

ADAS Graduation Project

Generated on Sun May 28 2023 01:35:11 for ADAS Graduation Project by Doxygen 1.9.5

Sun May 28 2023 01:35:11

1 ADAS Graudation Project	1
1.1 Description	1
1.1.1 Adaptive Cruise Control	1
1.1.2 Adaptive Light Control	1
1.1.3 Driver Drowsiness Detection	1
1.2 Hardware specifications	1
2 Module Index	3
2.1 Modules	3
3 Data Structure Index	5
3.1 Data Structures	5
4 File Index	7
4.1 File List	7
5 Module Documentation	11
5.1 DC Motor Directions	11
5.1.1 Detailed Description	11
5.1.2 Macro Definition Documentation	11
5.1.2.1 FORWARD	11
5.1.2.2 BACKWARD	11
5.2 DC Motor Speeds	12
5.2.1 Detailed Description	12
5.2.2 Macro Definition Documentation	12
5.2.2.1 SPEED_0_PERCENT	12
5.2.2.2 SPEED_10_PERCENT	13
5.2.2.3 SPEED_20_PERCENT	13
5.2.2.4 SPEED_30_PERCENT	13
5.2.2.5 SPEED_40_PERCENT	13
5.2.2.6 SPEED_50_PERCENT	14
5.2.2.7 SPEED_60_PERCENT	14
5.2.2.8 SPEED_70_PERCENT	14
5.2.2.9 SPEED_80_PERCENT	14
5.2.2.10 SPEED_90_PERCENT	15
5.2.2.11 SPEED_100_PERCENT	15
5.3 DC Motor Rotation Directions	15
5.3.1 Detailed Description	15
5.3.2 Macro Definition Documentation	15
5.3.2.1 CW	15
5.3.2.2 CCW	16
5.4 Bit Manipulation Math Macros	16
5.4.1 Detailed Description	16
5.4.2 Macro Definition Documentation	16

5.4.2.1 SET_BIT	16
5.4.2.2 CLR_BIT	16
5.4.2.3 TOGGLE_BIT	17
5.4.2.4 GET_BIT	17
5.5 Bit Group Manipulation Math Macros	17
5.5.1 Detailed Description	17
5.5.2 Macro Definition Documentation	17
5.5.2.1 SET_BITS	17
5.5.2.2 CLR_BITS	18
5.5.2.3 TOGGLE_BITS	18
5.5.2.4 GET_BITS	18
5.6 Compiler standard macros	19
5.6.1 Detailed Description	19
5.6.2 Macro Definition Documentation	19
5.6.2.1 VAR	19
5.6.2.2 FUNC	19
5.6.2.3 P2VAR	20
5.6.2.4 P2CONST	20
5.6.2.5 CONSTP2VAR	20
5.6.2.6 CONSTP2CONST	20
5.6.2.7 P2FUNC	21
5.6.2.8 CONST	21
5.6.2.9 STATIC	21
5.7 Utilities macros	21
5.7.1 Detailed Description	21
5.7.2 Macro Definition Documentation	21
5.7.2.1 MY_MS_DELAY	22
5.8 Standard types	22
5.8.1 Detailed Description	22
5.8.2 Typedef Documentation	22
5.8.2.1 bool_t	22
5.8.2.2 u8_t	23
5.8.2.3 c8_t	23
5.8.2.4 u16_t	23
5.8.2.5 u32_t	23
5.8.2.6 u64_t	24
5.8.2.7 s64_t	24
5.8.2.8 s8_t	24
5.8.2.9 s16_t	24
5.8.2.10 s32_t	25
5.8.2.11 f32_t	25
5.8.2.12 f64_t	25

5.9 Standard values	25
5.9.1 Detailed Description	26
5.9.2 Macro Definition Documentation	26
5.9.2.1 TRUE	26
5.9.2.2 FALSE	26
5.9.2.3 NULL	26
5.9.2.4 INITIAL_ZERO	26
5.9.2.5 FLAG_SET	27
5.9.2.6 FLAG_CLEARED	27
5.9.2.7 RUN	27
5.9.2.8 STOP	27
5.9.2.9 PRESSED	28
5.9.2.10 RELEASED	28
5.9.2.11 SAME_STRING	28
5.9.2.12 DIFFERENT_STRING	28
5.10 ADC Channels	29
5.10.1 Detailed Description	29
5.10.2 Macro Definition Documentation	29
5.10.2.1 CHANNEL0	30
5.10.2.2 CHANNEL1	30
5.10.2.3 CHANNEL2	30
5.10.2.4 CHANNEL3	30
5.10.2.5 CHANNEL4	31
5.10.2.6 CHANNEL5	31
5.10.2.7 CHANNEL6	31
5.10.2.8 CHANNEL7	31
5.10.2.9 CHANNEL8	32
5.10.2.10 CHANNEL9	32
5.10.2.11 CHANNEL10	32
5.10.2.12 CHANNEL11	32
5.10.2.13 CHANNEL12	33
5.10.2.14 CHANNEL13	33
5.10.2.15 CHANNEL14	33
5.10.2.16 CHANNEL15	33
5.10.2.17 CHANNEL16	34
5.10.2.18 CHANNEL17	34
5.10.2.19 CHANNEL18	34
5.11 Interrupt line IDs	35
5.11.1 Detailed Description	35
5.11.2 Macro Definition Documentation	35
5.11.2.1 EXTI_LINE0	35
5.11.2.2 EXTI_LINE1	36

5.11.2.3 EXTI_LINE2	36
5.11.2.4 EXTI_LINE3	36
5.11.2.5 EXTI_LINE4	36
5.11.2.6 EXTI_LINE5	37
5.11.2.7 EXTI_LINE6	37
5.11.2.8 EXTI_LINE7	37
5.11.2.9 EXTI_LINE8	37
5.11.2.10 EXTI_LINE9	38
5.11.2.11 EXTI_LINE10	38
5.11.2.12 EXTI_LINE11	38
5.11.2.13 EXTI_LINE12	38
5.11.2.14 EXTI_LINE13	39
5.11.2.15 EXTI_LINE14	39
5.11.2.16 EXTI_LINE15	39
5.12 Interrupt trigger status	39
5.12.1 Detailed Description	39
5.12.2 Macro Definition Documentation	39
5.12.2.1 EXTI_FallingEdge	40
5.12.2.2 EXTI_RisingEdge	40
5.12.2.3 EXTI_OnChange	40
5.13 EXTI Memory Addresses	40
5.13.1 Detailed Description	40
5.13.2 Macro Definition Documentation	40
5.13.2.1 EXTI_BASE_ADDRESS	41
5.14 EXTI Registers	41
5.14.1 Detailed Description	41
5.14.2 Macro Definition Documentation	41
5.14.2.1 MEXTI	41
5.15 EXTI Lines Status	41
5.15.1 Detailed Description	41
5.15.2 Macro Definition Documentation	41
5.15.2.1 ENABLE	42
5.15.2.2 DISABLE	42
5.16 EXTI Line Settings	42
5.16.1 Detailed Description	42
5.16.2 Macro Definition Documentation	42
5.16.2.1 EXTI_MAX_EXTI_NUM	42
5.17 SYSCFG Memory Addresses	43
5.17.1 Detailed Description	43
5.17.2 Macro Definition Documentation	43
5.17.2.1 SYSCFG_BASE_ADDR	43
5.18 SYSCFG Memory Registers	43

5.18.1 Detailed Description	43
5.18.2 Macro Definition Documentation	43
5.18.2.1 MSYSCFG	43
5.19 GPIO Modes	44
5.19.1 Detailed Description	44
5.19.2 Macro Definition Documentation	44
5.19.2.1 GPIOx_MODE_INPUT	44
5.19.2.2 GPIOx_MODE_OUTPUT	44
5.19.2.3 GPIOx_MODE_AF	44
5.19.2.4 GPIOx_MODE_ANALOG	45
5.20 GPIO Ports	45
5.20.1 Detailed Description	45
5.20.2 Macro Definition Documentation	45
5.20.2.1 GPIO_PORTA	45
5.20.2.2 GPIO_PORTB	45
5.20.2.3 GPIO_PORTC	46
5.21 GPIO Output Types	46
5.21.1 Detailed Description	46
5.21.2 Macro Definition Documentation	46
5.21.2.1 GPIOx_OPENDRAIN	46
5.21.2.2 GPIOx_PUSH_PULL	46
5.22 GPIO PIN Speed	47
5.22.1 Detailed Description	47
5.22.2 Macro Definition Documentation	47
5.22.2.1 GPIOx_LowSpeed	47
5.22.2.2 GPIOx_MediumSpeed	47
5.22.2.3 GPIOx_HighSpeed	47
5.22.2.4 GPIOx_VeryHighSpeed	48
5.23 GPIO Pull Types	48
5.23.1 Detailed Description	48
5.23.2 Macro Definition Documentation	48
5.23.2.1 GPIOx_NoPull	48
5.23.2.2 GPIOx_PullUp	48
5.23.2.3 GPIOx_PullDown	49
5.24 GPIO Output Values	49
5.24.1 Detailed Description	49
5.24.2 Macro Definition Documentation	49
5.24.2.1 GPIOx_HIGH	49
5.24.2.2 GPIOx_LOW	49
5.25 GPIO Output PINs	50
5.25.1 Detailed Description	50
5.25.2 Macro Definition Documentation	50

5.25.2.1 GPIOx_PIN0	50
5.25.2.2 GPIOx_PIN1	51
5.25.2.3 GPIOx_PIN2	51
5.25.2.4 GPIOx_PIN3	51
5.25.2.5 GPIOx_PIN4	51
5.25.2.6 GPIOx_PIN5	52
5.25.2.7 GPIOx_PIN6	52
5.25.2.8 GPIOx_PIN7	52
5.25.2.9 GPIOx_PIN8	52
5.25.2.10 GPIOx_PIN9	53
5.25.2.11 GPIOx_PIN10	53
5.25.2.12 GPIOx_PIN11	53
5.25.2.13 GPIOx_PIN12	53
5.25.2.14 GPIOx_PIN13	54
5.25.2.15 GPIOx_PIN14	54
5.25.2.16 GPIOx_PIN15	54
5.26 GPIO Alternate Functions	54
5.26.1 Detailed Description	55
5.26.2 Macro Definition Documentation	55
5.26.2.1 GPIOx_AF0	55
5.26.2.2 GPIOx_AF1	55
5.26.2.3 GPIOx_AF2	56
5.26.2.4 GPIOx_AF3	56
5.26.2.5 GPIOx_AF4	56
5.26.2.6 GPIOx_AF5	56
5.26.2.7 GPIOx_AF6	57
5.26.2.8 GPIOx_AF7	57
5.26.2.9 GPIOx_AF8	57
5.26.2.10 GPIOx_AF9	57
5.26.2.11 GPIOx_AF10	58
5.26.2.12 GPIOx_AF11	58
5.26.2.13 GPIOx_AF12	58
5.26.2.14 GPIOx_AF13	58
5.26.2.15 GPIOx_AF14	59
5.26.2.16 GPIOx_AF15	59
5.27 GPIO Addresses	59
5.27.1 Detailed Description	59
5.27.2 Macro Definition Documentation	59
5.27.2.1 GPIOA_BASE_ADDRESS	59
5.27.2.2 GPIOB_BASE_ADDRESS	60
5.27.2.3 GPIOC_BASE_ADDRESS	60
5.28 GPIO Registers	60

5.28.1 Detailed Description	60
5.28.2 Macro Definition Documentation	60
5.28.2.1 GPIOA	60
5.28.2.2 GPIOB	61
5.28.2.3 GPIOC	61
5.29 Interrupt priority grouping	61
5.29.1 Detailed Description	62
5.29.2 Macro Definition Documentation	62
5.29.2.1 _16GROUP_NoSub_Priorities	62
5.29.2.2 _8GROUP_2Sub_Priorities	62
5.29.2.3 _4GROUP_4Sub_Priorities	63
5.29.2.4 _2GROUP_8Sub_Priorities	63
5.29.2.5 NoGROUP_16Sub_Priorities	63
5.29.2.6 GROUP_4BITS	64
5.29.2.7 GROUP_3BITS	64
5.29.2.8 GROUP_2BITS	64
5.29.2.9 GROUP_1BITS	64
5.29.2.10 GROUP_0BITS	65
5.30 Group priorities	65
5.30.1 Detailed Description	66
5.30.2 Macro Definition Documentation	66
5.30.2.1 NO_GROUP_PRIORITY	66
5.30.2.2 GROUP_PRIORITY_0	66
5.30.2.3 GROUP_PRIORITY_1	66
5.30.2.4 GROUP_PRIORITY_2	67
5.30.2.5 GROUP_PRIORITY_3	67
5.30.2.6 GROUP_PRIORITY_4	67
5.30.2.7 GROUP_PRIORITY_5	67
5.30.2.8 GROUP_PRIORITY_6	68
5.30.2.9 GROUP_PRIORITY_7	68
5.30.2.10 GROUP_PRIORITY_8	68
5.30.2.11 GROUP_PRIORITY_9	68
5.30.2.12 GROUP_PRIORITY_10	69
5.30.2.13 GROUP_PRIORITY_11	69
5.30.2.14 GROUP_PRIORITY_12	69
5.30.2.15 GROUP_PRIORITY_13	69
5.30.2.16 GROUP_PRIORITY_14	70
5.30.2.17 GROUP_PRIORITY_15	70
5.31 Sub-Group priorities	70
5.31.1 Detailed Description	71
5.31.2 Macro Definition Documentation	71
5.31.2.1 NO_SUB_PRIORITY	71

5.31.2.2 SUB_PRIORITY_0	71
5.31.2.3 SUB_PRIORITY_1	72
5.31.2.4 SUB_PRIORITY_2	72
5.31.2.5 SUB_PRIORITY_3	72
5.31.2.6 SUB_PRIORITY_4	72
5.31.2.7 SUB_PRIORITY_5	73
5.31.2.8 SUB_PRIORITY_6	73
5.31.2.9 SUB_PRIORITY_7	73
5.31.2.10 SUB_PRIORITY_8	73
5.31.2.11 SUB_PRIORITY_9	74
5.31.2.12 SUB_PRIORITY_10	74
5.31.2.13 SUB_PRIORITY_11	74
5.31.2.14 SUB_PRIORITY_12	74
5.31.2.15 SUB_PRIORITY_13	75
5.31.2.16 SUB_PRIORITY_14	75
5.31.2.17 SUB_PRIORITY_15	75
5.32 NVIC Vector Table	75
5.32.1 Detailed Description	78
5.32.2 Macro Definition Documentation	78
5.32.2.1 WWDG	78
5.32.2.2 EXTI16	78
5.32.2.3 EXTI21	79
5.32.2.4 EXTI22	79
5.32.2.5 FLASH	79
5.32.2.6 RCC	79
5.32.2.7 EXTI0	80
5.32.2.8 EXTI1	80
5.32.2.9 EXTI2	80
5.32.2.10 EXTI3	80
5.32.2.11 EXTI4	81
5.32.2.12 DMA1_STREAM0	81
5.32.2.13 DMA1_STREAM1	81
5.32.2.14 DMA1_STREAM2	81
5.32.2.15 DMA1_STREAM3	82
5.32.2.16 DMA1_STREAM4	82
5.32.2.17 DMA1_STREAM5	82
5.32.2.18 DMA1_STREAM6	82
5.32.2.19 ADC	83
5.32.2.20 EXTI9	83
5.32.2.21 TIM1_BRK_TIM9	83
5.32.2.22 TIM1_UP_TIM10	83
5.32.2.23 TIM1_TRG_COM_TIM11	84

