] = "-";
        num_string[1] = "1";
        printsize = 2;
    }
    UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
    UART_send_n(my_string, 22);

    /***/
    /* MEMMOVE PROFILER */
    /***/
    /* standard library */
    /* 10 bytes */
    SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the standard memmove library with 10 bytes took ";
    UART_send_n(my_string, 58);
    start_value = SysTick_Base_Ptr->CVR;
    memmove(area_one, area_two, 10);
    end_value = SysTick_Base_Ptr->CVR;

```

```

SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the standard memmove library with 100 bytes took ";
UART_send_n(my_string, 59);
start_value = SysTick_Base_Ptr->CVR;
memmove(area_one, area_two, 100);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the standard memmove library with 1000 bytes took ";
UART_send_n(my_string, 60);
start_value = SysTick_Base_Ptr->CVR;
memmove(area_one, area_two, 1000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the standard memmove library with 5000 bytes took ";
UART_send_n(my_string, 60);
start_value = SysTick_Base_Ptr->CVR;
memmove(area_one, area_two, 5000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* my library */
/* 10 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memmove function with 10 bytes took ";
UART_send_n(my_string, 57);
start_value = SysTick_Base_Ptr->CVR;
my_memmove(area_one, area_two, 10);

```

```

end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memmove function with 100 bytes took ";
UART_send_n(my_string, 58);
start_value = SysTick_Base_Ptr->CVR;
my_memmove(area_one, area_two, 100);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memmove function with 1000 bytes took ";
UART_send_n(my_string, 59);
start_value = SysTick_Base_Ptr->CVR;
my_memmove(area_one, area_two, 1000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the custom memmovefunction with 5000 bytes took ";
UART_send_n(my_string, 59);
start_value = SysTick_Base_Ptr->CVR;
my_memmove(area_one, area_two, 5000);
end_value = SysTick_Base_Ptr->CVR;
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-end_value), num_string, 10);
UART_send_n(num_string, printsize);
    my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* dma library */
/* 10 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
    my_string = (unsigned char *) "Profiling the DMA memmove function with 10 bytes took ";
UART_send_n(my_string, 54);
start_value = SysTick_Base_Ptr->CVR;

```

```

retval = memmove_dma(area_one, area_two, 10, 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 100 bytes took ";
UART_send_n(my_string, 55);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 100, 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 1000 bytes took ";
UART_send_n(my_string, 56);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 1000, 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);

```

```

/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 5000 bytes took ";
UART_send_n(my_string, 56);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 5000, 1);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* dma library - 4 byte transfer */
/* 10 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 10 bytes and 4 byte transfer took
";
UART_send_n(my_string, 74);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 10, 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 100 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 100 bytes and 4 byte transfer took
";
UART_send_n(my_string, 75);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 100, 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{

```

```

    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 1000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 1000 bytes and 4 byte transfer
took ";
UART_send_n(my_string, 76);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 1000, 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
/* 5000 bytes */
SysTick_Base_Ptr->CSR |= __SYSTICK_ENABLE_MASK; /* enable counting */
my_string = (unsigned char *) "Profiling the DMA memmove function with 5000 bytes and 4 byte transfer
took ";
UART_send_n(my_string, 76);
start_value = SysTick_Base_Ptr->CVR;
retval = memmove_dma(area_one, area_two, 5000, 4);
SysTick_Base_Ptr->CVR = 0;
SysTick_Base_Ptr->CSR &= ~(__SYSTICK_ENABLE_MASK); /* disable counting */
printsize = my_itoa((int32_t)(start_value-DMA_end_value), num_string, 10);
if(dma_error_flag)
{
    dma_error_flag=0;
    num_string[0] = "-";
    num_string[1] = "1";
    printsize = 2;
}
UART_send_n(num_string, printsize);
my_string = (unsigned char *) " clock cycles to run\r\n";
UART_send_n(my_string, 22);
}
#endif
#ifdef BBB
void profiler()

```

```

{
/* TODO buildtime settings must be adjusted so that the heap can support these regions (at least 1kB) */
/* TODO refactor this code to be smaller */
/* TODO run with highest optimizations and lowest optimizations to see difference */

/* areas to copy and move to */
uint8_t *area_one = (uint8_t *) malloc(sizeof(uint8_t)*5020);
uint8_t *area_two = area_one+20;

/* timer values */
struct timespec start_value;
struct timespec end_value;

/*****/
/* MEMSET PROFILER */
/*****/

/* standard library */
/* 10 bytes */
    printf("Profiling the standard memset library with 10 bytes took ");
    clock_gettime(CLOCK_MONOTONIC,&start_value);
    memset(area_one, '1', 10);
    clock_gettime(CLOCK_MONOTONIC,&end_value);
    printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 100 bytes */
    printf("Profiling the standard memset library with 100 bytes took ");
    clock_gettime(CLOCK_MONOTONIC,&start_value);
    memset(area_one, '2', 100);
    clock_gettime(CLOCK_MONOTONIC,&end_value);
    printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 1000 bytes */
    printf("Profiling the standard memset library with 1000 bytes took ");
    clock_gettime(CLOCK_MONOTONIC,&start_value);
    memset(area_one, '3', 1000);
    clock_gettime(CLOCK_MONOTONIC,&end_value);
    printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 5000 bytes */
    printf("Profiling the standard memset library with 5000 bytes took ");
    clock_gettime(CLOCK_MONOTONIC,&start_value);
    memset(area_one, '4', 5000);
    clock_gettime(CLOCK_MONOTONIC,&end_value);
    printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* my library */
/* 10 bytes */
    printf("Profiling the custom memset function with 10 bytes took ");
    clock_gettime(CLOCK_MONOTONIC,&start_value);
    my_memset(area_one, 10, '1');
    clock_gettime(CLOCK_MONOTONIC,&end_value);

```



