CMSIS-Core Functions Quick Reference

The Cortex® Microcontroller Software Interface Standard contains a number of standardized functions:

- Core peripheral access functions
- Intrinsic functions

In this appendix the basic information about these standardized functions will be covered. Some of the functions in CMSIS use the standard data types defined in "stdint.h." For example:

Table E.1 Standard Data Types Used in CMSIS

Type	Description
uint32_t	Unsigned 32-bit integer
uint16_t	Unsigned 16-bit integer
uint8_t	Unsigned 8-bit integer

E.1 Exception and interrupt numbers

A number of functions in CMSIS use interrupt numbers to access interrupt features. The interrupt number definition is different from the processor IPSR definition. In CMSIS, peripheral interrupts start from value of zero, and negative numbers are used to indicate system exceptions.

Table E.2 Exception and Interrupt Number

CMSIS Interrupt Number	Exception Number in Processor (IPSR)	Exception	Exception Type Name (enum) - "IRQn_Type"	Exception Handler Name
-	-	Reset	-	Reset_Handler
-14	2	NMI	NonMaskableInt_IRQn	NMI_Handler
-13	3	Hard fault	HardFault_IRQn	HardFault_Handler
-12	4	Memory Management Fault	MemoryManagement_IRQn	MemManage_Handler

(*Continued*)

Table E.2 Exception and Interrupt Number—Cont'd

CMSIS Interrupt Number	Exception Number in Processor (IPSR)	Exception	Exception Type Name (enum) - "IRQn_Type"	Exception Handler Name
-11	5	Bus Fault	BusFault_IRQn	BusFault_Handler
-10	6	Usage Fault	UsageFault_IRQn	UsageFault_Handler
-5	11	SVC	SVCall_IRQn	SVC_Handler
-4	12	Debug Monitor	DebugMonitor_IRQn	DebugMon_Handler
-2	14	PendSV	PendSV_IRQn	PendSV_Handler
-1	15	SysTick	SysTick_IRQn	SysTick_Handler
0	16	Peripheral interrupt 0	<MCU specific>	<MCU specific>
1	17	Peripheral interrupt 1	<MCU specific>	<MCU specific>
2	18	Peripheral interrupt 2	<MCU specific>	<MCU specific>
…	…	…	<MCU specific>	<MCU specific>

E.2 NVIC access functions

The following functions are available for NVIC feature accesses:

Function Name	void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
Description	Set the Priority Grouping in NVIC Interrupt Controller. (This function is not available on Cortex-M0/M1.)
Parameter	Priority_grouping is priority grouping field
Return	None

Function Name	uint32_t NVIC_GetPriorityGrouping(void)
Description	Get the Priority Grouping from NVIC Interrupt Controller. (This function is not available on Cortex-M0/M1.)
Parameter	None
Return	Priority grouping field

Function Name	void NVIC_EnableIRQ(IRQn_Type IRQn)
Description	Enable Interrupt in NVIC Interrupt Controller.
Parameter	IRQn_Type IRQn specifies the positive interrupt number. It cannot be system exception.
Return	None

Function Name	void NVIC_DisableIRQ(IRQn_Type IRQn)
Description	Disable Interrupt in NVIC Interrupt Controller.
Parameter	IRQn_Type IRQn is the positive number of the external interrupt. It cannot be system exception.
Return	None

Function Name	uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)
Description	Read the interrupt pending bit for a device-specific interrupt source.
Parameter	IRQn_Type IRQn is the number of the device-specific interrupt. This function does not support system exception.
Return	1 if pending interrupt else 0.

Function Name	void NVIC_SetPendingIRQ(IRQn_Type IRQn)
Description	Set the pending bit for an external interrupt.
Parameter	IRQn_Type IRQn is the Number of the interrupt. This function does not support system exception.
Return	None

Function Name	void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
Description	Clear the pending bit for an external interrupt.
Parameter	IRQn_Type IRQn is the Number of the interrupt. This function does not support system exception.
Return	None

Function Name	uint32_t NVIC_GetActive(IRQn_Type IRQn)
Description	Read the active bit for an external interrupt. (This function is not available on Cortex-M0/M1.)
Parameter	IRQn_Type IRQn is the Number of the interrupt. This function does not support system exception.
Return	1 if active else 0

Function Name	void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
Description	Set the priority for an interrupt or system exceptions with programmable priority level.
Parameter	IRQn_Type IRQn is the Number of the interrupt. unint32_t priority is the priority for the interrupt. This function automatically shifts the input priority value left to put priority value in implemented bits.
Return	None

Function Name	uint32_t NVIC_GetPriority(IRQn_Type IRQn)
Description	Read the priority for an interrupt or system exceptions with programmable priority level.
Parameter	IRQn_Type IRQn is the Number of the interrupt.
Return	uint32_t priority is the priority for the interrupt. This function automatically shifts the input priority value right to remove unimplemented bits in the priority value register.