5.32.2.24 TIM1_CC	84
5.32.2.25 TIM2	84
5.32.2.26 TIM3	84
5.32.2.27 TIM4	85
5.32.2.28 I2C1_EV	85
5.32.2.29 I2C1_ER	85
5.32.2.30 I2C2_EV	85
5.32.2.31 I2C2_ER	86
5.32.2.32 SPI1	86
5.32.2.33 SPI2	86
5.32.2.34 USART1	86
5.32.2.35 USART2	87
5.32.2.36 EXTI15_10	87
5.32.2.37 EXTI17	87
5.32.2.38 EXTI18	87
5.32.2.39 DMA1_STREAM7	88
5.32.2.40 SDIO	88
5.32.2.41 TIM5	88
5.32.2.42 SPI3	88
5.32.2.43 DMA2_STREAM0	89
5.32.2.44 DMA2_STREAM1	89
5.32.2.45 DMA2_STREAM2	89
5.32.2.46 DMA2_STREAM3	89
5.32.2.47 DMA2_STREAM4	90
5.32.2.48 OTG_FS	90
5.32.2.49 DMA2_STREAM5	90
5.32.2.50 DMA2_STREAM6	90
5.32.2.51 DMA2_STREAM7	91
5.32.2.52 USART6	91
5.32.2.53 I2C3_EV	91
5.32.2.54 I2C3_ER	91
5.32.2.55 FPU	92
5.32.2.56 SPI4	92
5.33 NVIC Addresses	92
5.33.1 Detailed Description	92
5.33.2 Macro Definition Documentation	92
5.33.2.1 NVIC_BASE_ADDRESS	93
5.33.2.2 SCB_BASE_ADDRESS	93
5.34 NVIC Registers	93
5.34.1 Detailed Description	93
5.34.2 Macro Definition Documentation	93
5.34.2.1 MSCB	93

5.34.2.2 MNVIC	94
5.35 NVIC Settable Priorities	94
5.35.1 Detailed Description	94
5.35.2 Macro Definition Documentation	94
5.35.2.1 PEND_SV	94
5.35.2.2 SYSTICK	95
5.35.2.3 SV_CALL	95
5.35.2.4 MEMORY_MANAGE	95
5.35.2.5 BUS_FAULT	95
5.35.2.6 USAGE_FAULT	96
5.36 ID options	96
5.36.1 Detailed Description	96
5.36.2 Macro Definition Documentation	96
5.36.2.1 RCC_AHB1	96
5.36.2.2 RCC_AHB2	97
5.36.2.3 RCC_APB1	97
5.36.2.4 RCC_APB2	97
5.36.2.5 RCC_AHB1LPENR	97
5.37 ID options	98
5.37.1 Detailed Description	99
5.37.2 Macro Definition Documentation	99
5.37.2.1 AHB1ENR_DMA2EN	99
5.37.2.2 AHB1ENR_DMA1EN	99
5.37.2.3 AHB1ENR_CRCEN	100
5.37.2.4 AHB1ENR_GPIOHEN	100
5.37.2.5 AHB1ENR_GPIOEEN	100
5.37.2.6 AHB1ENR_GPIODEN	100
5.37.2.7 AHB1ENR_GPIOCEN	101
5.37.2.8 AHB1ENR_GPIOBEN	101
5.37.2.9 AHB1ENR_GPIOAEN	101
5.37.2.10 AHB2ENR_OTGFSEN	101
5.37.2.11 APB1ENR_PWREN	102
5.37.2.12 APB1ENR_I2C3EN	102
5.37.2.13 APB1ENR_I2C2EN	102
5.37.2.14 APB1ENR_I2C1EN	102
5.37.2.15 APB1ENR_USART2EN	103
5.37.2.16 APB1ENR_SPI3EN	103
5.37.2.17 APB1ENR_SPI2EN	103
5.37.2.18 APB1ENR_WWDGEN	103
5.37.2.19 APB1ENR_TIM5EN	104
5.37.2.20 APB1ENR_TIM4EN	104
5.37.2.21 APB1ENR_TIM3EN	104

5.37.2.22 APB1ENR_TIM2EN	104
5.37.2.23 APB2ENR_TIM11EN	105
5.37.2.24 APB2ENR_TIM10EN	105
5.37.2.25 APB2ENR_TIM9EN	105
5.37.2.26 APB2ENR_SYSCFGGEN	105
5.37.2.27 APB2ENR_SPI4EN	106
5.37.2.28 APB2ENR_SPI1EN	106
5.37.2.29 APB2ENR_SDIOEN	106
5.37.2.30 APB2ENR_ADC1EN	106
5.37.2.31 APB2ENR_USART6EN	107
5.37.2.32 APB2ENR_USART1EN	107
5.37.2.33 APB2ENR_TIM1EN	107
5.37.2.34 AHB1LPENR_FLITFLPEN	107
5.38 SPI Peripherals	107
5.38.1 Detailed Description	108
5.39 SPI Addresses	108
5.39.1 Detailed Description	108
5.39.2 Macro Definition Documentation	108
5.39.2.1 SPI1_BASE_ADDRESS	108
5.39.2.2 SPI2_BASE_ADDRESS	108
5.39.2.3 SPI3_BASE_ADDRESS	108
5.39.2.4 SPI4_BASE_ADDRESS	109
5.40 SPI Registers	109
5.40.1 Detailed Description	109
5.40.2 Macro Definition Documentation	109
5.40.2.1 SPI1	109
5.40.2.2 SPI2	109
5.40.2.3 SPI3	110
5.41 SPI Interfacing Macros	110
5.41.1 Detailed Description	110
5.41.2 Macro Definition Documentation	110
5.41.2.1 SPIx_MSTR	110
5.41.2.2 SPIx_SLAVE	110
5.41.2.3 SPIx_SIMPLEX	111
5.41.2.4 SPIx_HALF_DUPLEX	111
5.41.2.5 SPIx_FULL_DUPLEX	111
5.42 Systick Configuration	111
5.42.1 Detailed Description	111
5.42.2 Macro Definition Documentation	111
5.42.2.1 CLK_SOURCE	112
5.42.2.2 Exception_Request	112
5.42.2.3 SINGLE_INTERVAL_MODE	112

5.42.2.4 PERIODIC_INTERVAL_MODE	112
5.42.2.5 MAX_TICKS	113
5.43 Delay Units	113
5.43.1 Detailed Description	113
5.43.2 Macro Definition Documentation	113
5.43.2.1 MILLI_SEC	113
5.43.2.2 MICRO_SEC	113
5.43.2.3 SEC	114
5.44 Systick Addresses	114
5.44.1 Detailed Description	114
5.44.2 Macro Definition Documentation	114
5.44.2.1 SysTick_BASE_ADDRESS	114
5.45 Systick Registers	114
5.45.1 Detailed Description	114
5.45.2 Macro Definition Documentation	114
5.45.2.1 SysTick	115
5.46 Systick Register Bits Positions	115
5.46.1 Detailed Description	115
5.46.2 Macro Definition Documentation	115
5.46.2.1 COUNTFLAG	115
5.46.2.2 CLKSOURCE	115
5.46.2.3 TICKINT	116
5.46.2.4 COUNTER_ENABLE	116
5.47 Systick Clock Sources	116
5.47.1 Detailed Description	116
5.47.2 Macro Definition Documentation	116
5.47.2.1 AHB_DividedBy8	117
5.47.2.2 AHB	117
5.48 Systick Exception (Interrupt) Status	117
5.48.1 Detailed Description	117
5.48.2 Macro Definition Documentation	117
5.48.2.1 Dont AssertRequest	117
5.48.2.2 AssertRequest	118
5.49 UART Addresses	118
5.49.1 Detailed Description	118
5.49.2 Macro Definition Documentation	118
5.49.2.1 USART1_BASE_ADDRESS	118
5.49.2.2 USART2_BASE_ADDRESS	118
5.49.2.3 USART6_BASE_ADDRESS	119
5.50 UART Registers	119
5.50.1 Detailed Description	119
5.50.2 Macro Definition Documentation	119

5.50.2.1 USART1_REG	119
5.50.2.2 USART2_REG	119
5.50.2.3 USART6_REG	120
5.51 UART Oversampling	120
5.51.1 Detailed Description	120
5.51.2 Macro Definition Documentation	120
5.51.2.1 OVER_SAMPLING_16	120
5.51.2.2 OVER_SAMPLING_8	120
5.52 UART Operating Modes	121
5.52.1 Detailed Description	121
5.52.2 Macro Definition Documentation	121
5.52.2.1 TX_ONLY	121
5.52.2.2 RX_ONLY	121
5.52.2.3 TX_RX	121
5.53 UART Parity	122
5.53.1 Detailed Description	122
5.53.2 Macro Definition Documentation	122
5.53.2.1 EVEN_PARITY	122
5.53.2.2 ODD_PARITY	122
5.54 UART Bit Sizes	122
5.54.1 Detailed Description	122
5.54.2 Macro Definition Documentation	122
5.54.2.1 MODE_8BIT	123
5.54.2.2 MODE_9BIT	123
5.55 UART Stop Bits	123
5.55.1 Detailed Description	123
5.55.2 Macro Definition Documentation	123
5.55.2.1 STOP_BIT_1	123
5.55.2.2 STOP_BIT_0_5	124
5.55.2.3 STOP_BIT_2	124
5.55.2.4 STOP_BIT_1_5	124
6 Data Structure Documentation	125
6.1 ADC_MemoryMapType Struct Reference	125
6.1.1 Detailed Description	126
6.1.2 Field Documentation	126
6.1.2.1 SR	126
6.1.2.2 CR1	126
6.1.2.3 CR2	127
6.1.2.4 SMPR1	127
6.1.2.5 SMPR2	127
6.1.2.6 JOFR1	127

6.1.2.7 JOFR2	127
6.1.2.8 JOFR3	128
6.1.2.9 JOFR4	128
6.1.2.10 HTR	128
6.1.2.11 LTR	128
6.1.2.12 SQR1	128
6.1.2.13 SQR2	129
6.1.2.14 SQR3	129
6.1.2.15 JSQR	129
6.1.2.16 JDR1	129
6.1.2.17 JDR2	129
6.1.2.18 JDR3	130
6.1.2.19 JDR4	130
6.1.2.20 DR	130
6.1.2.21 CCR	130
6.2 BUZZER_BuzzerConfiguration Struct Reference	130
6.2.1 Detailed Description	131
6.2.2 Field Documentation	131
6.2.2.1 u8Port	131
6.2.2.2 u8Pin	131
6.3 DCM_MotorConfiguration Struct Reference	132
6.3.1 Detailed Description	132
6.3.2 Field Documentation	132
6.3.2.1 u8Port	132
6.3.2.2 u8Pin1	133
6.3.2.3 u8Pin2	133
6.3.2.4 u8Direction	133
6.3.2.5 u8SpeedPin	133
6.3.2.6 u32SpeedRatio	134
6.4 EXTI_ConfigType Struct Reference	134
6.4.1 Detailed Description	134
6.4.2 Field Documentation	134
6.4.2.1 LineNum	134
6.4.2.2 PortNum	135
6.4.2.3 TriggerStatus	135
6.5 EXTI_Type Struct Reference	135
6.5.1 Detailed Description	136
6.5.2 Field Documentation	136
6.5.2.1 IMR	136
6.5.2.2 EMR	136
6.5.2.3 RTSR	136
6.5.2.4 FTSR	136

6.5.2.5 SWIER	137
6.5.2.6 PR	137
6.6 GPIOx_MemoryMapType Struct Reference	137
6.6.1 Detailed Description	138
6.6.2 Field Documentation	138
6.6.2.1 MODERx	138
6.6.2.2 OTYPERx	138
6.6.2.3 OSPEEDRx	138
6.6.2.4 PUPDRx	138
6.6.2.5 IDRx	139
6.6.2.6 ODRx	139
6.6.2.7 BSRRx	139
6.6.2.8 LCKRx	139
6.6.2.9 AFRLx	139
6.6.2.10 AFRHx	140
6.7 HULTSNC_ConfigType Struct Reference	140
6.7.1 Detailed Description	140
6.7.2 Field Documentation	140
6.7.2.1 u8Port	140
6.7.2.2 u8Pin	141
6.8 LED_LEDConfiguration Struct Reference	141
6.8.1 Detailed Description	141
6.8.2 Field Documentation	141
6.8.2.1 u8Port	141
6.8.2.2 u8Pin	142
6.9 MGPIOp_ConfigType Struct Reference	142
6.9.1 Detailed Description	142
6.9.2 Field Documentation	142
6.9.2.1 Port	143
6.9.2.2 Pin	143
6.9.2.3 Mode	143
6.9.2.4 OutputType	143
6.9.2.5 OutputSpeed	144
6.9.2.6 InputType	144
6.9.2.7 AF_Type	144
6.10 MSYSCFG_MemMap_t Struct Reference	144
6.10.1 Detailed Description	145
6.10.2 Field Documentation	145
6.10.2.1 MEMRMP	145
6.10.2.2 PMC	145
6.10.2.3 EXTICR	145
6.10.2.4 CMPCR	146

6.11 NVIC_MemoryMapType Struct Reference	146
6.11.1 Detailed Description	146
6.11.2 Field Documentation	146
6.11.2.1 ISERx	147
6.11.2.2 RESERVED0x	147
6.11.2.3 ICERx	147
6.11.2.4 RSERVED1x	148
6.11.2.5 ISPRx	148
6.11.2.6 RESERVED2x	148
6.11.2.7 ICPRx	149
6.11.2.8 RESERVED3x	149
6.11.2.9 IABRx	149
6.11.2.10 RESERVED4x	150
6.11.2.11 IPRx	150
6.12 RCC_MemoryMapType Struct Reference	150
6.12.1 Detailed Description	152
6.12.2 Field Documentation	152
6.12.2.1 CR	152
6.12.2.2 PLLCFG	152
6.12.2.3 CFGR	153
6.12.2.4 CIR	153
6.12.2.5 AHB1RSTR	153
6.12.2.6 AHB2RSTR	153
6.12.2.7 Reserved1	153
6.12.2.8 Reserved2	154
6.12.2.9 APB1RSTR	154
6.12.2.10 APB2RSTR	154
6.12.2.11 Reserved3	154
6.12.2.12 Reserved4	154
6.12.2.13 AHB1ENR	155
6.12.2.14 AHB2ENR	155
6.12.2.15 Reserved5	155
6.12.2.16 Reserved6	155
6.12.2.17 APB1ENR	155
6.12.2.18 APB2ENR	156
6.12.2.19 Reserved7	156
6.12.2.20 Reserved8	156
6.12.2.21 AHB1LPENR	156
6.12.2.22 AHB2LPENR	156
6.12.2.23 Reserved9	157
6.12.2.24 Reserved10	157
6.12.2.25 APB1LPENR	157

6.12.2.26 APB2LPENR	157
6.12.2.27 Reserved11	157
6.12.2.28 Reserved12	158
6.12.2.29 BDCR	158
6.12.2.30 CSR	158
6.12.2.31 Reserved13	158
6.12.2.32 Reserved14	158
6.12.2.33 SSCGR	159
6.12.2.34 PLLI2SCFGR	159
6.12.2.35 DCKCFGR	159
6.13 SCB_MemoryMapType Struct Reference	159
6.13.1 Detailed Description	160
6.13.2 Field Documentation	160
6.13.2.1 CPUID	161
6.13.2.2 ICSR	161
6.13.2.3 VTOR	161
6.13.2.4 AIRCR	161
6.13.2.5 SCR	162
6.13.2.6 CCR	162
6.13.2.7 SHPR1	162
6.13.2.8 SHPR2	163
6.13.2.9 SHPR3	163
6.13.2.10 SHCSR	163
6.13.2.11 CFSR	164
6.13.2.12 HFSR	164
6.13.2.13 RESERVED	164
6.13.2.14 MMFAR	165
6.13.2.15 BFAR	165
6.14 SPI_MemoryMapType Struct Reference	165
6.14.1 Detailed Description	166
6.14.2 Field Documentation	166
6.14.2.1 CR1	166
6.14.2.2 CR2	166
6.14.2.3 SR	166
6.14.2.4 DR	166
6.14.2.5 CRCPR	167
6.14.2.6 RXCRCR	167
6.14.2.7 TXCRCR	167
6.14.2.8 I2SCFGR	167
6.14.2.9 I2SPR	167
6.15 SysTick_Type Struct Reference	168
6.15.1 Detailed Description	168