```

printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 100 bytes */
printf("Profiling the custom memset function with 100 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
my_memset(area_one, 100, '2');
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 1000 bytes */
printf("Profiling the custom memset function with 1000 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
my_memset(area_one, 1000, '3');
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 5000 bytes */
printf("Profiling the custom memset function with 5000 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
my_memset(area_one, 5000, '4');
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");

/*****
/* MEMMOVE PROFILER */
*****/
/* standard library */
/* 10 bytes */
printf("Profiling the standard memmove library with 10 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
memmove(area_one, area_two, 10);
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 100 bytes */
printf("Profiling the standard memmove library with 100 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
memmove(area_one, area_two, 100);
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 1000 bytes */
printf("Profiling the standard memmove library with 1000 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);
memmove(area_one, area_two, 1000);
clock_gettime(CLOCK_MONOTONIC, &end_value);
printf("%ld", (end_value.tv_nsec - start_value.tv_nsec) * 3 / 10);
printf(" clock cycles to run\r\n");
/* 5000 bytes */
printf("Profiling the standard memmove library with 5000 bytes took ");
clock_gettime(CLOCK_MONOTONIC, &start_value);

```

```

memmove(area_one, area_two, 5000);
clock_gettime(CLOCK_MONOTONIC,&end_value);
printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* my library */
/* 10 bytes */
    printf("Profiling the custom memmove function with 10 bytes took ");
clock_gettime(CLOCK_MONOTONIC,&start_value);
my_memmove(area_one, area_two, 10);
clock_gettime(CLOCK_MONOTONIC,&end_value);
printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 100 bytes */
    printf("Profiling the custom memmove function with 100 bytes took ");
clock_gettime(CLOCK_MONOTONIC,&start_value);
my_memmove(area_one, area_two, 100);
clock_gettime(CLOCK_MONOTONIC,&end_value);
printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 1000 bytes */
    printf("Profiling the custom memmove function with 1000 bytes took ");
clock_gettime(CLOCK_MONOTONIC,&start_value);
my_memmove(area_one, area_two, 1000);
clock_gettime(CLOCK_MONOTONIC,&end_value);
printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
/* 5000 bytes */
    printf("Profiling the custom memmove function with 5000 bytes took ");
clock_gettime(CLOCK_MONOTONIC,&start_value);
my_memmove(area_one, area_two, 5000);
clock_gettime(CLOCK_MONOTONIC,&end_value);
printf("%ld", (end_value.tv_nsec-start_value.tv_nsec)*3/10);
    printf(" clock cycles to run\r\n");
}
#endif

```

```

#ifdef KL25Z

```

```

void spi_setup_and_test()

```

```

{
    uint8_t my_string[] = "Starting project 3...";
    uint8_t return_string[] = "\r\n";
    uint8_t hex_string[] = "Status Reg is: 0x";
    UART_send_n(my_string, sizeof(my_string));
    UART_send_n(return_string, sizeof(return_string));
    uint8_t j = 0x00;
    uint8_t statreg = 0x00;
    uint8_t printsize = 0x00;
    nrf_write_register(0x05, j);
    statreg = nrf_read_register(0x05);
    printsize = my_itoa((int32_t)statreg, my_string, 16);
    UART_send_n(hex_string, sizeof(hex_string));
    UART_send_n(my_string, printsize);
}

```

```

UART_send_n(return_string, sizeof(return_string));
}
#endif
#ifdef BBB
void spi_setup_and_test()
{
    printf("Starting project 3...\r\n");
    /*uint8_t j = 0x00;*/
    uint8_t statreg = 0x00;
    /*nrf_write_register(0x05, j);
    statreg = nrf_read_register(0x05);*/
    printf("Status Reg is: 0x");
    printf("%d\r\n", (int)statreg);
}
#endif
/*
 * @file spi.c
 * @brief implementation of spi.h
 *
 * spi read, write, send, and flush
 *
 * @author Seth Miers and Jake Cazden
 * @date March 15, 2018
 */

#include "spi.h"
#ifdef KL25Z
#include "MKL25Z4.h"
#endif
#ifdef BBB
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#endif

/* TODO move chip enable to function in spi.c */

void SPI_init()
{
#ifdef KL25Z
    /* TODO implement function */

    /* Enable the clock for the SPI */
    SIM_SCGC5 |= SIM_SCGC5_PORTD_MASK;
    SIM_SCGC4 |= SIM_SCGC4_SPI0_MASK;

    /* configure output pins */
    PORTD_PCR0 = PORT_PCR_MUX(1); /*CE*/
    GPIOD_PDDR |= (1 << 0); /* set to output */
    GPIOD_PSOR |= (1 << 0); /* set high (off) */

```