Function Name	uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)
Description	Encode the priority for an interrupt – Encode the priority for an interrupt with the given priority group, preemptive priority (group priority) value and sub priority value. In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS) the smallest possible priority group is set. (This function is not available on Cortex-M0/M1)

—Cont'd

Function Name	uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)
Parameter	PriorityGroup is the used priority group PreemptPriority is the preemptive priority value (group priority) (starting from 0). SubPriority is the sub priority value (starting from 0).
Return	The priority for the interrupt.

Function Name	void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)
Description	Decode the priority of an interrupt – Decode an interrupt priority value with the given priority group to preemptive priority value (group priority) and sub priority value. In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS) the smallest possible priority group is set. (This function is not available on Cortex-M0/M1.)
Parameter	Priority is the priority for the interrupt PrioGroup is the used priority group pPreemptPrio is the preemptive priority value (starting from 0) pSubPrio is the sub priority value (starting from 0)
Return	None

E.3 System and systick functions

The following functions are for system setup:

Function Name	void SystemInit (void)
Description	Initialize the system
Parameter	None
Return	None

Function Name	void NVIC_SystemReset(void)
Description	Initiate a system reset request.
Parameter	None
Return	None

Function Name	uint32_t SysTick_Config(uint32_t ticks)
Description	Initialize and start the SysTick counter and its interrupt. This function program the SysTick to generate SysTick exception for every "ticks" number of core clock cycles.
Parameter	Ticks is the number of clock ticks between two interrupts.
Return	None

E.4 Core registers access functions

The following functions are for accessing special registers in the processor core:

Table E.3 Core Registers Access Functions

CMSIS-Core Functions for Accessing Special Registers Available for Cortex-M3 and Cortex-M4	
uint32_t	**__get_CONTROL** (void) Read the CONTROL register.
void	**__set_CONTROL** (uint32_t control) Set the CONTROL Register.
uint32_t	**__get_IPSR** (void) Read the IPSR register.
uint32_t	**__get_APSR** (void) Read the APSR register.
uint32_t	**__get_xPSR** (void) Read the xPSR register.
uint32_t	**__get_PSP** (void) Read the PSP register.
void	**__set_PSP** (uint32_t topOfProcStack) Set the PSP register.

Table E.3 Core Registers Access Functions—Cont'd

CMSIS-Core Functions for Accessing Special Registers Available for Cortex-M3 and Cortex-M4	
uint32_t	**__get_MSP** (void) Read the MSP register.
void	**__set_MSP** (uint32_t topOfMainStack) Set the MSP register.
uint32_t	**__get_PRIMASK** (void) Read the PRIMASK register bit.
void	**__set_PRIMASK** (uint32_t priMask) Set the PRIMASK bit.
uint32_t	**__get_BASEPRI** (void) Read the BASEPRI register [not for Cortex-M0 variants].
void	**__set_BASEPRI** (uint32_t basePri) Set the BASEPRI register [not for Cortex-M0 variants].
uint32_t	**__get_FAULTMASK** (void) Read the FAULTMASK register [not for Cortex-M0 variants].
void	**__set_FAULTMASK** (uint32_t faultMask) Set the FAULTMASK register [not for Cortex-M0 variants].
uint32_t	**__get_FPSCR** (void) Read the FPSCR register [only for Cortex-M4].
void	**__set_FPSCR** (uint32_t fpscr) Set the FPSC register [only for Cortex-M4].

E.5 CMSIS intrinsic functions

The CMSIS provide a number of intrinsic functions for access to instructions that cannot be generated by ISO/IEC C. The function "__enable_fault_irq" and "__disable_fault_irq" below are not available for Cortex®-M0/M1.

Functions for system features.