6.15.2 Field Documentation	168
6.15.2.1 CTRL	168
6.15.2.2 LOAD	168
6.15.2.3 VAL	169
6.15.2.4 CALIB	169
6.16 Task Struct Reference	169
6.16.1 Detailed Description	169
6.16.2 Field Documentation	169
6.16.2.1 TaskHandler	169
6.16.2.2 periodicity	170
6.16.2.3 Task_RunningState	170
6.17 TIM1_MemoryMapType Struct Reference	170
6.17.1 Detailed Description	171
6.17.2 Field Documentation	171
6.17.2.1 CR1	171
6.17.2.2 CR2	171
6.17.2.3 SMCR	171
6.17.2.4 DIER	171
6.17.2.5 SR	171
6.17.2.6 EGR	172
6.17.2.7 CCMR1	172
6.17.2.8 CCMR2	172
6.17.2.9 CCER	172
6.17.2.10 CNT	172
6.17.2.11 PSC	172
6.17.2.12 ARR	173
6.17.2.13 RCR	173
6.17.2.14 CCR1	173
6.17.2.15 CCR2	173
6.17.2.16 CCR3	173
6.17.2.17 CCR4	173
6.17.2.18 BDTR	174
6.17.2.19 DCR	174
6.17.2.20 DMAR	174
6.18 USART_ClockInitTypeDef Struct Reference	174
6.18.1 Detailed Description	174
6.18.2 Field Documentation	175
6.18.2.1 ClockOutput	175
6.18.2.2 ClockPolarity	175
6.18.2.3 ClockPhase	175
6.18.2.4 LastBitClockPulse	175
6.19 USART_InitTypeDef Struct Reference	176

6.19.1 Detailed Description	176
6.19.2 Field Documentation	176
6.19.2.1 BaudRate	176
6.19.2.2 DataWidth	177
6.19.2.3 StopBits	177
6.19.2.4 Parity_Enable	177
6.19.2.5 Parity_Selection	177
6.19.2.6 TransferDirection	178
6.19.2.7 HardwareFlowControl	178
6.19.2.8 Oversampling	178
6.20 USART_MemoryMapType Struct Reference	178
6.20.1 Detailed Description	179
6.20.2 Field Documentation	179
6.20.2.1 SR_REG	179
6.20.2.2 DR_REG	179
6.20.2.3 BRR_REG	180
6.20.2.4 CR1_REG	180
6.20.2.5 CR2_REG	180
6.20.2.6 CR3_REG	180
6.20.2.7 GTPR_REG	180
7 File Documentation	181
7.1 ACC_config.h	181
7.2 ACC_interface.h	181
7.3 ACC_private.h	181
7.4 ACC_program.c	182
7.5 Exit_State_config.h	184
7.6 Exit_State_interface.h	184
7.7 Exit_State_private.h	184
7.8 Exit_State_program.c	185
7.9 FCW_config.h	186
7.10 FCW_interface.h	186
7.11 FCW_private.h	186
7.12 FCW_program.c	186
7.13 Mob_APP_config.h	189
7.14 Mob_APP_interface.h	190
7.15 Mob_APP_private.h	190
7.16 Mob_APP_program.c	190
7.17 NCC_config.h	192
7.18 NCC_interface.h	192
7.19 NCC_private.h	193
7.20 NCC_program.c	193

7.21 Traditional_Mode_config.h	195
7.22 Traditional_Mode_interface.h	195
7.23 Traditional_Mode_private.h	195
7.24 Traditional_Mode_program.c	195
7.25 Bluetooth_config.c	197
7.26 Bluetooth_config.h	198
7.27 Bluetooth_interface.h	198
7.28 Bluetooth_private.h	199
7.29 Bluetooth_program.c	199
7.30 BUZZER_config.h	202
7.31 COTS/HAL/BUZZER/BUZZER_interface.h File Reference	202
7.31.1 Detailed Description	203
7.31.2 Function Documentation	203
7.31.2.1 HBUZZER_vInit()	203
7.31.2.2 HBUZZER_vSoundOn()	204
7.31.2.3 HBUZZER_vSoundOff()	204
7.31.2.4 HBUZZER_vToggleSound()	204
7.32 BUZZER_interface.h	205
7.33 BUZZER_private.h	205
7.34 COTS/HAL/BUZZER/BUZZER_program.c File Reference	205
7.34.1 Detailed Description	206
7.34.2 Function Documentation	206
7.34.2.1 HBUZZER_vInit()	206
7.34.2.2 HBUZZER_vSoundOn()	207
7.34.2.3 HBUZZER_vSoundOff()	207
7.34.2.4 HBUZZER_vToggleSound()	208
7.35 BUZZER_program.c	208
7.36 Car_Movement_config.h	209
7.37 Car_Movement_interface.h	209
7.38 Car_Movement_private.h	210
7.39 Car_Movement_program.c	210
7.40 COTS/HAL/DCMOTOR/DCM_config.h File Reference	212
7.40.1 Detailed Description	212
7.40.2 Macro Definition Documentation	212
7.40.2.1 MAX_SPEED	212
7.41 DCM_config.h	213
7.42 COTS/HAL/DCMOTOR/DCM_interface.h File Reference	213
7.42.1 Detailed Description	214
7.42.2 Function Documentation	214
7.42.2.1 HDCM_vInitMotor()	214
7.42.2.2 HDCM_vMoveForward()	215
7.42.2.3 HDCM_vMoveBackward()	216

7.42.2.4 HDCM_vStopMotor()	216
7.42.2.5 HDCM_vMotorSpeedCntrl()	217
7.42.2.6 HDCM_vGetSpeedValue()	217
7.43 DCM_interface.h	218
7.44 COTS/HAL/DCMOTOR/DCM_private.h File Reference	219
7.44.1 Detailed Description	219
7.45 DCM_private.h	220
7.46 COTS/HAL/DCMOTOR/DCM_program.c File Reference	220
7.46.1 Detailed Description	220
7.46.2 Function Documentation	221
7.46.2.1 HDCM_vInitMotor()	221
7.46.2.2 HDCM_vMoveForward()	222
7.46.2.3 HDCM_vMoveBackward()	222
7.46.2.4 HDCM_vStopMotor()	223
7.46.2.5 HDCM_vMotorSpeedCntrl()	223
7.46.2.6 HDCM_vGetSpeedValue()	224
7.47 DCM_program.c	225
7.48 LCD_config.h	227
7.49 LCD_interface.h	228
7.50 LCD_private.h	229
7.51 LCD_program.c	230
7.52 LDR_config.c	236
7.53 LDR_config.h	237
7.54 LDR_interface.h	237
7.55 LDR_private.h	238
7.56 LDR_program.c	238
7.57 LED_config.h	239
7.58 LED_interface.h	239
7.59 LED_private.h	240
7.60 LED_program.c	240
7.61 TFT_Config.c	241
7.62 TFT_config.h	242
7.63 TFT_interface.h	242
7.64 TFT_private.h	243
7.65 TFT_program.c	243
7.66 UltraSonic_config.h	246
7.67 COTS/HAL/UltraSonic/UltraSonic_interface.h File Reference	246
7.67.1 Detailed Description	247
7.67.2 Function Documentation	247
7.67.2.1 HULTSNC_vInit()	247
7.67.2.2 HULTSNC_vTrigger()	248
7.67.2.3 HULTSNC_f32GetDistance()	248

7.68 UltraSonic_interface.h	249
7.69 UltraSonic_private.h	249
7.70 UltraSonic_program.c	250
7.71 COTS/LIB/LSTD_BITMATH.h File Reference	251
7.71.1 Detailed Description	251
7.72 LSTD_BITMATH.h	252
7.73 COTS/LIB/LSTD_COMPILER.h File Reference	252
7.73.1 Detailed Description	252
7.74 LSTD_COMPILER.h	253
7.75 COTS/LIB/LSTD MCU UTILITIES.h File Reference	253
7.75.1 Detailed Description	253
7.76 LSTD MCU UTILITIES.h	254
7.77 COTS/LIB/LSTD TYPES.h File Reference	254
7.77.1 Detailed Description	254
7.78 LSTD TYPES.h	255
7.79 COTS/LIB/LSTD VALUES.h File Reference	255
7.79.1 Detailed Description	255
7.80 LSTD VALUES.h	256
7.81 ADC_config.h	257
7.82 COTS/MCAL/ADC/ADC_interface.h File Reference	264
7.82.1 Detailed Description	265
7.82.2 Function Documentation	265
7.82.2.1 MADC_vInit()	266
7.82.2.2 MADC_u16ConvertToDigital()	269
7.82.2.3 MADC_vDisable()	270
7.82.2.4 MADC_vEnable()	270
7.82.2.5 MADC_vRegINT_Disable()	271
7.82.2.6 MADC_vRegINTEnable()	271
7.82.2.7 MADC_vSelectChannel()	271
7.82.2.8 MADC_vSetCallBack()	271
7.83 ADC_interface.h	272
7.84 ADC_private.h	273
7.85 ADC_program.c	278
7.86 EXTI_config.h	284
7.87 COTS/MCAL/EXTI/EXTI_interface.h File Reference	286
7.87.1 Detailed Description	288
7.87.2 Function Documentation	288
7.87.2.1 MEXTI_vInit()	288
7.87.2.2 MEXTI_vInit_WithStruct()	291
7.87.2.3 MEXTI_vEnableLine()	291
7.87.2.4 MEXTI_vDisableLine()	292
7.87.2.5 MEXTI_vSWITrigger()	292

7.87.2.6 MEXTI_vSetTrigger()	293
7.87.2.7 MEXTI_vSetCallback()	293
7.87.2.8 MSYSCFG_vSetEXTIPort()	294
7.88 EXTI_interface.h	294
7.89 COTS/MCAL/EXTI/EXTI_private.h File Reference	295
7.89.1 Detailed Description	296
7.89.2 Macro Definition Documentation	296
7.89.2.1 EXTI IRQs	296
7.90 EXTI_private.h	296
7.91 COTS/MCAL/EXTI/EXTI_program.c File Reference	297
7.91.1 Detailed Description	298
7.91.2 Function Documentation	298
7.91.2.1 MSYSCFG_vSetEXTIPort()	298
7.91.2.2 MEXTI_vInit()	299
7.91.2.3 MEXTI_vInit_WithStruct()	301
7.91.2.4 MEXTI_vEnableLine()	302
7.91.2.5 MEXTI_vDisableLine()	302
7.91.2.6 MEXTI_vSWITTrigger()	303
7.91.2.7 MEXTI_vSetTrigger()	303
7.91.2.8 MEXTI_vSetCallback()	304
7.91.2.9 EXTI0_IRQHandler()	304
7.91.2.10 EXTI1_IRQHandler()	305
7.91.2.11 EXTI2_IRQHandler()	305
7.91.2.12 EXTI3_IRQHandler()	305
7.91.2.13 EXTI4_IRQHandler()	306
7.91.2.14 EXTI9_5_IRQHandler()	306
7.91.2.15 EXTI15_10_IRQHandler()	306
7.92 EXTI_program.c	307
7.93 COTS/MCAL/EXTI/SYSCFG_private.h File Reference	312
7.93.1 Detailed Description	312
7.94 SYSCFG_private.h	313
7.95 COTS/MCAL/GPIO/GPIO_config.h File Reference	313
7.95.1 Detailed Description	313
7.95.2 Macro Definition Documentation	314
7.95.2.1 GPIOA_PIN_POS	314
7.95.2.2 GPIOB_PIN_POS	314
7.95.2.3 LCKK_BIT_POS	314
7.95.2.4 PORTA_BIT_MANIPULATION	314
7.95.2.5 PORTB_BIT_MANIPULATION	314
7.96 GPIO_config.h	315
7.97 COTS/MCAL/GPIO/GPIO_interface.h File Reference	315
7.97.1 Detailed Description	318

7.97.2 Macro Definition Documentation	318
7.97.2.1 GPIOA_LOW_NIBBLE_HIGH	318
7.97.2.2 GPIOA_LOW_NIBBLE_LOW	319
7.97.3 Function Documentation	319
7.97.3.1 MGPOx_vLockedPins()	319
7.97.3.2 MGPOx_vSetPinMode()	319
7.97.3.3 MGPOx_vSetPinOutputType()	320
7.97.3.4 MGPOx_vSetPinOutputSpeed()	321
7.97.3.5 MGPOx_vSetPinInputPullType()	322
7.97.3.6 MGPOx_u8GetPinValue()	322
7.97.3.7 MGPOx_vSetPinValue()	323
7.97.3.8 MGPOx_vSetResetAtomic()	324
7.97.3.9 MGPOx_vSetAlternateFunctionON()	325
7.97.3.10 MGPOx_vSetPortConfigLock()	327
7.97.3.11 MGPOx_vInit()	327
7.97.3.12 MGPOx_vTogglePinValue()	328
7.97.3.13 GPIO_vSetNibbleLowValue()	328
7.98 GPIO_interface.h	329
7.99 COTS/MCAL/GPIO/GPIO_private.h File Reference	331
7.99.1 Detailed Description	331
7.100 GPIO_private.h	332
7.101 COTS/MCAL/GPIO/GPIO_program.c File Reference	332
7.101.1 Detailed Description	333
7.101.2 Function Documentation	333
7.101.2.1 MGPOx_vLockedPins()	334
7.101.2.2 MGPOx_vSetPinMode()	334
7.101.2.3 MGPOx_vSetPinOutputType()	335
7.101.2.4 MGPOx_vSetPinOutputSpeed()	336
7.101.2.5 MGPOx_vSetPinInputPullType()	337
7.101.2.6 MGPOx_u8GetPinValue()	337
7.101.2.7 MGPOx_vSetPinValue()	338
7.101.2.8 MGPOx_vSetResetAtomic()	339
7.101.2.9 MGPOx_vSetAlternateFunctionON()	340
7.101.2.10 MGPOx_vInit()	342
7.101.2.11 MGPOx_vTogglePinValue()	342
7.101.2.12 GPIO_vSetNibbleLowValue()	343
7.102 GPIO_program.c	343
7.103 NVIC_config.h	349
7.104 COTS/MCAL/NVIC/NVIC_interface.h File Reference	349
7.104.1 Detailed Description	354
7.104.2 Function Documentation	354
7.104.2.1 MNVIC_vEnablePeriphral()	354