```

PORTD_PCR1 = PORT_PCR_MUX(2); /*SCLK*/
PORTD_PCR2 = PORT_PCR_MUX(2); /*MOSI*/
PORTD_PCR3 = PORT_PCR_MUX(2); /*MISO*/
/*PORTD_PCR5 = PORT_PCR_MUX(1); CSN*/
/*GPIOD_PDDR |= (1 << 5); set to output */

/* set as master device: 0x10 */
SPI0_C1 = 0x10;
/* auto control CE */
SPI0_C2 = 0x00;
/* set the baud rate divisor to 64 */
SPI0_BR = 0x0A;
/* enable spi */
SPI0_C1 |= 0x40;
#endif
#ifdef BBB
/* TODO implement function */
#endif
}

void SPI_read_byte(uint8_t * byte)
{
#ifdef KL25Z
/* wait until empty*/
while((SPI0_S & SPI_S_SPTEF_MASK) != SPI_S_SPTEF_MASK);
SPI0_D = 0x00;
/* wait until byte exists*/
while((SPI0_S & SPI_S_SPRF_MASK) != SPI_S_SPRF_MASK);
*byte = SPI0_D;
#endif
#ifdef BBB
/* TODO implement function */
#endif
}

void SPI_write_byte(uint8_t byte)
{
#ifdef KL25Z
/* wait until empty*/
while((SPI0_S & SPI_S_SPTEF_MASK) != SPI_S_SPTEF_MASK);
SPI0_D = byte;
/* wait until byte exists in order to block*/
while((SPI0_S & SPI_S_SPRF_MASK) != SPI_S_SPRF_MASK);
/* discard byte */
byte = SPI0_D;
#endif
#ifdef BBB
/* TODO implement function */
#endif
}

void SPI_send_packet(uint8_t * p, size_t length)

```

```

{
#ifdef KL25Z
    uint8_t i;
    for(i = 0; i < length; i++)
        SPI_write_byte(p[i]);
#endif
#ifdef BBB
    /* TODO implement function */
#endif
}

void SPI_flush()
{
#ifdef KL25Z
    /* TODO implement function */
#endif
#ifdef BBB
    /* TODO implement function */
#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:            b150220
**
** 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 (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 "MKL25Z4.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) */
#ifdef 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 */
#ifdef SYSTEM_SMC_PMPROT_VALUE
    SMC->PMPROT = SYSTEM_SMC_PMPROT_VALUE;
#endif

```

```

/* System clock initialization */
/* Internal reference clock trim initialization */
#if defined(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) */
#if defined(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
#endif /* 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_PLLCLKEN0_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
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_EREFS0_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_EREFS0_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 */
    }
    #if (MCG_MODE == MCG_MODE_PEE)
        MCG->C1 &= (uint8_t)(~(MCG_C1_CLKS_MASK));
    #endif
#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 LPTMR */
    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_PBYN_MASK); /* Clock source: LPO, Prescaler
bypass enable */
    LPTMR0->CSR = LPTMR_CSR_TEN_MASK; /* LPTMR enable */
    while((LPTMR0_CSR & LPTMR_CSR_TCF_MASK) == 0u) {
    }
    LPTMR0_CSR = 0x00; /* Disable LPTMR */
    SIM_SCGC5 &= (uint32_t)(~(uint32_t)SIM_SCGC5_LPTMR_MASK);
#elif ((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 */
    }
#elif ((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 */
    }
#elif (MCG_MODE == MCG_MODE_PEE)
    while((MCG->S & MCG_S_CLKST_MASK) != 0x0CU) { /* Wait until output of the PLL is selected */
    }
}

```

```

#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 uart.c
 * @brief implementation of uart.h
 *
 * this file implements uart.h, implementing interrupt functions for the
 * UART communication system
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 */
#include "uart.h"

#include "cirbuf.h" /* so we can use the circular buffer in our uart */

#ifdef KL25Z
#include "MKL25Z4.h"
#endif

extern CB_t* recieve_buffer;
extern CB_t* transmit_buffer;

UART_e UART_configure()
{
    /* set PTE0/PTE20/PTA2/PTA14/PTB17/PTD7 to UART1_TX (i think it should be UART0)*/
    /* set PTE1/PTE21/PTA1/PTA15/PTB16/PTD6 to UART1_RX */

    /*turn on PORTA clocking with SIM_SCGC5 bit 9 (PORTA) (this may be active already)
    * thereby allowing pin manipulation*/
    SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

    /*set PTA1 to ALT2, PTA2 to ALT2 using PORTA_PCR1, PORTA_PCR2
    * PORTA_PCR1[IRQC] = 0000; - no innterrupt from the pins directly
    * PORTA_PCR2[IRQC] = 0000; - no innterrupt from the pins directly
    * PORTA_PCR1[MUX] = 010; - ALT 2
    * PORTA_PCR2[MUX] = 010; - ALT 2
    * thereby clearing the lines for UART0*/
    PORTA_PCR1 = PORT_PCR_ISF(CLEAR_PCR_ISF)
        |PORT_PCR_IRQC(DISABLE_PCR_IRQC)
        |PORT_PCR_MUX(PCR_MUX_ALT2);