Table E.4 System Instruction Functions

Instructions	CMSIS Functions Available for Cortex-M3 and Cortex-M4	
CPSIE I	**void**	`__enable_irq(void)` Clear PRIMASK, enable interrupts
CPSID I	**void**	`__disable_irq(void)` Set PRIMASK, disable interrupts
CPSIE F	**void**	`__enable_fault_irq(void)` Clear FAULTMASK, enable interrupts – not available on Cortex-M0/M0+/M1
CPSID F	**void**	`__disable_fault_irq(void)` Set FAULTMASK, disable interrupts including HardFault – not available on Cortex-M0/M0+/M1
DMB	**void**	`__DMB(void)` Data Memory Barrier
DSB	**void**	`__DSB(void)` Data Synchronization Barrier
ISB	**void**	`__ISB(void)` Instruction Synchronization Barrier
NOP	**void**	`__NOP(void)` No Operation
SEV	**void**	`__SEV(void)` Send Event
WFI	**void**	`__WFI(void)` Wait for Interrupt (enter sleep)
WFE	**void**	`__WFE(void)` Wait for Event (enter sleep conditionally, clear event latch)
BKPT	**void**	`__BKPT(uint8_t value)` Set a software break point (available From CMSIS V3.20)

Functions for exclusive memory accesses – these functions are not available on Cortex-M0/M1.

Table E.5 Exclusive Access Instruction Functions

Instructions	CMSIS Functions Available for Cortex-M3 and Cortex-M4	
CLREX	**void**	`__CLREX(void)` Remove the exclusive access state created by exclusive load
LDREXB	**uint8_t**	`__LDREXB(volatile uint8_t *addr)` Exclusive load byte

Table E.5 Exclusive Access Instruction Functions—Cont'd

Instructions	CMSIS Functions Available for Cortex-M3 and Cortex-M4	
LDREXH	**uint16_t**	`__LDREXH(volatile uint16_t *addr)`
		Exclusive load half word
LDREX	**uint32_t**	`__LDREXW(volatile uint32_t *addr)`
		Exclusive store word
STREXB	**uint32_t**	`__STREXB(uint8_t value, volatile uint8_t *addr)`
		Exclusive store byte. Return value is the access status (success = 0, failed = 1).
STREXH	**uint32_t**	`__STREXH(uint16_t value, volatile uint16_t *addr)`
		Exclusive store halfword. Return value is the access status (success = 0, failed = 1).
STREX	**uint32_t**	`__STREXW(uint32_t value, volatile uint32_t *addr)`
		Exclusive store word. Return value is the access status (success = 0, failed = 1).

Functions for data processing − CLZ, RBIT, SSAT and USAT are not available for Cortex-M0/M1.

Table E.6 Data Processing Instruction Functions for Cortex-M3 and Cortex-M4 Processors

Instructions	CMSIS Functions Available for Cortex-M3 and Cortex-M4	
CLZ	**uint8_t**	`__CLZ(unsigned int val)`
		Count Leading Zero
RBIT	**uint32_t**	`__RBIT(uint32_t val)`
		Reverse bits in word
REV	**uint32_t**	`__REV(uint32_t value)`
		Reverse byte order within a word
REV16	**uint32_t**	`__REV16(uint16_t value)`
		Reverse byte order within each half word independently
REVSH	**int32_t**	`__REVSH(int16_t value)`
		Reverse byte order in the lower halfword, and then sign extend the result in a 32-bit word

(*Continued*)

Table E.6 Data Processing Instruction Functions for Cortex-M3 and Cortex-M4 Processors—Cont'd

Instructions	CMSIS Functions Available for Cortex-M3 and Cortex-M4	
ROR	**uint32_t**	`__ROR(uint32_t val, uint32_t shift)` `Rotate value right by a number of bits`
SSAT	**uint32_t**	`__SSAT(uint32_t value, uint32_t sat);` `Signed saturate`
USAT	**uint32_t**	`__USAT(uint32_t value, uint32_t sat)` `Unsigned saturate`

Table E.7 CMSIS-Core Intrinsic Functions for DSP Related Operations in Cortex-M4 Processor