7.104.2.2 MNVIC_vDisablePeriphral()	355
7.104.2.3 MNVIC_vSetPendingFlag()	355
7.104.2.4 MNVIC_vClearPendingFlag()	356
7.104.2.5 MNVIC_u8GetActive()	356
7.104.2.6 MNVIC_vSetPriorityConfig()	357
7.104.2.7 MNVIC_vSetPriority()	357
7.104.2.8 NVIC_GetPriority()	358
7.105 NVIC_interface.h	359
7.106 COTS/MCAL/NVIC/NVIC_private.h File Reference	360
7.106.1 Detailed Description	361
7.106.2 Macro Definition Documentation	361
7.106.2.1 MNVIC_STIR	361
7.106.2.2 VECTKEY_PASSWORD	362
7.107 NVIC_private.h	362
7.108 COTS/MCAL/NVIC/NVIC_program.c File Reference	363
7.108.1 Detailed Description	363
7.108.2 Macro Definition Documentation	364
7.108.2.1 REGISTER_SIZE	364
7.108.3 Function Documentation	364
7.108.3.1 MNVIC_vEnablePeriphral()	364
7.108.3.2 MNVIC_vDisablePeriphral()	364
7.108.3.3 MNVIC_vSetPendingFlag()	365
7.108.3.4 MNVIC_vClearPendingFlag()	365
7.108.3.5 MNVIC_u8GetActive()	366
7.108.3.6 MNVIC_vSetPriorityConfig()	366
7.108.3.7 MNVIC_vSetPriority()	367
7.108.3.8 NVIC_GetPriority()	368
7.109 NVIC_program.c	369
7.110 COTS/MCAL/RCC/MRCC_config.h File Reference	371
7.110.1 Detailed Description	372
7.110.2 Macro Definition Documentation	372
7.110.2.1 PLLI2S	372
7.110.2.2 PLL	373
7.110.2.3 CSS	373
7.110.2.4 HSEBYP	373
7.110.2.5 HSE_EN	373
7.110.2.6 HSI_EN	373
7.110.2.7 PLLQ	373
7.110.2.8 PLLSRC	374
7.110.2.9 PLLP	374
7.110.2.10 PLLN	374
7.110.2.11 PLLM	374

7.110.2.12 MCO2	374
7.110.2.13 MCO2PRE	374
7.110.2.14 MCO1PRE	375
7.110.2.15 I2SSRC	375
7.110.2.16 MCO1	375
7.110.2.17 RTCPRE	375
7.110.2.18 PPREG2	375
7.110.2.19 PPREG1	375
7.110.2.20 HPRE	376
7.110.2.21 SWS	376
7.110.2.22 SW	376
7.110.2.23 DMA2CLK	376
7.110.2.24 DMA1CLK	376
7.110.2.25 CRCCLK	376
7.110.2.26 GPIOHCLK	377
7.110.2.27 GPIOECLK	377
7.110.2.28 GPIODCLK	377
7.110.2.29 GPIOCCLK	377
7.110.2.30 GPIOBCLK	377
7.110.2.31 GPIOACLK	377
7.110.2.32 OTGFS	378
7.110.2.33 PWREN	378
7.110.2.34 I2C3EN	378
7.110.2.35 I2C2EN	378
7.110.2.36 I2C1EN	378
7.110.2.37 USART2EN	378
7.110.2.38 SPI3EN	379
7.110.2.39 SPI2EN	379
7.110.2.40 WWDGEN	379
7.110.2.41 TIM5EN	379
7.110.2.42 TIM4EN	379
7.110.2.43 TIM3EN	379
7.110.2.44 TIM2EN	380
7.110.2.45 TIM11EN	380
7.110.2.46 TIM10EN	380
7.110.2.47 TIM9EN	380
7.110.2.48 SYSCFGEN	380
7.110.2.49 SPI4EN	380
7.110.2.50 SPI1EN	381
7.110.2.51 SDIOEN	381
7.110.2.52 ADC1EN	381
7.110.2.53 USART6EN	381

7.110.2.54 USART1EN	381
7.110.2.55 TIM1EN	381
7.111 MRCC_config.h	382
7.112 COTS/MCAL/RCC/MRCC_interface.h File Reference	389
7.112.1 Detailed Description	391
7.112.2 Function Documentation	392
7.112.2.1 MRCC_vInit()	392
7.112.2.2 MRCC_vEnablePeriphralCLK()	396
7.112.2.3 MRCC_vDisablePeriphralCLK()	397
7.113 MRCC_interface.h	398
7.114 COTS/MCAL/RCC/MRCC_private.h File Reference	399
7.114.1 Detailed Description	404
7.114.2 Macro Definition Documentation	404
7.114.2.1 RCC_BASE_ADDRESS	404
7.114.2.2 RCC	404
7.114.2.3 RCC_CR_PLLI2SRDY	405
7.114.2.4 RCC_CR_PLLI2SON	405
7.114.2.5 RCC_CR_PLLRDY	405
7.114.2.6 RCC_CR_PLLON	405
7.114.2.7 RCC_CR_CSSON	405
7.114.2.8 RCC_CR_HSEBYP	405
7.114.2.9 RCC_CR_HSERDY	406
7.114.2.10 RCC_CR_HSEON	406
7.114.2.11 RCC_CR_HSIRDY	406
7.114.2.12 RCC_CR_HSION	406
7.114.2.13 RCC_PLLCFGR_PLLQ_b0	406
7.114.2.14 RCC_PLLCFGR_PLLQ_b1	406
7.114.2.15 RCC_PLLCFGR_PLLQ_b2	407
7.114.2.16 RCC_PLLCFGR_PLLQ_b3	407
7.114.2.17 RCC_PLLCFGR_PLLSRC	407
7.114.2.18 RCC_PLLCFGR_PLLP_b0	407
7.114.2.19 RCC_PLLCFGR_PLLP_b1	407
7.114.2.20 RCC_PLLCFGR_PLLN_b0	407
7.114.2.21 RCC_PLLCFGR_PLLN_b1	408
7.114.2.22 RCC_PLLCFGR_PLLN_b2	408
7.114.2.23 RCC_PLLCFGR_PLLN_b3	408
7.114.2.24 RCC_PLLCFGR_PLLN_b4	408
7.114.2.25 RCC_PLLCFGR_PLLN_b5	408
7.114.2.26 RCC_PLLCFGR_PLLN_b6	408
7.114.2.27 RCC_PLLCFGR_PLLN_b7	409
7.114.2.28 RCC_PLLCFGR_PLLN_b8	409
7.114.2.29 RCC_PLLCFGR_PLLM_b0	409

7.114.2.30 RCC_PLLCFGR_PLLM_b1	409
7.114.2.31 RCC_PLLCFGR_PLLM_b2	409
7.114.2.32 RCC_PLLCFGR_PLLM_b3	409
7.114.2.33 RCC_PLLCFGR_PLLM_b4	410
7.114.2.34 RCC_PLLCFGR_PLLM_b5	410
7.114.2.35 RCC_CFGR_MOC2_b0	410
7.114.2.36 RCC_CFGR_MOC2_b1	410
7.114.2.37 RCC_CFGR_MOC2PRE_b0	410
7.114.2.38 RCC_CFGR_MOC2PRE_b1	410
7.114.2.39 RCC_CFGR_MOC2PRE_b2	411
7.114.2.40 RCC_CFGR_MOC1PRE_b0	411
7.114.2.41 RCC_CFGR_MOC1PRE_b1	411
7.114.2.42 RCC_CFGR_MOC1PRE_b2	411
7.114.2.43 RCC_CFGR_I2SSRC	411
7.114.2.44 RCC_CFGR_MOC1_b0	411
7.114.2.45 RCC_CFGR_MOC1_b1	412
7.114.2.46 RCC_CFGR_RTCPRE_b0	412
7.114.2.47 RCC_CFGR_RTCPRE_b1	412
7.114.2.48 RCC_CFGR_RTCPRE_b2	412
7.114.2.49 RCC_CFGR_RTCPRE_b3	412
7.114.2.50 RCC_CFGR_RTCPRE_b4	412
7.114.2.51 RCC_CFGR_PPREG2_b0	413
7.114.2.52 RCC_CFGR_PPREG2_b1	413
7.114.2.53 RCC_CFGR_PPREG2_b2	413
7.114.2.54 RCC_CFGR_PPREG1_b0	413
7.114.2.55 RCC_CFGR_PPREG1_b1	413
7.114.2.56 RCC_CFGR_PPREG1_b2	413
7.114.2.57 RCC_CFGR_HPRE_b0	414
7.114.2.58 RCC_CFGR_HPRE_b1	414
7.114.2.59 RCC_CFGR_HPRE_b2	414
7.114.2.60 RCC_CFGR_HPRE_b3	414
7.114.2.61 RCC_CFGR_SWS_b0	414
7.114.2.62 RCC_CFGR_SWS_b1	414
7.114.2.63 RCC_CFGR_SW_b0	415
7.114.2.64 RCC_CFGR_SW_b1	415
7.114.2.65 RCC_AHB1ENR_DMA2EN	415
7.114.2.66 RCC_AHB1ENR_DMA1EN	415
7.114.2.67 RCC_AHB1ENR_CRCEN	415
7.114.2.68 RCC_AHB1ENR_GPIOHEN	415
7.114.2.69 RCC_AHB1ENR_GPIOEEN	416
7.114.2.70 RCC_AHB1ENR_GPIODEN	416
7.114.2.71 RCC_AHB1ENR_GPIOCEN	416

7.114.2.72 RCC_AHB1ENR_GPIOBEN	416
7.114.2.73 RCC_AHB1ENR_GPIOAEN	416
7.114.2.74 RCC_AHB2ENR_OTGFSEN	416
7.114.2.75 RCC_APB1ENR_PWREN	417
7.114.2.76 RCC_APB1ENR_I2C3EN	417
7.114.2.77 RCC_APB1ENR_I2C2EN	417
7.114.2.78 RCC_APB1ENR_I2C1EN	417
7.114.2.79 RCC_APB1ENR_USART2EN	417
7.114.2.80 RCC_APB1ENR_SPI3EN	417
7.114.2.81 RCC_APB1ENR_SPI2EN	418
7.114.2.82 RCC_APB1ENR_WWDGEN	418
7.114.2.83 RCC_APB1ENR_TIM5EN	418
7.114.2.84 RCC_APB1ENR_TIM4EN	418
7.114.2.85 RCC_APB1ENR_TIM3EN	418
7.114.2.86 RCC_APB1ENR_TIM2EN	418
7.114.2.87 RCC_APB2ENR_TIM11EN	419
7.114.2.88 RCC_APB2ENR_TIM10EN	419
7.114.2.89 RCC_APB2ENR_TIM9EN	419
7.114.2.90 RCC_APB2ENR_SYSCFGEN	419
7.114.2.91 RCC_APB2ENR_SPI4EN	419
7.114.2.92 RCC_APB2ENR_SPI1EN	419
7.114.2.93 RCC_APB2ENR_SDIOEN	420
7.114.2.94 RCC_APB2ENR_ADC1EN	420
7.114.2.95 RCC_APB2ENR_USART6EN	420
7.114.2.96 RCC_APB2ENR_USART1EN	420
7.114.2.97 RCC_APB2ENR_TIM1EN	420
7.114.2.98 RCC_AHB1LPENR_FLITFLPEN	420
7.114.2.99 ENABLE	421
7.114.2.100 DISABLE	421
7.114.2.101 BYBASED	421
7.114.2.102 NOTBYBASED	421
7.114.2.103 SYSCLK	421
7.114.2.104 PLLI2SCLK	421
7.114.2.105 HSE	422
7.114.2.106 PLLCLK [1/2]	422
7.114.2.107 NoDivision	422
7.114.2.108 DivisionBy2	422
7.114.2.109 DivisionBy3	422
7.114.2.110 DivisionBy4	422
7.114.2.111 DivisionBy5	423
7.114.2.112 I2S_CKIN	423
7.114.2.113 HSI	423

7.114.2.114 LSE	423
7.114.2.115 PLLCLK [2/2]	423
7.114.2.116 SYSCLKby2	423
7.114.2.117 SYSCLKby4	424
7.114.2.118 SYSCLKby8	424
7.114.2.119 SYSCLKby16	424
7.114.2.120 SYSCLKby64	424
7.114.2.121 SYSCLKby128	424
7.114.2.122 SYSCLKby256	424
7.114.2.123 SYSCLKby512	425
7.114.2.124 NoCLK0	425
7.114.2.125 NoCLK1	425
7.114.2.126 HSEby2	425
7.114.2.127 HSEby3	425
7.114.2.128 HSEby4	425
7.114.2.129 HSEby5	426
7.114.2.130 HSEby6	426
7.114.2.131 HSEby7	426
7.114.2.132 HSEby8	426
7.114.2.133 HSEby9	426
7.114.2.134 HSEby10	426
7.114.2.135 HSEby11	427
7.114.2.136 HSEby12	427
7.114.2.137 HSEby13	427
7.114.2.138 HSEby14	427
7.114.2.139 HSEby15	427
7.114.2.140 HSEby16	427
7.114.2.141 HSEby17	428
7.114.2.142 HSEby18	428
7.114.2.143 HSEby19	428
7.114.2.144 HSEby20	428
7.114.2.145 HSEby21	428
7.114.2.146 HSEby22	428
7.114.2.147 HSEby23	429
7.114.2.148 HSEby24	429
7.114.2.149 HSEby25	429
7.114.2.150 HSEby26	429
7.114.2.151 HSEby27	429
7.114.2.152 HSEby28	429
7.114.2.153 HSEby29	430
7.114.2.154 HSEby30	430
7.114.2.155 HSEby31	430

7.114.2.156 AHBby2	430
7.114.2.157 AHBby4	430
7.114.2.158 AHBby8	430
7.114.2.159 AHBby16	431
7.114.2.160 Equal_0	431
7.114.2.161 Equal_1	431
7.114.2.162 Equal_2	431
7.114.2.163 Equal_3	431
7.114.2.164 Equal_4	431
7.114.2.165 Equal_5	432
7.114.2.166 Equal_6	432
7.114.2.167 Equal_7	432
7.114.2.168 Equal_8	432
7.114.2.169 Equal_9	432
7.114.2.170 Equal_10	432
7.114.2.171 Equal_11	433
7.114.2.172 Equal_12	433
7.114.2.173 Equal_13	433
7.114.2.174 Equal_14	433
7.114.2.175 Equal_15	433
7.114.2.176 Equal_16	433
7.114.2.177 Equal_17	434
7.114.2.178 Equal_18	434
7.114.2.179 Equal_19	434
7.114.2.180 Equal_20	434
7.114.2.181 Equal_21	434
7.114.2.182 Equal_22	434
7.114.2.183 Equal_23	435
7.114.2.184 Equal_24	435
7.114.2.185 Equal_25	435
7.114.2.186 Equal_26	435
7.114.2.187 Equal_27	435
7.114.2.188 Equal_28	435
7.114.2.189 Equal_29	436
7.114.2.190 Equal_30	436
7.114.2.191 Equal_31	436
7.114.2.192 Equal_32	436
7.114.2.193 Equal_33	436
7.114.2.194 Equal_34	436
7.114.2.195 Equal_35	437
7.114.2.196 Equal_36	437
7.114.2.197 Equal_37	437

7.114.2.198 Equal_38	437
7.114.2.199 Equal_39	437
7.114.2.200 Equal_40	437
7.114.2.201 Equal_41	438
7.114.2.202 Equal_42	438
7.114.2.203 Equal_43	438
7.114.2.204 Equal_44	438
7.114.2.205 Equal_45	438
7.114.2.206 Equal_46	438
7.114.2.207 Equal_47	439
7.114.2.208 Equal_48	439
7.114.2.209 Equal_49	439
7.114.2.210 Equal_50	439
7.114.2.211 Equal_51	439
7.114.2.212 Equal_52	439
7.114.2.213 Equal_53	440
7.114.2.214 Equal_54	440
7.114.2.215 Equal_55	440
7.114.2.216 Equal_56	440
7.114.2.217 Equal_57	440
7.114.2.218 Equal_58	440
7.114.2.219 Equal_59	441
7.114.2.220 Equal_60	441
7.114.2.221 Equal_61	441
7.114.2.222 Equal_62	441
7.114.2.223 Equal_63	441
7.115 MRCC_private.h	442
7.116 COTS/MCAL/RCC/MRCC_program.c File Reference	446
7.116.1 Detailed Description	446
7.116.2 Function Documentation	447
7.116.2.1 MRCC_vInit()	447
7.116.2.2 MRCC_vEnablePeriphralCLK()	451
7.116.2.3 MRCC_vDisablePeriphralCLK()	452
7.117 MRCC_program.c	453
7.118 SPI_config.h	459
7.119 SPI_interface.h	462
7.120 SPI_private.h	463
7.121 SPI_program.c	465
7.122 COTS/MCAL/SysTick/SysTick_config.h File Reference	473
7.122.1 Detailed Description	473
7.123 SysTick_config.h	473
7.124 COTS/MCAL/SysTick/SysTick_interface.h File Reference	474