    PORTA_PCR2 = PORT_PCR_ISF(CLEAR_PCR_ISF)
        |PORT_PCR_IRQC(DISABLE_PCR_IRQC)
        |PORT_PCR_MUX(PCR_MUX_ALT2);

    /*set UART0 clock source with SIM_SOPT2
    * pins 27-26 (UART0SRC) = 01 for FLL select
    * pin 16 (PLL_FLLSEL) = 0 for MCG_FLLCLK (no division by 2)*/
    SIM_SOPT2 &= ~SIM_SOPT2_UART0SRC(SIM_SOPT2_UART0SRC_CLEAR);/*clear UART0SRC bits*/

```

```

SIM_SOPT2 |= SIM_SOPT2_UART0SRC(SIM_SOPT2_UART0SRC_MCGIRCLK);/*set UART0SRC
bits*/

/*SIM_SOPT2 &= ~SIM_SOPT2_PLLFLLSEL(SIM_SOPT2_PLLFLLSEL_CLEAR); clear PLLFLLSEL*/
/*SIM_SOPT2 |= SIM_SOPT2_PLLFLLSEL(SIM_SOPT2_PLLFLLSEL_FLLSRC); set PLLFLLSEL to 0*/
MCG_C1 |= MCG_C1_IRCLKEN(MCG_C1_IRCLKEN_ENABLED);
MCG_C2 |= MCG_C2_IRCS(MCG_C2_IRCS_FAST);
MCG_SC &= ~MCG_SC_FCRDIV(MCG_C2_FCRDIV_CLEAR);
MCG_SC |= MCG_SC_FCRDIV(MCG_C2_FCRDIV_NODIVISION);
/*choose UART0 tx and rx source with SIM_SOPT5
 * pin 2 = 0 for UART0_RX pin (UART0RXSRC)
 * pin 1-0 = 00 for UART_TX pin (UART0TXSRC)*/
SIM_SOPT5 &= ~SIM_SOPT5_UART0RXSRC(SIM_SOPT5_UART0RXSRC_CLEAR);/*clear
UART0RXSRC bits*/
SIM_SOPT5 |= SIM_SOPT5_UART0RXSRC(SIM_SOPT5_UART0RXSRC_RXPIN);/*set
UART0RXSRC bits to 0*/

SIM_SOPT5 &= ~SIM_SOPT5_UART0TXSRC(SIM_SOPT5_UART0TXSRC_CLEAR);/*clear
UART0TXSRC*/
SIM_SOPT5 |= SIM_SOPT5_UART0TXSRC(SIM_SOPT5_UART0TXSRC_TXPIN);/*set UART0TXSRC
to 0*/

/*enable clock to UART0 with SIM_SCGC4
 * pin 10 (UART0) =1 to activate clock gate*/
SIM_SCGC4 |= SIM_SCGC4_UART0_MASK;/*set UART0 to receive clocking*/

UART0_C4 |= UART0_C4_OSR(UART0_C4_OSR_SAMPLERATE);

/*set uart0 baud rate high register
 * LBKDIE = 0, we arent using this interrupt
 * RXEDGIE = 0, we arent using this interrupt
 * SBNS = 0, 1 stop bit desired
 * SBR = (5 highest bits of baud rate)*/
UART0_BDH = UART0_BDH_LBKDIE(UART0_BDH_LBKDIE_DISABLE)
|UART0_BDH_RXEDGIE(UART0_BDH_RXEDGIE_DISABLE)
|UART0_BDH_SBNS(UART0_BDH_SBNS_SINGLSTOPBIT)
|UART0_BDH_SBR((CALCULATED_BAUD_MASK &
SBR_HIGHMASK)>>UART0_BDL_SBR_WIDTH);

/*set uart0 baud rate low register
 * SBR = (8 lower bits of baud rate)*/
UART0_BDL = UART0_BDL_SBR(CALCULATED_BAUD_MASK&SBR_LOWMASK);

/*set uart0 control 1 register
 * LOOPS = 0 - no looback
 * DOZEEN = 0 - UART enabled in wait mode (doesnt really matter)
 * RSRC = 0 - meaningless when LOOPS=0
 * M = 0 - use 8 bit transmission
 * WAKE = 0 - meaningless because RWU is 0
 * ILT = 1 - idle character count starts after stopbit
 * PE = 0 - parity disabled
 * PT = 0 - even parity (meaningless)

```

```

* */
UART0_C1 = UART0_C1_LOOPS(UART0_C1_LOOPS_NORMALOPERATION)
|UART0_C1_DOZEEN(UART0_C1_DOZEEN_ENABLED)
|UART0_C1_RSRC(UART0_C1_RSRC_DEFAULT)
|UART0_C1_M(UART0_C1_M_8BIT)
|UART0_C1_WAKE(UART0_C1_WAKE_DEFAULT)
|UART0_C1_ILT(UART0_C1_ILT_AFTERSTOP)
|UART0_C1_PE(UART0_C1_PE_NOPARITY)
|UART0_C1_PT(UART0_C1_PT_DEFAULTPARTIY);