CMSIS Functions Available for Cortex-M4	
uint32_t	**__SADD8** (uint32_t val1, uint32_t val2) GE setting quad 8-bit signed addition.
uint32_t	**__QADD8** (uint32_t val1, uint32_t val2) Q setting quad 8-bit saturating addition.
uint32_t	**__SHADD8** (uint32_t val1, uint32_t val2) Quad 8-bit signed addition with halved results.
uint32_t	**__UADD8** (uint32_t val1, uint32_t val2) GE setting quad 8-bit unsigned addition.
uint32_t	**__UQADD8** (uint32_t val1, uint32_t val2) Quad 8-bit unsigned saturating addition.
uint32_t	**__UHADD8** (uint32_t val1, uint32_t val2) Quad 8-bit unsigned addition with halved results.
uint32_t	**__SSUB8** (uint32_t val1, uint32_t val2) GE setting quad 8-bit signed subtraction.
uint32_t	**__QSUB8** (uint32_t val1, uint32_t val2) Q setting quad 8-bit saturating subtract.
uint32_t	**__SHSUB8** (uint32_t val1, uint32_t val2) Quad 8-bit signed subtraction with halved results.
uint32_t	**__USUB8** (uint32_t val1, uint32_t val2) GE setting quad 8-bit unsigned subtract.
uint32_t	**__UQSUB8** (uint32_t val1, uint32_t val2) Quad 8-bit unsigned saturating subtraction.
uint32_t	**__UHSUB8** (uint32_t val1, uint32_t val2) Quad 8-bit unsigned subtraction with halved results.
uint32_t	**__SADD16** (uint32_t val1, uint32_t val2) GE setting dual 16-bit signed addition.

Table E.7 CMSIS-Core Intrinsic Functions for DSP Related Operations in Cortex-M4 Processor—Cont'd

CMSIS Functions Available for Cortex-M4	
uint32_t	**__QADD16** (uint32_t val1, uint32_t val2) Q setting dual 16-bit saturating addition.
uint32_t	**__SHADD16** (uint32_t val1, uint32_t val2) Dual 16-bit signed addition with halved results.
uint32_t	**__UADD16** (uint32_t val1, uint32_t val2) GE setting dual 16-bit unsigned addition.
uint32_t	**__UQADD16** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned saturating addition.
uint32_t	**__UHADD16** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned addition with halved results.
uint32_t	**__SSUB16** (uint32_t val1, uint32_t val2) GE setting dual 16-bit signed subtraction.
uint32_t	**__QSUB16** (uint32_t val1, uint32_t val2) Q setting dual 16-bit saturating subtract.
uint32_t	**__SHSUB16** (uint32_t val1, uint32_t val2) Dual 16-bit signed subtraction with halved results.
uint32_t	**__USUB16** (uint32_t val1, uint32_t val2) GE setting dual 16-bit unsigned subtract.
uint32_t	**__UQSUB16** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned saturating subtraction.
uint32_t	**__UHSUB16** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned subtraction with halved results.
uint32_t	**__SASX** (uint32_t val1, uint32_t val2) GE setting dual 16-bit addition and subtraction with exchange.
uint32_t	**__QASX** (uint32_t val1, uint32_t val2) Q setting dual 16-bit add and subtract with exchange.
uint32_t	**__SHASX** (uint32_t val1, uint32_t val2) Dual 16-bit signed addition and subtraction with halved results.
uint32_t	**__UASX** (uint32_t val1, uint32_t val2) GE setting dual 16-bit unsigned addition and subtraction with exchange.
uint32_t	**__UQASX** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned saturating addition and subtraction with exchange.
uint32_t	**__UHASX** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned addition and subtraction with halved results and exchange.

(*Continued*)

Table E.7 CMSIS-Core Intrinsic Functions for DSP Related Operations in Cortex-M4 Processor—Cont'd

CMSIS Functions Available for Cortex-M4	
uint32_t	**__SSAX** (uint32_t val1, uint32_t val2) GE setting dual 16-bit signed subtraction and addition with exchange.
uint32_t	**__QSAX** (uint32_t val1, uint32_t val2) Q setting dual 16-bit subtract and add with exchange.
uint32_t	**__SHSAX** (uint32_t val1, uint32_t val2) Dual 16-bit signed subtraction and addition with halved results.
uint32_t	**__USAX** (uint32_t val1, uint32_t val2) GE setting dual 16-bit unsigned subtract and add with exchange.
uint32_t	**__UQSAX** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned saturating subtraction and addition with exchange.
uint32_t	**__UHSAX** (uint32_t val1, uint32_t val2) Dual 16-bit unsigned subtraction and addition with halved results and exchange.
uint32_t	**__USAD8** (uint32_t val1, uint32_t val2) Unsigned sum of quad 8-bit unsigned absolute difference.
uint32_t	**__USADA8** (uint32_t val1, uint32_t val2, uint32_t val3) Unsigned sum of quad 8-bit unsigned absolute difference with 32-bit accumulate.
uint32_t	**__SSAT16** (uint32_t val1, const uint32_t val2) Q setting dual 16-bit saturate.
uint32_t	**__USAT16** (uint32_t val1, const uint32_t val2) Q setting dual 16-bit unsigned saturate.
uint32_t	**__UXTB16** (uint32_t val) Dual extract 8-bits and zero-extend to 16-bits.
uint32_t	**__UXTAB16** (uint32_t val1, uint32_t val2) Extracted 16-bit to 32-bit unsigned addition.
uint32_t	**__SXTB16** (uint32_t val) Dual extract 8-bits and sign extend each to 16-bits.
uint32_t	**__SXTAB16** (uint32_t val1, uint32_t val2) Dual extracted 8-bit to 16-bit signed addition.
uint32_t	**__SMUAD** (uint32_t val1, uint32_t val2) Q setting sum of dual 16-bit signed multiply.
uint32_t	**__SMUADX** (uint32_t val1, uint32_t val2) Q setting sum of dual 16-bit signed multiply with exchange.