7.124.1 Detailed Description	475
7.124.2 Macro Definition Documentation	475
7.124.2.1 BUSY_TICK_TIME	475
7.124.2.2 SINGLE_INTERVAL_TICK_TIME	475
7.124.2.3 PERIODIC_INTERVAL_TICK_TIME	475
7.124.3 Function Documentation	476
7.124.3.1 MSysTick_vInit()	476
7.124.3.2 MSysTick_vSetBusyWait()	476
7.124.3.3 MSysTick_vDelay()	477
7.124.3.4 MSysTick_vDelayMicroSec()	478
7.124.3.5 MSysTick_vDelayMilliSec()	478
7.124.3.6 MSysTick_vDelaySec()	479
7.124.3.7 MSysTick_vSetSingleInterval()	480
7.124.3.8 MSysTick_vSetPeriodicInterval()	481
7.124.3.9 MSysTick_vStopInterval()	481
7.124.3.10 MSysTick_u32GetElapsedTime()	482
7.124.3.11 MSysTick_u32GetRemainingTime()	482
7.124.3.12 MSysTick_vEnable()	483
7.124.3.13 MSysTick_vDisable()	483
7.124.3.14 MSysTick_vEnableException()	483
7.124.3.15 MSysTick_vDisableException()	484
7.125 SysTick_interface.h	484
7.126 COTS/MCAL/SysTick/SysTick_private.h File Reference	485
7.126.1 Detailed Description	485
7.127 SysTick_private.h	486
7.128 COTS/MCAL/SysTick/SysTick_program.c File Reference	486
7.128.1 Detailed Description	487
7.128.2 Function Documentation	487
7.128.2.1 MSysTick_vInit()	488
7.128.2.2 MSysTick_vSetBusyWait()	488
7.128.2.3 MSysTick_vDelay()	489
7.128.2.4 MSysTick_vDelayMicroSec()	490
7.128.2.5 MSysTick_vDelayMilliSec()	490
7.128.2.6 MSysTick_vDelaySec()	491
7.128.2.7 MSysTick_vSetSingleInterval()	492
7.128.2.8 MSysTick_vSetPeriodicInterval()	492
7.128.2.9 MSysTick_vStopInterval()	493
7.128.2.10 MSysTick_u32GetElapsedTime()	494
7.128.2.11 MSysTick_u32GetRemainingTime()	494
7.128.2.12 MSysTick_vEnable()	494
7.128.2.13 MSysTick_vDisable()	495
7.128.2.14 MSysTick_vDisableException()	495

7.128.2.15 MSysTick_vEnableException()	495
7.128.2.16 SysTick_Handler()	496
7.129 SysTick_program.c	496
7.130 TIM1_config.h	501
7.131 TIM1_interface.h	510
7.132 TIM1_private.h	511
7.133 TIM1_program.c	517
7.134 UART_config.h	540
7.135 COTS/MCAL/UART/UART_interface.h File Reference	542
7.135.1 Detailed Description	543
7.135.2 Macro Definition Documentation	544
7.135.2.1 ENABLE	544
7.135.2.2 DISABLE	544
7.135.2.3 __BAUDRATE__	544
7.135.3 Function Documentation	544
7.135.3.1 MUSART_vInit()	544
7.135.3.2 MUSART_vEnable()	546
7.135.3.3 MUSART_vDisable()	547
7.135.3.4 MUSART_vTransmitByte()	547
7.135.3.5 MUSART_vTransmitString()	548
7.135.3.6 MUSART_u8ReceiveByteSynchNonBlocking()	548
7.135.3.7 MUSART_u8ReceiveByteSynchBlocking()	549
7.135.3.8 MUSART_ptrReceiveStringSynchNonBlocking()	550
7.135.3.9 MUSART_vRecieveString()	550
7.135.3.10 MUSART_u8CompareString()	551
7.135.3.11 MUSART_u8ReadDataRegister()	552
7.135.3.12 MUSART_vClearFlags()	552
7.135.3.13 MUSART_vRxIntSetStatus()	552
7.135.3.14 MUSART1_vSetCallBack()	553
7.135.3.15 MUSART2_vSetCallBack()	553
7.135.3.16 MUSART6_vSetCallBack()	554
7.136 UART_interface.h	554
7.137 COTS/MCAL/UART/UART_private.h File Reference	556
7.137.1 Detailed Description	557
7.137.2 Macro Definition Documentation	557
7.137.2.1 UART_DIV_SAMPLING16	558
7.137.2.2 UART_DIVMANT_SAMPLING16	558
7.137.2.3 UART_DIVFRAQ_SAMPLING16	558
7.137.2.4 UART_BRR_SAMPLING16	558
7.137.2.5 UART_DIV_SAMPLING8	559
7.137.2.6 UART_DIVMANT_SAMPLING8	559
7.137.2.7 UART_DIVFRAQ_SAMPLING8	559

7.137.2.8 UART_BRR_SAMPLING	559
7.137.2.9 MUSART_SR_PE_BIT	559
7.137.2.10 MUSART_SR_FE_BIT	560
7.137.2.11 MUSART_SR_NE_BIT	560
7.137.2.12 MUSART_SR_ORE_BIT	560
7.137.2.13 MUSART_SR_IDLE_BIT	560
7.137.2.14 MUSART_SR_RXNE_BIT	560
7.137.2.15 MUSART_SR_TC_BIT	560
7.137.2.16 MUSART_SR_TXE_BIT	561
7.137.2.17 MUSART_SR_LBD_BIT	561
7.137.2.18 MUSART_SR_CTS_BIT	561
7.137.2.19 MUSART_CR1_SBK_BIT	561
7.137.2.20 MUSART_CR1_RWU_BIT	561
7.137.2.21 MUSART_CR1_RE_BIT	561
7.137.2.22 MUSART_CR1_TE_BIT	562
7.137.2.23 MUSART_CR1_IDLEIE_BIT	562
7.137.2.24 MUSART_CR1_RXNEIE_BIT	562
7.137.2.25 MUSART_CR1_TCIE_BIT	562
7.137.2.26 MUSART_CR1_TXEIE_BIT	562
7.137.2.27 MUSART_CR1_PEIE_BIT	562
7.137.2.28 MUSART_CR1_PS_BIT	563
7.137.2.29 MUSART_CR1_PCE_BIT	563
7.137.2.30 MUSART_CR1_WAKE_BIT	563
7.137.2.31 MUSART_CR1_M_BIT	563
7.137.2.32 MUSART_CR1_UE_BIT	563
7.137.2.33 MUSART_CR1_OVER8_BIT	563
7.137.2.34 MUSART_CR2_ADD0_BIT	564
7.137.2.35 MUSART_CR2_ADD1_BIT	564
7.137.2.36 MUSART_CR2_ADD2_BIT	564
7.137.2.37 MUSART_CR2_ADD3_BIT	564
7.137.2.38 MUSART_CR2_LBDL_BIT	564
7.137.2.39 MUSART_CR2_LBDIE_BIT	564
7.137.2.40 MUSART_CR2_LBCL_BIT	565
7.137.2.41 MUSART_CR2_CPHA_BIT	565
7.137.2.42 MUSART_CR2_CPOL_BIT	565
7.137.2.43 MUSART_CR2_CLKEN_BIT	565
7.137.2.44 MUSART_CR2_STOP_BIT	565
7.137.2.45 MUSART_CR2_STOP0_BIT	565
7.137.2.46 MUSART_CR2_STOP1_BIT	566
7.137.2.47 MUSART_CR2_LINEN_BIT	566
7.137.2.48 MUSART_CR3_CTSIE_BIT	566
7.137.2.49 MUSART_CR3_CTSE_BIT	566

7.137.2.50 MUSART_CR3_RTSE_BIT	566
7.137.2.51 MUSART_CR3_DMAT_BIT	566
7.137.2.52 MUSART_CR3_DMAR_BIT	567
7.137.2.53 MUSART_CR3_SCEN_BIT	567
7.137.2.54 MUSART_CR3_NACK_BIT	567
7.137.2.55 MUSART_CR3_HDSEL_BIT	567
7.137.2.56 MUSART_CR3_IRLP_BIT	567
7.137.2.57 MUSART_CR3_IREN_BIT	567
7.137.2.58 MUSART_CR3_EIE_BIT	568
7.138 UART_private.h	568
7.139 UART_program.c	570
7.140 MyRTOS_config.h	574
7.141 MyRTOS_interface.h	575
7.142 MyRTOS_private.h	575
7.143 MyRTOS_program.c	576
Index	581

Chapter 1

ADAS Graduation Project

1.1 Description

The graduation project is ADAS and consists of 3 systems:

1. Adaptive Cruise Control
2. Adaptive Light Control
3. Driver Health Care

1.1.1 Adaptive Cruise Control

Adaptive cruise control (ACC) is an enhancement of conventional cruise control. ACC automatically adjusts the speed of your car to match the speed of the car in front of you. If the car ahead slows down, ACC can automatically match it.

1.1.2 Adaptive Light Control

Adaptive light control is a system that changes the brightness of the car's headlights automatically based on the environment's light as well as the cars that are coming in the opposite way. The lighter the environment, the darker the headlights.

1.1.3 Driver Drowsiness Detection

Driver health care is a system that is responsible for the detection of the driver's sleepiness and give the suitable alters to warn the driver as well as the other cars in real-time using Image Processing and Machine Learning techniques.

1.2 Hardware specifications

This project is developed using:

- Microcontroller: STM32F401CDU6

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

DC Motor Directions	11
DC Motor Speeds	12
DC Motor Rotation Directions	15
Bit Manipulation Math Macros	16
Bit Group Manipulation Math Macros	17
Compiler standard macros	19
Utilities macros	21
Standard types	22
Standard values	25
ADC Channels	29
Interrupt line IDs	35
Interrupt trigger status	39
EXTI Memory Addresses	40
EXTI Registers	41
EXTI Lines Status	41
EXTI Line Settings	42
SYSCFG Memory Addresses	43
SYSCFG Memory Registers	43
GPIO Modes	44
GPIO Ports	45
GPIO Output Types	46
GPIO PIN Speed	47
GPIO Pull Types	48
GPIO Output Values	49
GPIO Output PINs	50
GPIO Alternate Functions	54
GPIO Addresses	59
GPIO Registers	60
Interrupt priority grouping	61
Group priorities	65
Sub-Group priorities	70
NVIC Vector Table	75
NVIC Addresses	92
NVIC Registers	93
NVIC Settable Priorities	94

ID options	96
ID options	98
SPI Peripherals	107
SPI Addresses	108
SPI Registers	109
SPI Interfacing Macros	110
Systick Configuration	111
Delay Units	113
Systick Addresses	114
Systick Registers	114
Systick Register Bits Positions	115
Systick Clock Sources	116
Systick Exception (Interrupt) Status	117
UART Addresses	118
UART Registers	119
UART Oversampling	120
UART Operating Modes	121
UART Parity	122
UART Bit Sizes	122
UART Stop Bits	123

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

ADC_MemoryMapType	ADC declaration structure for its registers	125
BUZZER_BuzzerConfiguration	Buzzer configuration structure for Buzzer initialization	130
DCM_MotorConfiguration	DC Motor configuration structure for motor initialization	132
EXTI_ConfigType	Interrupt configuration structure to initialize the interrupt with	134
EXTI_Type	EXTI Configuration structure for interrupt initialization process	135
GPIOx_MemoryMapType	GPIO declaration structure for its registers	137
HULTSNC_ConfigType	Ultrasonic Configurations	140
LED_LEDConfiguration	LED configuration structure for LED initialization	141
MGPIOx_ConfigType	MDIO Configuration structure for a specific PIN initialization	142
MSYSCFG_MemMap_t	System configuration structure to initialize the SYSCFG module with	144
NVIC_MemoryMapType	NVIC configuration structure for NVIC memory map	146
RCC_MemoryMapType	RCC declaration structure for its registers	150
SCB_MemoryMapType	System control block memory map structure	159
SPI_MemoryMapType	SPI Registers	165
SysTick_Type	Systick memory map structure declaration for the Systick's registers	168
Task		169
TIM1_MemoryMapType	TIM1 declaration structure for its registers	170
USART_ClockInitTypeDef	USART Clock declaration structure for its registers	174

USART_InitType	USART Clock declaration structure for	176
USART_MemoryMapType	USART declaration structure for its registers	178

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

COTS/APP/ACC/ ACC_config.h	181
COTS/APP/ACC/ ACC_interface.h	181
COTS/APP/ACC/ ACC_private.h	181
COTS/APP/ACC/ ACC_program.c	182
COTS/APP/Exit_State/ Exit_State_config.h	184
COTS/APP/Exit_State/ Exit_State_interface.h	184
COTS/APP/Exit_State/ Exit_State_private.h	184
COTS/APP/Exit_State/ Exit_State_program.c	185
COTS/APP/FCW/ FCW_config.h	186
COTS/APP/FCW/ FCW_interface.h	186
COTS/APP/FCW/ FCW_private.h	186
COTS/APP/FCW/ FCW_program.c	186
COTS/APP/Mob_APP/ Mob_APP_config.h	189
COTS/APP/Mob_APP/ Mob_APP_interface.h	190
COTS/APP/Mob_APP/ Mob_APP_private.h	190
COTS/APP/Mob_APP/ Mob_APP_program.c	190
COTS/APP/NCC/ NCC_config.h	192
COTS/APP/NCC/ NCC_interface.h	192
COTS/APP/NCC/ NCC_private.h	193
COTS/APP/NCC/ NCC_program.c	193
COTS/APP/Traditional_Mode/ Traditional_Mode_config.h	195
COTS/APP/Traditional_Mode/ Traditional_Mode_interface.h	195
COTS/APP/Traditional_Mode/ Traditional_Mode_private.h	195
COTS/APP/Traditional_Mode/ Traditional_Mode_program.c	195
COTS/HAL/Bluetooth/ Bluetooth_config.c	197
COTS/HAL/Bluetooth/ Bluetooth_config.h	198
COTS/HAL/Bluetooth/ Bluetooth_interface.h	198
COTS/HAL/Bluetooth/ Bluetooth_private.h	199
COTS/HAL/Bluetooth/ Bluetooth_program.c	199
COTS/HAL/BUZZER/ BUZZER_config.h	202
COTS/HAL/BUZZER/ BUZZER_interface.h This file contains the interface information for the Buzzer module	202
COTS/HAL/BUZZER/ BUZZER_private.h	205
COTS/HAL/BUZZER/ BUZZER_program.c This file contains the logical operations of the buzzer module	205