/*set uart0 control 2 register
* TIE = 0, using TCIE interrupt for transmit complete
* TCIE = 1, transmit complete interupt *is* enabled (when Transmit Complete=1)
* RIE = 1, reciever interrupt *is* enabled (when Recieve data register full=1)
* ILIE = 0, dont care about idle line interrupt
* TE = 0, transmitter disabled during setup
* RE = 0, reciever disabled during setup
* RWU = 0, normal operation
* SBK = 0, dont send break character
* */
UART0_C2 = UART0_C2_TIE(UART0_C2_TIE_DISABLED)
|UART0_C2_TCIE(UART0_C2_TCIE_DISABLED)
|UART0_C2_RIE(UART0_C2_RIE_ENABLED)
|UART0_C2_ILIE(UART0_C2_TLIE_DISABLED)
|UART0_C2_TE(UART0_C2_TE_DISABLED)
|UART0_C2_RE(UART0_C2_RE_DISABLED)
|UART0_C2_RWU(UART0_C2_RWU_NOWAKEUP)
|UART0_C2_SBK(UART0_C2_SBK_NOBREAK);

/*register uart0_irqhandler with NVIC*/
/*should be done by having the samem symbol name UART0_IRQHandler*/

/*enable specific UART0_IRQ in NVIC
* NVIC_ISER (interrupt set enable register)
* or NVIC_EnableIRQ(IRQn_Type IRQn)*/
NVIC_ClearPendingIRQ(UART0_IRQn);
NVIC_EnableIRQ(UART0_IRQn);

/*enable general NVIC interrupts
* __enable_irq
* cpsie/cpsid lowest bit, i think*/
__enable_irq();

/* initialize the buffers */
CB_e bufferinitreturn1 = CB_init(recieve_buffer, BUFFER_LENGTH);
CB_e bufferinitreturn2 = CB_init(transmit_buffer, BUFFER_LENGTH);

/*turn on UART0 transmit and recieve
* UART0_C2[TE]=1
* UART0_C2[RE]=1;
* */

```

```

UART0_C2 |= UART0_C2_TE(UART0_C2_TE_ENABLED)/*this will be turned on as needed*/
|UART0_C2_RE(UART0_C2_RE_ENABLED);

if(bufferinitreturn1 != SUCCESS || bufferinitreturn2 !=SUCCESS)
{
    return UART_FAILURE;
}

return UART_SUCCESS;
}

UART_e UART_send(uint8_t *data)
{
    if(data==NULL)
    {
        return UART_FAILURE;
    }
    uint8_t transmitenabledflag;
    uint8_t transmitinterruptenabledflag;
    if(UART0_C2 & UART0_C2_TCIE_MASK)/*check if transmit interrupt is enabled*/
    {
        transmitinterruptenabledflag=1;/*save initial state*/
        UART0_C2 &= ~UART0_C2_TCIE(UART0_C2_TCIE_ENABLED);/*disable if not off*/
    }
    else
    {
        transmitinterruptenabledflag=0;/*save initial state*/
    }
    if(UART0_C2 & UART0_C2_TE_MASK)/*check if transmit is enabled*/
    {
        transmitenabledflag=1;/*save initial state*/
    }
    else
    {
        transmitenabledflag=0;/*save initial state*/
        UART0_C2 |= UART0_C2_TE(UART0_C2_TE_ENABLED);/*enable if off*/
    }

    UART0_D = *data;/*push data into UART0_D register*/
    while(((UART0_S1 &
UART0_S1_TC_MASK)>>UART0_S1_TC_SHIFT)==UART0_S1_TC_ACTIVE);/*block on transmit*/

    if(!transmitenabledflag)
    { /*restore transmit state*/
        UART0_C2 &= ~UART0_C2_TE(UART0_C2_TE_ENABLED);
    }
    if(transmitinterruptenabledflag)
    { /*restore transmit interrupt state*/
        UART0_C2 |= UART0_C2_TCIE(UART0_C2_TCIE_ENABLED);
    }
    return UART_SUCCESS;
}

```



```

UART_e UART_send_n(uint8_t *data, size_t num_bytes)
{
    if(data==NULL)
    {
        return UART_FAILURE;
    }
    if(num_bytes==0)
    {
        return UART_SUCCESS;
    }
    uint8_t transmitenabledflag;
    uint8_t transmitinterruptenabledflag;
    if(UART0_C2 & UART0_C2_TCIE_MASK)/*check if transmit interrupt is enabled*/
    {
        transmitinterruptenabledflag=1;/*save initial state*/
        UART0_C2 &= ~UART0_C2_TCIE(UART0_C2_TCIE_ENABLED);/*disable if not off*/
    }
    else
    {
        transmitinterruptenabledflag=0;/*save initial state*/
    }
    if(UART0_C2 & UART0_C2_TE_MASK)/*check if transmit is enabled*/
    {
        transmitenabledflag=1;/*save initial state*/
    }
    else
    {
        transmitenabledflag=0;/*save initial state*/
        UART0_C2 |= UART0_C2_TE(UART0_C2_TE_ENABLED);/*enable if off*/
    }

    size_t i = 0;
    for(i=0;i<num_bytes;i++)
    {/*block here and transmit all the data using UART_send*/
        UART0_D = *(data+i);/*push data into UART0_D register*/
        while(((UART0_S1 &
UART0_S1_TC_MASK)>>UART0_S1_TC_SHIFT)==UART0_S1_TC_ACTIVE);/*block on transmit*/
    }

    if(!transmitenabledflag)
    { /*restore transmit state*/
        UART0_C2 &= ~UART0_C2_TE(UART0_C2_TE_ENABLED);
    }
    if(transmitinterruptenabledflag)
    { /*restore transmit interrupt state*/
        UART0_C2 |= UART0_C2_TCIE(UART0_C2_TCIE_ENABLED);
    }
    return UART_SUCCESS;
}