Table E.7 CMSIS-Core Intrinsic Functions for DSP Related Operations in Cortex-M4 Processor—Cont'd

CMSIS Functions Available for Cortex-M4	
uint32_t	**__SMLAD** (uint32_t val1, uint32_t val2, uint32_t val3) Q setting dual 16-bit signed multiply with single 32-bit accumulator.
uint32_t	**__SMLADX** (uint32_t val1, uint32_t val2, uint32_t val3) Q setting pre-exchanged dual 16-bit signed multiply with single 32-bit accumulator.
uint64_t	**__SMLALD** (uint32_t val1, uint32_t val2, uint64_t val3) Dual 16-bit signed multiply with single 64-bit accumulator.
unsigned long long	**__SMLALDX** (uint32_t val1, uint32_t val2, unsigned long long val3) Dual 16-bit signed multiply with exchange with single 64-bit accumulator.
uint32_t	**__SMUSD** (uint32_t val1, uint32_t val2) Dual 16-bit signed multiply returning difference.
uint32_t	**__SMUSDX** (uint32_t val1, uint32_t val2) Dual 16-bit signed multiply with exchange returning difference.
uint32_t	**__SMLSD** (uint32_t val1, uint32_t val2, uint32_t val3) Q setting dual 16-bit signed multiply subtract with 32-bit accumulate.
uint32_t	**__SMLSDX** (uint32_t val1, uint32_t val2, uint32_t val3) Q setting dual 16-bit signed multiply with exchange subtract with 32-bit accumulate.
uint64_t	**__SMLSLD** (uint32_t val1, uint32_t val2, uint64_t val3) Q setting dual 16-bit signed multiply subtract with 64-bit accumulate.
unsigned long long	**__SMLSLDX** (uint32_t val1, uint32_t val2, unsigned long long val3) Q setting dual 16-bit signed multiply with exchange subtract with 64-bit accumulate.
uint32_t	**__SEL** (uint32_t val1, uint32_t val2) Select bytes based on GE bits.
uint32_t	**__QADD** (uint32_t val1, uint32_t val2) Q setting saturating add.
uint32_t	**__QSUB** (uint32_t val1, uint32_t val2) Q setting saturating subtract.
uint32_t	**__PKHBT** (uint32_t val1, uint32_t val2, uint32_t val3) Halfword packing instruction. Combines bits[15:0] of val1 with bits[31:16] of *val2* levitated with the *val3*.
uint32_t	**__PKHTB** (uint32_t val1, uint32_t val2, uint32_t val3) Halfword packing instruction. Combines bits[31:16] of *val1* with bits[15:0] of *val2* right-shifted with the *val3*.

E.6 Debug message output function

A debug message output function is defined to use ITM for message output.

Function Name	uint32_t ITM_SendChar(uint32_t ch)
Description	Output a character via the ITM output channel 0. When no debugger is connected the function return immediately. If debugger is connected and instrumentation trace is enabled, the function output the character to ITM and stalls if the ITM is still busy on the last transfer.
Parameter	“ch” is the character to be output.
Return	The output character “ch”.

Function Name	uint32_t ITM_ReceiveChar(void)
Description	Check the ITM_RxBuffer variable to see if it is empty. If it is empty then return -1, otherwise return the character received.
Parameter	None
Return	The receive character if available, or -1 if the buffer is empty.

Function Name	uint32_t ITM_CheckChar(void)
Description	Check the ITM_RxBuffer variable to see if it is empty. If it is empty then return 0, otherwise return 1.
Parameter	None
Return	Return 0 is no received data, and return 1 if a data is received.