COTS/HAL/Car_Movement/ Car_Movement_config.h	209
COTS/HAL/Car_Movement/ Car_Movement_interface.h	209
COTS/HAL/Car_Movement/ Car_Movement_private.h	210
COTS/HAL/Car_Movement/ Car_Movement_program.c	210
COTS/HAL/DCMOTOR/ DCM_config.h This file contains the configuration information for the DC Motor module	212
COTS/HAL/DCMOTOR/ DCM_interface.h This file contains the interfacing information for the DC Motor module	213
COTS/HAL/DCMOTOR/ DCM_private.h This file contains the private information for the DC Motor module	219
COTS/HAL/DCMOTOR/ DCM_program.c This file contains the source code of the interfacing information for the DC Motor module	220
COTS/HAL/LCD/ LCD_config.h	227
COTS/HAL/LCD/ LCD_interface.h	228
COTS/HAL/LCD/ LCD_private.h	229
COTS/HAL/LCD/ LCD_program.c	230
COTS/HAL/LDR/ LDR_config.c	236
COTS/HAL/LDR/ LDR_config.h	237
COTS/HAL/LDR/ LDR_interface.h	237
COTS/HAL/LDR/ LDR_private.h	238
COTS/HAL/LDR/ LDR_program.c	238
COTS/HAL/LED/ LED_config.h	239
COTS/HAL/LED/ LED_interface.h	239
COTS/HAL/LED/ LED_private.h	240
COTS/HAL/LED/ LED_program.c	240
COTS/HAL/TFT/ TFT_Config.c	241
COTS/HAL/TFT/ TFT_config.h	242
COTS/HAL/TFT/ TFT_interface.h	242
COTS/HAL/TFT/ TFT_private.h	243
COTS/HAL/TFT/ TFT_program.c	243
COTS/HAL/UltraSonic/ UltraSonic_config.h	246
COTS/HAL/UltraSonic/ UltraSonic_interface.h This file is a header file that contains the Ultrasonic sensor functions prototypes	246
COTS/HAL/UltraSonic/ UltraSonic_private.h	249
COTS/HAL/UltraSonic/ UltraSonic_program.c	250
COTS/LIB/ LSTD_BITMATH.h This file contains the bit math manipulation macro-functions	251
COTS/LIB/ LSTD_COMPILER.h This file contains the compiler standard macros	252
COTS/LIB/ LSTD MCU UTILITIES.h This file contains the MCU utility macro-functions	253
COTS/LIB/ LSTD TYPES.h This file contains the standard types	254
COTS/LIB/ LSTD VALUES.h This file contains the standard values	255
COTS/MCAL/ADC/ ADC_config.h	257
COTS/MCAL/ADC/ ADC_interface.h This file contains the interfacing information of ADC Module in STM32F103C8 ARM Microcontroller	264
COTS/MCAL/ADC/ ADC_private.h	273
COTS/MCAL/ADC/ ADC_program.c	278
COTS/MCAL/EXTI/ EXTI_config.h	284
COTS/MCAL/EXTI/ EXTI_interface.h This file contains the interfacing information for the EXTI module	286
COTS/MCAL/EXTI/ EXTI_private.h This file contains the private information related to the EXTI module	295
COTS/MCAL/EXTI/ EXTI_program.c This file contains the source code of the interfacing for the EXTI module	297

COTS/MCAL/EXTI/ SYSCFG_private.h	312
This file contains the private information regarding the SYSCFG	312
COTS/MCAL/GPIO/ GPIO_config.h	313
This file contains the GPIO configurations	313
COTS/MCAL/GPIO/ GPIO_interface.h	315
This file contains the interfacing information for the GPIO module	315
COTS/MCAL/GPIO/ GPIO_private.h	331
This file contains the registers information and addresses for the GPIO module	331
COTS/MCAL/GPIO/ GPIO_program.c	332
This file contains the source code of the interfacing for the GPIO modules	332
COTS/MCAL/NVIC/ NVIC_config.h	349
COTS/MCAL/NVIC/ NVIC_interface.h	349
This file contains the interface information and addresses for the NVIC module	349
COTS/MCAL/NVIC/ NVIC_private.h	360
This file contains the registers information and addresses for the NVIC module	360
COTS/MCAL/NVIC/ NVIC_program.c	363
This file contains the source code of the interfacing for the NVIC modules	363
COTS/MCAL/RCC/ MRCC_config.h	371
This file contains the RCC configurations	371
COTS/MCAL/RCC/ MRCC_interface.h	389
This file contains the interfacing information for the RCC module	389
COTS/MCAL/RCC/ MRCC_private.h	399
This file contains the registers information and addresses for the RCC module	399
COTS/MCAL/RCC/ MRCC_program.c	446
This file contains the source code of the interfacing for the RCC module	446
COTS/MCAL/SPI/ SPI_config.h	459
COTS/MCAL/SPI/ SPI_interface.h	462
COTS/MCAL/SPI/ SPI_private.h	463
COTS/MCAL/SPI/ SPI_program.c	465
COTS/MCAL/SysTick/ SysTick_config.h	473
Configuration file for SysTick driver	473
COTS/MCAL/SysTick/ SysTick_interface.h	474
This file contains the interfacing information for the Systick module	474
COTS/MCAL/SysTick/ SysTick_private.h	485
This file contains the registers information and addresses for the Systick module	485
COTS/MCAL/SysTick/ SysTick_program.c	486
This file contains the source code of the interfacing for the Systick modules	486
COTS/MCAL/TIM1/ TIM1_config.h	501
COTS/MCAL/TIM1/ TIM1_interface.h	510
COTS/MCAL/TIM1/ TIM1_private.h	511
COTS/MCAL/TIM1/ TIM1_program.c	517
COTS/MCAL/UART/ UART_config.h	540
COTS/MCAL/UART/ UART_interface.h	542
This file contains the interfacing information of UART driver	542
COTS/MCAL/UART/ UART_private.h	556
This file contains the private information of UART driver	556
COTS/MCAL/UART/ UART_program.c	570
COTS/MyRTOS/ MyRTOS_config.h	574
COTS/MyRTOS/ MyRTOS_interface.h	575
COTS/MyRTOS/ MyRTOS_private.h	575
COTS/MyRTOS/ MyRTOS_program.c	576

Chapter 5

Module Documentation

5.1 DC Motor Directions

Macros

- #define FORWARD (CW)
Motor forward direction.
- #define BACKWARD (CCW)
Motor backward direction.

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 FORWARD

```
#define FORWARD (CW)

#include <COTS/HAL/DCMOTOR/DCM_interface.h>

Motor forward direction.

Definition at line 115 of file DCM_interface.h.
```

5.1.2.2 BACKWARD

```
#define BACKWARD (CCW)

#include <COTS/HAL/DCMOTOR/DCM_interface.h>

Motor backward direction.

Definition at line 122 of file DCM_interface.h.
```

5.2 DC Motor Speeds

Macros

- #define SPEED_0_PERCENT 0
Motor speed 0%.
- #define SPEED_10_PERCENT 1000U
Motor speed 10%.
- #define SPEED_20_PERCENT 2000U
Motor speed 20%.
- #define SPEED_30_PERCENT 3000U
Motor speed 30%.
- #define SPEED_40_PERCENT 4000U
Motor speed 40%.
- #define SPEED_50_PERCENT 5000U
Motor speed 50%.
- #define SPEED_60_PERCENT 6000U
Motor speed 60%.
- #define SPEED_70_PERCENT 7000U
Motor speed 70%.
- #define SPEED_80_PERCENT 8000U
Motor speed 80%.
- #define SPEED_90_PERCENT 9000U
Motor speed 90%.
- #define SPEED_100_PERCENT 10000U
Motor speed 100%.

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 SPEED_0_PERCENT

```
#define SPEED_0_PERCENT 0

#include <COTS/HAL/DCMOTOR/DCM_interface.h>

Motor speed 0%.
```

Definition at line 136 of file [DCM_interface.h](#).

5.2.2.2 SPEED_10_PERCENT

```
#define SPEED_10_PERCENT 1000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 10%.

Definition at line 142 of file [DCM_interface.h](#).

5.2.2.3 SPEED_20_PERCENT

```
#define SPEED_20_PERCENT 2000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 20%.

Definition at line 148 of file [DCM_interface.h](#).

5.2.2.4 SPEED_30_PERCENT

```
#define SPEED_30_PERCENT 3000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 30%.

Definition at line 154 of file [DCM_interface.h](#).

5.2.2.5 SPEED_40_PERCENT

```
#define SPEED_40_PERCENT 4000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 40%.

Definition at line 160 of file [DCM_interface.h](#).

5.2.2.6 SPEED_50_PERCENT

```
#define SPEED_50_PERCENT 5000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 50%.

Definition at line 166 of file [DCM_interface.h](#).

5.2.2.7 SPEED_60_PERCENT

```
#define SPEED_60_PERCENT 6000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 60%.

Definition at line 172 of file [DCM_interface.h](#).

5.2.2.8 SPEED_70_PERCENT

```
#define SPEED_70_PERCENT 7000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 70%.

Definition at line 178 of file [DCM_interface.h](#).

5.2.2.9 SPEED_80_PERCENT

```
#define SPEED_80_PERCENT 8000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 80%.

Definition at line 184 of file [DCM_interface.h](#).

5.2.2.10 SPEED_90_PERCENT

```
#define SPEED_90_PERCENT 9000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 90%.

Definition at line 190 of file [DCM_interface.h](#).

5.2.2.11 SPEED_100_PERCENT

```
#define SPEED_100_PERCENT 10000U

#include <COTS/HAL/DCMOTOR/DCM_interface.h>
```

Motor speed 100%.

Definition at line 196 of file [DCM_interface.h](#).

5.3 DC Motor Rotation Directions

Macros

- #define [CW](#) (1)
Motor clockwise rotation direction.
- #define [CCW](#) (2)
Motor counter-clockwise rotation direction.

5.3.1 Detailed Description

5.3.2 Macro Definition Documentation

5.3.2.1 CW

```
#define CW (1)

#include <COTS/HAL/DCMOTOR/DCM_private.h>
```

Motor clockwise rotation direction.

Definition at line 24 of file [DCM_private.h](#).

5.3.2.2 CCW

```
#define CCW (2)

#include <COTS/HAL/DCMOTOR/DCM_private.h>
```

Motor counter-clockwise rotation direction.

Definition at line 31 of file [DCM_private.h](#).

5.4 Bit Manipulation Math Macros

Macros

- #define [SET_BIT](#)(Reg, bitnum) (Reg) |= (1 << (bitnum))
- #define [CLR_BIT](#)(Reg, bitnum) (Reg) &= ~(1 << (bitnum))
- #define [TOGGLE_BIT](#)(Reg, bitnum) (Reg) ^= (1 << (bitnum))
- #define [GET_BIT](#)(Reg, bitnum) (((Reg)>>(bitnum)) & 1)

5.4.1 Detailed Description

5.4.2 Macro Definition Documentation

5.4.2.1 SET_BIT

```
#define SET_BIT(
    Reg,
    bitnum ) (Reg) |= (1 << (bitnum))

#include <COTS/LIB/LSTD_BITMATH.h>
```

Sets a certain bit's value

Definition at line 23 of file [LSTD_BITMATH.h](#).

5.4.2.2 CLR_BIT

```
#define CLR_BIT(
    Reg,
    bitnum ) (Reg) &= ~(1 << (bitnum))

#include <COTS/LIB/LSTD_BITMATH.h>
```

Clears a certain bit's value to

Definition at line 30 of file [LSTD_BITMATH.h](#).

5.4.2.3 TOGGLE_BIT

```
#define TOGGLE_BIT(
    Reg,
    bitnum ) (Reg) ^= (1 << (bitnum))
```

```
#include <COTS/LIB/LSTD_BITMATH.h>
```

Toggle a bit to 0 if it's 1, 1 otherwise

Definition at line 37 of file [LSTD_BITMATH.h](#).

5.4.2.4 GET_BIT

```
#define GET_BIT(
    Reg,
    bitnum ) (((Reg)>>(bitnum)) & 1)
```

```
#include <COTS/LIB/LSTD_BITMATH.h>
```

Returns the value of the bit whether it's 1 or 0

Definition at line 44 of file [LSTD_BITMATH.h](#).

5.5 Bit Group Manipulation Math Macros

Macros

- #define SET_BITs(Reg, bits, bitnum, factor) (Reg) |= ((bits) << (bitnum * factor))
- #define CLR_BITs(Reg, bits, bitnum, factor) (Reg) &= ~((bits) << (bitnum * factor))
- #define TOGGLE_BITs(Reg, bits, bitnum, factor) (Reg) ^= ((bits) << (bitnum * factor))
- #define GET_BITs(Reg, bits, bitnum, factor) (((Reg) >> (bitnum * factor)) & (bits))

5.5.1 Detailed Description

5.5.2 Macro Definition Documentation

5.5.2.1 SET_BITs

```
#define SET_BITs(
    Reg,
    bits,
    bitnum,
    factor ) (Reg) |= ((bits) << (bitnum * factor))
```

```
#include <COTS/LIB/LSTD_BITMATH.h>
```

Sets the values of a group of bits

Definition at line 59 of file [LSTD_BITMATH.h](#).

5.5.2.2 CLR_BITs

```
#define CLR_BITs(
    Reg,
    bits,
    bitnum,
    factor ) (Reg) &= ~((bits) << (bitnum * factor))

#include <COTS/LIB/LSTD_BITMATH.h>
```

Clears the value of a group of bits

Definition at line 66 of file [LSTD_BITMATH.h](#).

5.5.2.3 TOGGLE_BITs

```
#define TOGGLE_BITs(
    Reg,
    bits,
    bitnum,
    factor ) (Reg) ^= ((bits) << (bitnum * factor))

#include <COTS/LIB/LSTD_BITMATH.h>
```

Toggles the value of a group of bits

Definition at line 73 of file [LSTD_BITMATH.h](#).

5.5.2.4 GET_BITs

```
#define GET_BITs(
    Reg,
    bits,
    bitnum,
    factor ) (((Reg) >> (bitnum * factor)) & (bits))

#include <COTS/LIB/LSTD_BITMATH.h>
```

Returns the value of a group of bits

Definition at line 80 of file [LSTD_BITMATH.h](#).

5.6 Compiler standard macros

Macros

- #define **VAR**(vartype) vartype
- #define **FUNC**(rettype) rettype
- #define **P2VAR**(ptrtype) ptrtype *
- #define **P2CONST**(ptrtype) const ptrtype *
- #define **CONSTP2VAR**(ptrtype) ptrtype * const
- #define **CONSTP2CONST**(ptrtype) const ptrtype * const
- #define **P2FUNC**(rettype, fctname) rettype (*fctname)
- #define **CONST**(consttype) const consttype
- #define **STATIC** static

5.6.1 Detailed Description

5.6.2 Macro Definition Documentation

5.6.2.1 VAR

```
#define VAR(  
    vartype ) vartype  
  
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a variable with the specified type

Definition at line 23 of file [LSTD_COMPILER.h](#).

5.6.2.2 FUNC

```
#define FUNC(  
    rettype ) rettype  
  
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a function's return type

Definition at line 30 of file [LSTD_COMPILER.h](#).

5.6.2.3 P2VAR

```
#define P2VAR(  
    ptrtype ) ptrtype *
```



```
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-variable with the specified type

Definition at line [37](#) of file [LSTD_COMPILER.h](#).

5.6.2.4 P2CONST

```
#define P2CONST(  
    ptrtype ) const ptrtype *
```



```
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a constant pointer-to-variable with the specified type

Definition at line [44](#) of file [LSTD_COMPILER.h](#).

5.6.2.5 CONSTP2VAR

```
#define CONSTP2VAR(  
    ptrtype ) ptrtype * const
```



```
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-variable constant with the specified type

Definition at line [51](#) of file [LSTD_COMPILER.h](#).

5.6.2.6 CONSTP2CONST

```
#define CONSTP2CONST(  
    ptrtype ) const ptrtype * const
```



```
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a constant pointer-to-variable constant with the specified type

Definition at line [58](#) of file [LSTD_COMPILER.h](#).

5.6.2.7 P2FUNC

```
#define P2FUNC(  
    rettype,  
    fctname ) rettype (*fctname)  
  
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a pointer-to-function with the specified type

Definition at line 65 of file [LSTD_COMPILER.h](#).