UART_e UART_recieve(uint8_t *data)

```

```

{
    if(data==NULL)
    {
        return UART_FAILURE;
    }
    uint8_t recieveenabledflag;
    uint8_t recieveinterruptenabledflag;
    if(UART0_C2 & UART0_C2_RIE_MASK)/*check if reciever interrupt is enabled*/
    {
        recieveinterruptenabledflag=1;/*save initial state*/
        UART0_C2 &= ~UART0_C2_RIE(UART0_C2_RIE_ENABLED);/*disable if not off*/
    }
    else
    {
        recieveinterruptenabledflag=0;/*save initial state*/
    }
    if(UART0_C2 & UART0_C2_RE_MASK)/*check if recieve is enabled*/
    {
        recieveenabledflag=1;/*save initial state*/
    }
    else
    {
        recieveenabledflag=0;/*save initial state*/
        UART0_C2 |= UART0_C2_RE(UART0_C2_RE_ENABLED);/*enable if off*/
    }

    while(((UART0_S1 &
UART0_S1_RDRF_MASK)>>UART0_S1_RDRF_SHIFT)==UART0_S1_RDRF_EMPTY);/*block on
recieve*/
    *data = UART0_D;/*read from UART0_D into data*/

    if(!recieveenabledflag)
    { /*restore recieve state to disabled if necessary*/
        UART0_C2 &= ~UART0_C2_RE(UART0_C2_RE_ENABLED);
    }
    if(recieveinterruptenabledflag)
    { /*restore recieve interrupt state to enabled if necessary*/
        UART0_C2 |= UART0_C2_RIE(UART0_C2_RIE_ENABLED);
    }
    return UART_SUCCESS;
}

UART_e UART_recieve_n(uint8_t *data, size_t num_bytes)
{
    if(data==NULL)
    {
        return UART_FAILURE;
    }
    if(num_bytes==0)
    {
        return UART_SUCCESS;
    }
}

```

```

uint8_t recieveenabledflag;
uint8_t recieveinterruptenabledflag;
if(UART0_C2 & UART0_C2_RIE_MASK)/*check if reciever interrupt is enabled*/
{
    recieveinterruptenabledflag=1;/*save initial state*/
    UART0_C2 &= ~UART0_C2_RIE(UART0_C2_RIE_ENABLED);/*disable if not off*/
}
else
{
    recieveinterruptenabledflag=0;/*save initial state*/
}
if(UART0_C2 & UART0_C2_RE_MASK)/*check if recieve is enabled*/
{
    recieveenabledflag=1;/*save initial state*/
}
else
{
    recieveenabledflag=0;/*save initial state*/
    UART0_C2 |= UART0_C2_RE(UART0_C2_RE_ENABLED);/*enable if off*/
}

size_t i = 0;
for(i=0;i<num_bytes;i++){
    while(((UART0_S1 &
UART0_S1_RDRF_MASK)>>UART0_S1_RDRF_SHIFT)==UART0_S1_RDRF_EMPTY);/*block on
recieve*/
    *(data+i) = UART0_D;/*read from UART0_D into data*/
}

if(!recieveenabledflag)
{ /*restore recieve state to disabled if necessary*/
    UART0_C2 &= ~UART0_C2_RE(UART0_C2_RE_ENABLED);
}
if(recieveinterruptenabledflag)
{ /*restore recieve interrupt state to enabled if necessary*/
    UART0_C2 |= UART0_C2_RIE(UART0_C2_RIE_ENABLED);
}
return UART_SUCCESS;
}

void UART0_IRQHandler()
{
    if(((UART0_S1 & UART0_S1_RDRF_MASK)>>UART0_S1_RDRF_SHIFT)==UART0_S1_RDRF_FULL)
    {
        volatile uint8_t sink = UART0_D;
        if(((UART0_S1 & UART0_S1_TC_MASK)>>UART0_S1_TC_SHIFT)==UART0_S1_TC_IDLE)
        {
            UART0_D = sink;
        }
    }
    else if(transmit_buffer!=NULL)
    {

```

```

    CB_buffer_add_item(transmit_buffer,sink);
    UART0_C2 |= (UART0_C2_TIE(UART0_C2_TIE_ENABLED));
}

if(recieve_buffer!=NULL)
{
    /*discard the data to clear the flag*/
    CB_buffer_add_item(recieve_buffer,sink);
}
sink=0;
}

if(((UART0_S1 &
UART0_S1_TDRE_MASK)>>UART0_S1_TDRE_SHIFT)==UART0_S1_TDRE_EMPTY)
{
    CB_e ret = EMPTY;
    uint8_t temp;
    if(transmit_buffer!=NULL)
    {
        ret = CB_buffer_remove_item(transmit_buffer, &temp);
        UART0_D = temp;
    }
    if(ret==EMPTY)
    {
        UART0_C2 &= ~(UART0_C2_TIE(UART0_C2_TIE_ENABLED));
    }
}
NVIC_ClearPendingIRQ(UART0_IRQn);
}