5.6.2.8 CONST

```
#define CONST(  
    consttype ) const consttype  
  
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a standard constant variable with the specified type

Definition at line 73 of file [LSTD_COMPILER.h](#).

5.6.2.9 STATIC

```
#define STATIC static  
  
#include <COTS/LIB/LSTD_COMPILER.h>
```

Declare a standard static variable

Definition at line 82 of file [LSTD_COMPILER.h](#).

5.7 Utilities macros

Macros

- #define MY_MS_DELAY(T) do{ u32 Timer = (T * 500); while (Timer--) { asm ("nop"); } } while(0);

5.7.1 Detailed Description

5.7.2 Macro Definition Documentation

5.7.2.1 MY_MS_DELAY

```
#define MY_MS_DELAY( T ) do{ u32 Timer = (T * 500); while (Timer--) { asm ("nop"); } } while(0);

#include <COTS/LIB/LSTD MCU UTILITIES.h>
```

Declare a delay with a T microseconds with a NOP delay

Definition at line 23 of file [LSTD MCU UTILITIES.h](#).

5.8 Standard types

Typedefs

- `typedef unsigned char bool_t`
- `typedef unsigned char u8_t`
- `typedef char c8_t`
- `typedef unsigned short int u16_t`
- `typedef unsigned int u32_t`
- `typedef unsigned long int u64_t`
- `typedef signed long int s64_t`
- `typedef signed char s8_t`
- `typedef signed short int s16_t`
- `typedef signed int s32_t`
- `typedef float f32_t`
- `typedef double f64_t`

5.8.1 Detailed Description

5.8.2 Typedef Documentation

5.8.2.1 bool_t

```
bool_t

#include <COTS/LIB/LSTD TYPES.h>
```

Type definition for boolean

Definition at line 23 of file [LSTD TYPES.h](#).

5.8.2.2 u8_t

u8_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 8-bit unsigned INT

Definition at line 30 of file [LSTD_TYPES.h](#).

5.8.2.3 c8_t

c8_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 8-bit char

Definition at line 37 of file [LSTD_TYPES.h](#).

5.8.2.4 u16_t

u16_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 16-bit unsigned INT

Definition at line 44 of file [LSTD_TYPES.h](#).

5.8.2.5 u32_t

u32_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 32-bit unsigned INT

Definition at line 51 of file [LSTD_TYPES.h](#).

5.8.2.6 u64_t

u64_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 64-bit unsigned INT

Definition at line 58 of file [LSTD_TYPES.h](#).

5.8.2.7 s64_t

s64_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 64-bit signed INT

Definition at line 65 of file [LSTD_TYPES.h](#).

5.8.2.8 s8_t

s8_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 8-bit signed INT

Definition at line 72 of file [LSTD_TYPES.h](#).

5.8.2.9 s16_t

s16_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 16-bit signed INT

Definition at line 79 of file [LSTD_TYPES.h](#).

5.8.2.10 s32_t

s32_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 32-bit signed INT

Definition at line 86 of file [LSTD_TYPES.h](#).

5.8.2.11 f32_t

f32_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 32-bit float

Definition at line 93 of file [LSTD_TYPES.h](#).

5.8.2.12 f64_t

f64_t

```
#include <COTS/LIB/LSTD_TYPES.h>
```

Type definition for 64-bit float

Definition at line 100 of file [LSTD_TYPES.h](#).

5.9 Standard values

Macros

- #define TRUE (1)
- #define FALSE (0)
- #define NULL ((void*)0)
- #define INITIAL_ZERO (0)
- #define FLAG_SET (1)
- #define FLAG_CLEARED (0)
- #define RUN (1)
- #define STOP (0)
- #define PRESSED (1)
- #define RELEASED (0)
- #define SAME_STRING (0)
- #define DIFFERENT_STRING (1)

5.9.1 Detailed Description

5.9.2 Macro Definition Documentation

5.9.2.1 TRUE

```
#define TRUE (1)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for TRUE

Definition at line 24 of file [LSTD_VALUES.h](#).

5.9.2.2 FALSE

```
#define FALSE (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for FALSE

Definition at line 33 of file [LSTD_VALUES.h](#).

5.9.2.3 NULL

```
#define NULL ( (void*)0 )

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for NULL

Definition at line 42 of file [LSTD_VALUES.h](#).

5.9.2.4 INITIAL_ZERO

```
#define INITIAL_ZERO (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for INITIAL_ZERO

Definition at line 56 of file [LSTD_VALUES.h](#).

5.9.2.5 FLAG_SET

```
#define FLAG_SET (1)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for FLAG_SET

Definition at line 65 of file [LSTD_VALUES.h](#).

5.9.2.6 FLAG_CLEARED

```
#define FLAG_CLEARED (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for FLAG_CLEARED

Definition at line 74 of file [LSTD_VALUES.h](#).

5.9.2.7 RUN

```
#define RUN (1)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for RUN

Definition at line 83 of file [LSTD_VALUES.h](#).

5.9.2.8 STOP

```
#define STOP (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for STOP

Definition at line 93 of file [LSTD_VALUES.h](#).

5.9.2.9 **PRESSED**

```
#define PRESSED (1)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for PRESSED

Definition at line 102 of file [LSTD_VALUES.h](#).

5.9.2.10 **RELEASED**

```
#define RELEASED (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for RELEASED

Definition at line 111 of file [LSTD_VALUES.h](#).

5.9.2.11 **SAME_STRING**

```
#define SAME_STRING (0)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for SAME_STRING

Definition at line 120 of file [LSTD_VALUES.h](#).

5.9.2.12 **DIFFERENT_STRING**

```
#define DIFFERENT_STRING (1)

#include <COTS/LIB/LSTD_VALUES.h>
```

Type definition for DIFFERENT_STRING

Definition at line 129 of file [LSTD_VALUES.h](#).

5.10 ADC Channels

Macros

- #define CHANNEL0 0
Channel 0.
- #define CHANNEL1 1
Channel 1.
- #define CHANNEL2 2
Channel 2.
- #define CHANNEL3 3
Channel 3.
- #define CHANNEL4 4
Channel 4.
- #define CHANNEL5 5
Channel 5.
- #define CHANNEL6 6
Channel 6.
- #define CHANNEL7 7
Channel 7.
- #define CHANNEL8 8
Channel 8.
- #define CHANNEL9 9
Channel 9.
- #define CHANNEL10 10
Channel 10.
- #define CHANNEL11 11
Channel 11.
- #define CHANNEL12 12
Channel 12.
- #define CHANNEL13 13
Channel 13.
- #define CHANNEL14 14
Channel 14.
- #define CHANNEL15 15
Channel 15.
- #define CHANNEL16 16
Channel 16.
- #define CHANNEL17 17
Channel 17.
- #define CHANNEL18 18
Channel 18.

5.10.1 Detailed Description

5.10.2 Macro Definition Documentation

5.10.2.1 CHANNEL0

```
#define CHANNEL0 0  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 0.

This macro is used to select channel 0

Definition at line [83](#) of file [ADC_interface.h](#).

5.10.2.2 CHANNEL1

```
#define CHANNEL1 1  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 1.

This macro is used to select channel 1

Definition at line [90](#) of file [ADC_interface.h](#).

5.10.2.3 CHANNEL2

```
#define CHANNEL2 2  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 2.

This macro is used to select channel 2

Definition at line [97](#) of file [ADC_interface.h](#).

5.10.2.4 CHANNEL3

```
#define CHANNEL3 3  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 3.

This macro is used to select channel 3

Definition at line [104](#) of file [ADC_interface.h](#).

5.10.2.5 CHANNEL4

```
#define CHANNEL4 4  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 4.

This macro is used to select channel 4

Definition at line 111 of file [ADC_interface.h](#).

5.10.2.6 CHANNEL5

```
#define CHANNEL5 5  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 5.

This macro is used to select channel 5

Definition at line 118 of file [ADC_interface.h](#).

5.10.2.7 CHANNEL6

```
#define CHANNEL6 6  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 6.

This macro is used to select channel 6

Definition at line 125 of file [ADC_interface.h](#).

5.10.2.8 CHANNEL7

```
#define CHANNEL7 7  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 7.

This macro is used to select channel 7

Definition at line 132 of file [ADC_interface.h](#).

5.10.2.9 CHANNEL8

```
#define CHANNEL8 8  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 8.

This macro is used to select channel 8

Definition at line [139](#) of file [ADC_interface.h](#).

5.10.2.10 CHANNEL9

```
#define CHANNEL9 9  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 9.

This macro is used to select channel 9

Definition at line [146](#) of file [ADC_interface.h](#).

5.10.2.11 CHANNEL10

```
#define CHANNEL10 10  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 10.

This macro is used to select channel 10

Definition at line [153](#) of file [ADC_interface.h](#).

5.10.2.12 CHANNEL11

```
#define CHANNEL11 11  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 11.

This macro is used to select channel 11

Definition at line [160](#) of file [ADC_interface.h](#).

5.10.2.13 CHANNEL12

```
#define CHANNEL12 12  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 12.

This macro is used to select channel 12

Definition at line [167](#) of file [ADC_interface.h](#).

5.10.2.14 CHANNEL13

```
#define CHANNEL13 13  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 13.

This macro is used to select channel 13

Definition at line [174](#) of file [ADC_interface.h](#).

5.10.2.15 CHANNEL14

```
#define CHANNEL14 14  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 14.

This macro is used to select channel 14

Definition at line [181](#) of file [ADC_interface.h](#).

5.10.2.16 CHANNEL15

```
#define CHANNEL15 15  
  
#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 15.

This macro is used to select channel 15

Definition at line [188](#) of file [ADC_interface.h](#).

5.10.2.17 CHANNEL16

```
#define CHANNEL16 16

#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 16.

This macro is used to select channel 16

Definition at line 195 of file [ADC_interface.h](#).

5.10.2.18 CHANNEL17

```
#define CHANNEL17 17

#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 17.

This macro is used to select channel 17

Definition at line 202 of file [ADC_interface.h](#).

5.10.2.19 CHANNEL18

```
#define CHANNEL18 18

#include <COTS/MCAL/ADC/ADC_interface.h>
```

Channel 18.

This macro is used to select channel 18

Definition at line 209 of file [ADC_interface.h](#).

5.11 Interrupt line IDs

Macros

- #define `EXTI_LINE0` (0)
Line 0.
- #define `EXTI_LINE1` (1)
Line 1.
- #define `EXTI_LINE2` (2)
Line 2.
- #define `EXTI_LINE3` (3)
Line 3.
- #define `EXTI_LINE4` (4)
Line 4.
- #define `EXTI_LINE5` (5)
Line 5.
- #define `EXTI_LINE6` (6)
Line 6.
- #define `EXTI_LINE7` (7)
Line 7.
- #define `EXTI_LINE8` (8)
Line 8.
- #define `EXTI_LINE9` (9)
Line 9.
- #define `EXTI_LINE10` (10)
Line 10.
- #define `EXTI_LINE11` (11)
Line 11.
- #define `EXTI_LINE12` (12)
Line 12.
- #define `EXTI_LINE13` (13)
Line 13.
- #define `EXTI_LINE14` (14)
Line 14.
- #define `EXTI_LINE15` (15)
Line 15.

5.11.1 Detailed Description

5.11.2 Macro Definition Documentation

5.11.2.1 EXTI_LINE0

```
#define EXTI_LINE0 (0)

#include <COTS/MCAL/EXTI/EXTI_interface.h>

Line 0.
```

Definition at line 104 of file [EXTI_interface.h](#).

5.11.2.2 EXTI_LINE1

```
#define EXTI_LINE1 (1)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 1.

Definition at line 110 of file [EXTI_interface.h](#).

5.11.2.3 EXTI_LINE2

```
#define EXTI_LINE2 (2)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 2.

Definition at line 116 of file [EXTI_interface.h](#).

5.11.2.4 EXTI_LINE3

```
#define EXTI_LINE3 (3)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 3.

Definition at line 122 of file [EXTI_interface.h](#).

5.11.2.5 EXTI_LINE4

```
#define EXTI_LINE4 (4)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 4.

Definition at line 128 of file [EXTI_interface.h](#).

5.11.2.6 EXTI_LINE5

```
#define EXTI_LINE5 (5)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 5.

Definition at line 134 of file [EXTI_interface.h](#).

5.11.2.7 EXTI_LINE6

```
#define EXTI_LINE6 (6)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 6.

Definition at line 140 of file [EXTI_interface.h](#).

5.11.2.8 EXTI_LINE7

```
#define EXTI_LINE7 (7)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 7.

Definition at line 146 of file [EXTI_interface.h](#).

5.11.2.9 EXTI_LINE8

```
#define EXTI_LINE8 (8)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 8.

Definition at line 152 of file [EXTI_interface.h](#).

5.11.2.10 EXTI_LINE9

```
#define EXTI_LINE9 (9)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 9.

Definition at line 158 of file [EXTI_interface.h](#).

5.11.2.11 EXTI_LINE10

```
#define EXTI_LINE10 (10)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 10.

Definition at line 164 of file [EXTI_interface.h](#).

5.11.2.12 EXTI_LINE11

```
#define EXTI_LINE11 (11)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 11.

Definition at line 170 of file [EXTI_interface.h](#).

5.11.2.13 EXTI_LINE12

```
#define EXTI_LINE12 (12)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 12.

Definition at line 176 of file [EXTI_interface.h](#).

5.11.2.14 EXTI_LINE13

```
#define EXTI_LINE13 (13)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 13.

Definition at line 182 of file [EXTI_interface.h](#).

5.11.2.15 EXTI_LINE14

```
#define EXTI_LINE14 (14)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 14.

Definition at line 188 of file [EXTI_interface.h](#).

5.11.2.16 EXTI_LINE15

```
#define EXTI_LINE15 (15)

#include <COTS/MCAL/EXTI/EXTI_interface.h>
```

Line 15.

Definition at line 194 of file [EXTI_interface.h](#).

5.12 Interrupt trigger status

Macros

- #define [EXTI_FallingEdge](#) (1)
trigger interrupt on falling edge
- #define [EXTI_RisingEdge](#) (2)
trigger interrupt on rising edge
- #define [EXTI_OnChange](#) (3)
trigger interrupt on level change

5.12.1 Detailed Description

5.12.2 Macro Definition Documentation

5.12.2.1 EXTI_FallingEdge

```
#define EXTI_FallingEdge (1)

#include <COTS/MCAL/EXTI/EXTI_interface.h>

trigger interrupt on falling edge

Definition at line 212 of file EXTI_interface.h.
```

5.12.2.2 EXTI_RisingEdge

```
#define EXTI_RisingEdge (2)

#include <COTS/MCAL/EXTI/EXTI_interface.h>

trigger interrupt on rising edge

Definition at line 218 of file EXTI_interface.h.
```

5.12.2.3 EXTI_OnChange

```
#define EXTI_OnChange (3)

#include <COTS/MCAL/EXTI/EXTI_interface.h>

trigger interrupt on level change

Definition at line 224 of file EXTI_interface.h.
```

5.13 EXTI Memory Addresses

Macros

- #define EXTI_BASE_ADDRESS (0x40013C00)

5.13.1 Detailed Description

5.13.2 Macro Definition Documentation

5.13.2.1 EXTI_BASE_ADDRESS

```
#define EXTI_BASE_ADDRESS (0x40013C00)

#include <COTS/MCAL/EXTI/EXTI_private.h>
```

EXTI base address

Definition at line 57 of file [EXTI_private.h](#).