UART_e UART_start_buffered_transmission()
{
    UART0_C2 |= UART0_C2_TE(UART0_C2_TE_ENABLED);/*begin transmission*/
    UART0_C2 |= UART0_C2_TIE(UART0_C2_TIE_ENABLED); /*enable transmission interrupt*/
    return UART_SUCCESS;
}

/**
 * @file unittest.c
 * status = CB_buffer_remove_item(my_cbuf, &data);
 * @brief implementation of unittest.h
 *
 * this file implements unittest.h,
 * testing an array of different functinos used
 * in this project
 *
 * @author Seth Miers and Jake Cazden
 * @date March 04, 2018
 *
 */
#include "cmocka.h"
#include "memory.h"
#include "conversion.h"
#include "data.h"
#include "circular.h"

```

```
#define TEST_LENGTH (256)
```

```
void memory_test(void **state)
{
    size_t length = TEST_LENGTH;
    uint8_t * src = (uint8_t *)reserve_words(length);
    uint8_t * dst = (uint8_t *)reserve_words(length);
    uint8_t * status;

    /* send null src pointer and assert that null is returned
     * and that src and dst are still null for a memmove*/
    status = my_memmove((void*)NULL, dst, length);
    assert_null(status);

    /* send null dst pointer and assert that null is returned
     * and that src and dst are still null for a memmove*/
    status = my_memmove(src, (void*)NULL, length);
    assert_null(status);

    /* send null pointers and assert that null is returned
     * and that src and dst are still null for a memmove*/
    status = my_memmove((void*)NULL, (void*)NULL, length);
    assert_null(status);

    /* with src and dst in different places assert that
     * the arrays are equal after a memmove */
    for(uint16_t i = 0; i < (int)length; i++)
    {
        *(src+i) = i;
        *(dst+i) = (length - 1) - i;
    }
    status = my_memmove(src, dst, length);
    assert_non_null(status);
    for(uint16_t i = 0; i < length; i++)
    {
        assert_int_equal(*(src+i), *(dst+i));
    }

    /* with dst and src pointing tot he same location
     * assert that the array is the same after a memmove */
    for(uint16_t i = 0; i < length; i++)
    {
        *(src+i) = i;
    }
    status = my_memmove(src, src, length);
    assert_non_null(status);
    for(uint16_t i = 0; i < length; i++)
    {
        assert_int_equal(*(src+i), i);
    }

    /* with dst starting at the middle of src
```

```

* assert that values are correct after memmove */
for(uint16_t i = 0; i < length; i++)
{
    *(src+i) = i;
}
status = my_memmove(src, src+length/3, length/2);
assert_non_null(status);
for(uint16_t i = 0; i < length/2; i++)
{
    assert_int_equal(*(src+length/3+i), i);
}

```

```

/* with src starting at the middle of dst
* asset that values are correct after memmove */
for(uint16_t i = 0; i < length/2; i++)
{
    *(src+length/3+i) = i;
}
status = my_memmove(src+length/3, src, length/2);
assert_non_null(status);
for(uint16_t i = 0; i < length/2; i++)
{
    assert_int_equal(*(src+i), i);
}

```

```

/* send null for src pointer and assert
* that null is returned for a memset*/
status = my_memset((void*)NULL, length, 200);
assert_null(status);

```

```

/* send src and a value, assert that all elements
* of the array are that value for a memset */
status = my_memset(src, length, 200);
assert_non_null(status);
for(uint16_t i = 0; i < length; i++)
{
    assert_int_equal(*(src+i), 200);
}

```

```

/* send null for src pointer and assert
* that null is returned for a memzero*/
status = my_memzero((void*)NULL, length);
assert_null(status);

```

```

/* send src assert that all elements
* of the array are 0 for a memzero */
status = my_memzero(src, length);
assert_non_null(status);
for(uint16_t i = 0; i < length; i++)
{
    assert_int_equal(*(src+i), 0);
}

```

```

/* send null pointer to reverse and assert
 * that a null pointer is returned */
status = my_reverse((void*)NULL, length);
assert_null(status);

/* run mem reverse on a known array of odd length
 * assert that the returned array is reversed */
for(uint16_t i = 0; i < length; i++)
{
    *(src+i) = i;
}
status = my_reverse(src, length);
assert_non_null(status);
for(uint16_t i = 0; i < length; i++)
{
    assert_int_equal(*(src+i), (length-1)-i);
}

/* run mem reverse on a known array of even length
 * assert that the returned array is reversed */
for(uint16_t i = 0; i < length-1; i++)
{
    *(src+i) = i;
}
status = my_reverse(src, length-1);
assert_non_null(status);
for(uint16_t i = 0; i < length-1; i++)
{
    assert_int_equal(*(src+i), (length-2)-i);
}
}

void conversion_test(void **state)
{
    uint8_t ptr_max[10] = "2147483647";
    uint8_t ptr_min[11] = "-2147483647";
    uint8_t ptr_min2[11] = "-2147483647";
    uint32_t statusi = 0;
    uint8_t statusa[10] = "0000000000";
    uint8_t statusl = 0;
    /* send null pointer to atoi and assert that
     * a null pointer is returned */
    statusi = my_atoi((void*)NULL, 3, 10);
    assert_int_equal(statusi, 0);

    /* test sending max sized integer to atoi
     * and assert that the returned value is correct */
    statusi = my_atoi(ptr_max, 10, 10);
    assert_int_equal(statusi, 2147483647);

    /* test sending 0 to atoi and assert that

```

```

    * the returned value is correct */
statusi = my_atoi(ptr_min, 1, 10);
assert_int_equal(statusi, 0);

/* send null pointer to itoi and assert that
 * a null pointer is returned */
statusl = my_itoa(2147483647, (void*)NULL, 10);
assert_int_equal(statusl, 0);

/* test sending max sized integer to itoa
 * and assert that the returned value is correct */
statusl = my_itoa(2147483647, statusa, 10);
assert_int_equal(statusl, 10);
for(uint16_t i = 0; i < 10; i++)
{
    assert_int_equal(*(statusa+i), *(ptr_max+i));
}

/* test sending min to itoa and assert that
 * the returned value is correct */
statusl = my_itoa(-2147483647, statusa, 10);
assert_int_equal(statusl, 11);
for(uint16_t i = 0; i < 1; i++)
{
    assert_int_equal(*(statusa+i), *(ptr_min2+i));
}
}

void data_test(void **state)
{
    int32_t status = 0;
    uint8_t data_orig[4] = {1, 2, 4, 8};
    uint8_t data[4] = {1, 2, 4, 8};
    size_t type_length = 4; /* bytes in a uint32 */
    /* send a null pointer to endiannes conversion
     * asset that a null pointer is returned */
    status = swap_data_endianness((void*)NULL, type_length);
    assert_int_equal(status, SWAP_ERROR);

    /* test that a big endian to little endian
     * conversion works */
    status = swap_data_endianness(data, type_length);
    assert_int_equal(status, SWAP_NO_ERROR);
    for(int i = 0; i < type_length; i++)
    {
        assert_int_equal(*(data+i), *(data_orig+(type_length-1)-i));
    }
    /* undo swap */
    status = swap_data_endianness(data, type_length);
    assert_int_equal(status, SWAP_NO_ERROR);

    /* test that a little endian to big endian

```



```

    * conversion works */
status = swap_data_endianness(data, type_length);
assert_int_equal(status, SWAP_NO_ERROR);
for(int i = 0; i < type_length; i++)
{
    assert_int_equal(*(data+i), *(data_orig+(type_length-1)-i));
}
}

```

```

void cirbuf_test(void **state)
{
    CB_t *my_cbuf = malloc(sizeof(CB_t));
    size_t length = TEST_LENGTH;
    uint8_t data = '0';
    CB_e status = SUCCESS;
    /* snd a null pointer to each of the functions
     * and assert that a null pointer is returned */
    status = CB_init((void*)NULL, length);
    assert_int_equal(status, BAD_POINTER);

    /* initialize a circular buffer and
     * assert that a buffer has been created */
    status = CB_init(my_cbuf, length);
    assert_int_equal(status, SUCCESS);

    /* add then immediately remove and
     * assert that the same item was returned */
    status = CB_buffer_add_item(my_cbuf, 'S');
    assert_int_equal(status, SUCCESS);
    assert_int_equal(status, SUCCESS);
    status = CB_buffer_remove_item(my_cbuf, &data);
    assert_int_equal(status, SUCCESS);
    assert_int_equal(data, 'S');

    /* fill the buffer and assert that itf
     * reports it is full, this also tests if the
     * adding can wrap around properly */
    for(int i = 0; i < length; i++)
    {
        status = CB_buffer_add_item(my_cbuf, i);
        assert_int_equal(status, SUCCESS);
    }

    /* add another item and assert that it
     * fails because the buffer is full. Also
     * assert that the overfil flag is set */
    status = CB_buffer_add_item(my_cbuf, 'S');
    assert_int_equal(status, FULL);
    assert_int_equal(my_cbuf->buff_full_flag, SET);

    /* empty the buffer and assert that
     * it reports that the buffer is empty,

```

```

    * this also tests if removing can wrap around
    * properly */
for(int i = 0; i<length; i++)
{
    status = CB_buffer_remove_item(my_cbuf, &data);
    assert_int_equal(status, SUCCESS);
    assert_int_equal(data, i);
}

/* remove one more item and assert that the buffer
 * fails to remove the item because there
 * are no items in the buffer */
status = CB_buffer_remove_item(my_cbuf, &data);
assert_int_equal(status, EMPTY);

}

int unittest()
{
    /* TODO create these functions in the header files */
    const struct CMUnitTest tests[] =
    {
        cmocka_unit_test(memory_test),
        cmocka_unit_test(conversion_test),
        cmocka_unit_test(data_test),
        cmocka_unit_test(circbuf_test),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}

```