5.14 EXTI Registers

Macros

- #define MEXTI ((volatile P2VAR(EXTI_Type))(EXTI_BASE_ADDRESS))

5.14.1 Detailed Description

5.14.2 Macro Definition Documentation

5.14.2.1 MEXTI

```
#define MEXTI ((volatile P2VAR(EXTI_Type))(EXTI_BASE_ADDRESS))

#include <COTS/MCAL/EXTI/EXTI_private.h>

EXTI register
```

Definition at line 71 of file [EXTI_private.h](#).

5.15 EXTI Lines Status

Macros

- #define ENABLE (1)
- #define DISABLE (2)

5.15.1 Detailed Description

5.15.2 Macro Definition Documentation

5.15.2.1 ENABLE

```
#define ENABLE (1)

#include <COTS/MCAL/EXTI/EXTI_private.h>
```

Enable a certain line ID from the config

Definition at line 85 of file [EXTI_private.h](#).

5.15.2.2 DISABLE

```
#define DISABLE (2)

#include <COTS/MCAL/EXTI/EXTI_private.h>
```

Disable a certain line ID from the config

Definition at line 91 of file [EXTI_private.h](#).

5.16 EXTI Line Settings

Macros

- #define [EXTI_MAX_EXTI_NUM](#) (22)

5.16.1 Detailed Description

5.16.2 Macro Definition Documentation

5.16.2.1 EXTI_MAX_EXTI_NUM

```
#define EXTI_MAX_EXTI_NUM (22)

#include <COTS/MCAL/EXTI/EXTI_private.h>

Maximum number of available interrupt lines

Definition at line 105 of file EXTI\_private.h.
```

5.17 SYSCFG Memory Addresses

Macros

- #define `SYSCFG_BASE_ADDR` (0x40013800)

5.17.1 Detailed Description

5.17.2 Macro Definition Documentation

5.17.2.1 `SYSCFG_BASE_ADDR`

```
#define SYSCFG_BASE_ADDR (0x40013800)

#include <COTS/MCAL/EXTI/SYSCFG_private.h>

SYSCFG base address

Definition at line 51 of file SYSCFG_private.h.
```

5.18 SYSCFG Memory Registers

Macros

- #define `MSYSCFG` ((volatile `MSYSCFG_MemMap_t` *)(`SYSCFG_BASE_ADDR`))

5.18.1 Detailed Description

5.18.2 Macro Definition Documentation

5.18.2.1 `MSYSCFG`

```
#define MSYSCFG ((volatile MSYSCFG_MemMap_t *) (SYSCFG_BASE_ADDR))

#include <COTS/MCAL/EXTI/SYSCFG_private.h>

SYSCFG register

Definition at line 64 of file SYSCFG_private.h.
```

5.19 GPIO Modes

Macros

- `#define GPIOx_MODE_INPUT (0b00)`
Control input mode.
- `#define GPIOx_MODE_OUTPUT (0b01)`
Control output mode.
- `#define GPIOx_MODE_AF (0b10)`
Control alternate function mode.
- `#define GPIOx_MODE_ANALOG (0b11)`
Control analog mode.

5.19.1 Detailed Description

5.19.2 Macro Definition Documentation

5.19.2.1 GPIOx_MODE_INPUT

```
#define GPIOx_MODE_INPUT (0b00)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

Control input mode.

Definition at line 170 of file GPIO\_interface.h.
```

5.19.2.2 GPIOx_MODE_OUTPUT

```
#define GPIOx_MODE_OUTPUT (0b01)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

Control output mode.

Definition at line 177 of file GPIO\_interface.h.
```

5.19.2.3 GPIOx_MODE_AF

```
#define GPIOx_MODE_AF (0b10)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

Control alternate function mode.

Definition at line 184 of file GPIO\_interface.h.
```

5.19.2.4 GPIOx_MODE_ANALOG

```
#define GPIOx_MODE_ANALOG (0b11)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

Control analog mode.

Definition at line 191 of file [GPIO_interface.h](#).

5.20 GPIO Ports

Macros

- #define [GPIO_PORTA](#) (0)
GPIO Port A.
- #define [GPIO_PORTB](#) (1)
GPIO Port B.
- #define [GPIO_PORTC](#) (2)
GPIO Port C.

5.20.1 Detailed Description

5.20.2 Macro Definition Documentation

5.20.2.1 GPIO_PORTA

```
#define GPIO_PORTA (0)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO Port A.
```

Definition at line 206 of file [GPIO_interface.h](#).

5.20.2.2 GPIO_PORTB

```
#define GPIO_PORTB (1)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO Port B.
```

Definition at line 213 of file [GPIO_interface.h](#).

5.20.2.3 GPIO_PORTC

```
#define GPIO_PORTC (2)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Port C.

Definition at line 220 of file [GPIO_interface.h](#).

5.21 GPIO Output Types

Macros

- `#define GPIOx_OPENDRAIN (1)`
GPIO open-drain.
- `#define GPIOx_PUSH_PULL (2)`
GPIO push-pull.

5.21.1 Detailed Description

5.21.2 Macro Definition Documentation

5.21.2.1 GPIOx_OPENDRAIN

```
#define GPIOx_OPENDRAIN (1)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO open-drain.
```

Definition at line 235 of file [GPIO_interface.h](#).

5.21.2.2 GPIOx_PUSH_PULL

```
#define GPIOx_PUSH_PULL (2)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO push-pull.
```

Definition at line 242 of file [GPIO_interface.h](#).

5.22 GPIO PIN Speed

Macros

- #define **GPIOx_LowSpeed** (0b00)
GPIO low speed.
- #define **GPIOx_MediumSpeed** (0b01)
GPIO medium speed.
- #define **GPIOx_HighSpeed** (0b10)
GPIO high speed.
- #define **GPIOx_VeryHighSpeed** (0b11)
GPIO very high speed.

5.22.1 Detailed Description

5.22.2 Macro Definition Documentation

5.22.2.1 **GPIOx_LowSpeed**

```
#define GPIOx_LowSpeed (0b00)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO low speed.
```

Definition at line 257 of file [GPIO_interface.h](#).

5.22.2.2 **GPIOx_MediumSpeed**

```
#define GPIOx_MediumSpeed (0b01)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO medium speed.
```

Definition at line 264 of file [GPIO_interface.h](#).

5.22.2.3 **GPIOx_HighSpeed**

```
#define GPIOx_HighSpeed (0b10)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO high speed.
```

Definition at line 271 of file [GPIO_interface.h](#).

5.22.2.4 GPIOx_VeryHighSpeed

```
#define GPIOx_VeryHighSpeed (0b11)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO very high speed.

Definition at line 278 of file [GPIO_interface.h](#).

5.23 GPIO Pull Types

Macros

- #define [GPIOx_NoPull](#) (0b00)
GPIO No PULL.
- #define [GPIOx_PullUp](#) (0b01)
GPIO Pull UP.
- #define [GPIOx_PullDown](#) (0b10)
GPIO Pull Down.

5.23.1 Detailed Description

5.23.2 Macro Definition Documentation

5.23.2.1 GPIOx_NoPull

```
#define GPIOx_NoPull (0b00)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO No PULL.

Definition at line 293 of file [GPIO_interface.h](#).

5.23.2.2 GPIOx_PullUp

```
#define GPIOx_PullUp (0b01)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Pull UP.

Definition at line 300 of file [GPIO_interface.h](#).

5.23.2.3 GPIOx_PullDown

```
#define GPIOx_PullDown (0b10)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Pull Down.

Definition at line 307 of file [GPIO_interface.h](#).

5.24 GPIO Output Values

Macros

- #define **GPIOx_HIGH** (1)
GPIO output high.
- #define **GPIOx_LOW** (2)
GPIO output low.

5.24.1 Detailed Description

5.24.2 Macro Definition Documentation

5.24.2.1 GPIOx_HIGH

```
#define GPIOx_HIGH (1)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO output high.
```

Definition at line 322 of file [GPIO_interface.h](#).

5.24.2.2 GPIOx_LOW

```
#define GPIOx_LOW (2)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO output low.
```

Definition at line 329 of file [GPIO_interface.h](#).

5.25 GPIO Output PINs

Macros

- `#define GPIOx_PIN0 (0)`
GPIO PIN 0.
- `#define GPIOx_PIN1 (1)`
GPIO PIN 1.
- `#define GPIOx_PIN2 (2)`
GPIO PIN 2.
- `#define GPIOx_PIN3 (3)`
GPIO PIN 3.
- `#define GPIOx_PIN4 (4)`
GPIO PIN 4.
- `#define GPIOx_PIN5 (5)`
GPIO PIN 5.
- `#define GPIOx_PIN6 (6)`
GPIO PIN 6.
- `#define GPIOx_PIN7 (7)`
GPIO PIN 7.
- `#define GPIOx_PIN8 (8)`
GPIO PIN 8.
- `#define GPIOx_PIN9 (9)`
brief GPIO PIN 9
- `#define GPIOx_PIN10 (10)`
GPIO PIN 10.
- `#define GPIOx_PIN11 (11)`
GPIO PIN 11.
- `#define GPIOx_PIN12 (12)`
GPIO PIN 12.
- `#define GPIOx_PIN13 (13)`
GPIO PIN 13.
- `#define GPIOx_PIN14 (14)`
GPIO PIN 14.
- `#define GPIOx_PIN15 (15)`
GPIO PIN 15.

5.25.1 Detailed Description

5.25.2 Macro Definition Documentation

5.25.2.1 GPIOx_PIN0

```
#define GPIOx_PIN0 (0)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO PIN 0.

Definition at line 344 of file GPIO\_interface.h.
```

5.25.2.2 GPIOx_PIN1

```
#define GPIOx_PIN1 (1)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 1.

Definition at line 351 of file [GPIO_interface.h](#).

5.25.2.3 GPIOx_PIN2

```
#define GPIOx_PIN2 (2)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 2.

Definition at line 358 of file [GPIO_interface.h](#).

5.25.2.4 GPIOx_PIN3

```
#define GPIOx_PIN3 (3)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 3.

Definition at line 365 of file [GPIO_interface.h](#).

5.25.2.5 GPIOx_PIN4

```
#define GPIOx_PIN4 (4)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 4.

Definition at line 372 of file [GPIO_interface.h](#).

5.25.2.6 GPIOx_PIN5

```
#define GPIOx_PIN5 (5)

#include <COTS/MCAL/GPIO(GPIO_interface.h>
```

GPIO PIN 5.

Definition at line 379 of file [GPIO_interface.h](#).

5.25.2.7 GPIOx_PIN6

```
#define GPIOx_PIN6 (6)

#include <COTS/MCAL/GPIO(GPIO_interface.h>
```

GPIO PIN 6.

Definition at line 386 of file [GPIO_interface.h](#).

5.25.2.8 GPIOx_PIN7

```
#define GPIOx_PIN7 (7)

#include <COTS/MCAL/GPIO(GPIO_interface.h>
```

GPIO PIN 7.

Definition at line 393 of file [GPIO_interface.h](#).

5.25.2.9 GPIOx_PIN8

```
#define GPIOx_PIN8 (8)

#include <COTS/MCAL/GPIO(GPIO_interface.h>
```

GPIO PIN 8.

Definition at line 400 of file [GPIO_interface.h](#).

5.25.2.10 GPIOx_PIN9

```
#define GPIOx_PIN9 (9)

#include <COTS/MCAL/GPIO(GPIO_interface.h>

brief GPIO PIN 9

Definition at line 407 of file GPIO\_interface.h.
```

5.25.2.11 GPIOx_PIN10

```
#define GPIOx_PIN10 (10)

#include <COTS/MCAL/GPIO(GPIO_interface.h>

GPIO PIN 10.

Definition at line 414 of file GPIO\_interface.h.
```

5.25.2.12 GPIOx_PIN11

```
#define GPIOx_PIN11 (11)

#include <COTS/MCAL/GPIO(GPIO_interface.h>

GPIO PIN 11.

Definition at line 421 of file GPIO\_interface.h.
```

5.25.2.13 GPIOx_PIN12

```
#define GPIOx_PIN12 (12)

#include <COTS/MCAL/GPIO(GPIO_interface.h>

GPIO PIN 12.

Definition at line 428 of file GPIO\_interface.h.
```

5.25.2.14 GPIOx_PIN13

```
#define GPIOx_PIN13 (13)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 13.

Definition at line [435](#) of file [GPIO_interface.h](#).

5.25.2.15 GPIOx_PIN14

```
#define GPIOx_PIN14 (14)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 14.

Definition at line [442](#) of file [GPIO_interface.h](#).

5.25.2.16 GPIOx_PIN15

```
#define GPIOx_PIN15 (15)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO PIN 15.

Definition at line [449](#) of file [GPIO_interface.h](#).

5.26 GPIO Alternate Functions

Macros

- #define [GPIOx_AF0](#) (0)
GPIO Alternate function 0.
- #define [GPIOx_AF1](#) (1)
GPIO Alternate function 1.
- #define [GPIOx_AF2](#) (2)
GPIO Alternate function 2.
- #define [GPIOx_AF3](#) (3)
GPIO Alternate function 3.
- #define [GPIOx_AF4](#) (4)
GPIO Alternate function 4.
- #define [GPIOx_AF5](#) (5)
GPIO Alternate function 5.

- #define [GPIOx_AF6](#) (6)
GPIO Alternate function 6.
- #define [GPIOx_AF7](#) (7)
GPIO Alternate function 7.
- #define [GPIOx_AF8](#) (8)
GPIO Alternate function 8.
- #define [GPIOx_AF9](#) (9)
GPIO Alternate function 9.
- #define [GPIOx_AF10](#) (10)
GPIO Alternate function 10.
- #define [GPIOx_AF11](#) (11)
GPIO Alternate function 11.
- #define [GPIOx_AF12](#) (12)
GPIO Alternate function 12.
- #define [GPIOx_AF13](#) (13)
GPIO Alternate function 13.
- #define [GPIOx_AF14](#) (14)
GPIO Alternate function 14.
- #define [GPIOx_AF15](#) (15)
GPIO Alternate function 15.

5.26.1 Detailed Description

5.26.2 Macro Definition Documentation

5.26.2.1 GPIOx_AF0

```
#define GPIOx_AF0 (0)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO Alternate function 0.
```

Definition at line [464](#) of file [GPIO_interface.h](#).

5.26.2.2 GPIOx_AF1

```
#define GPIOx_AF1 (1)

#include <COTS/MCAL/GPIO/GPIO_interface.h>

GPIO Alternate function 1.

Definition at line 471 of file GPIO\_interface.h.
```

5.26.2.3 GPIOx_AF2

```
#define GPIOx_AF2 (2)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 2.

Definition at line [478](#) of file [GPIO_interface.h](#).

5.26.2.4 GPIOx_AF3

```
#define GPIOx_AF3 (3)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 3.

Definition at line [485](#) of file [GPIO_interface.h](#).

5.26.2.5 GPIOx_AF4

```
#define GPIOx_AF4 (4)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 4.

Definition at line [492](#) of file [GPIO_interface.h](#).

5.26.2.6 GPIOx_AF5

```
#define GPIOx_AF5 (5)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 5.

Definition at line [499](#) of file [GPIO_interface.h](#).

5.26.2.7 GPIOx_AF6

```
#define GPIOx_AF6 (6)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 6.

Definition at line 506 of file [GPIO_interface.h](#).

5.26.2.8 GPIOx_AF7

```
#define GPIOx_AF7 (7)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 7.

Definition at line 513 of file [GPIO_interface.h](#).

5.26.2.9 GPIOx_AF8

```
#define GPIOx_AF8 (8)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 8.

Definition at line 520 of file [GPIO_interface.h](#).

5.26.2.10 GPIOx_AF9

```
#define GPIOx_AF9 (9)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 9.

Definition at line 527 of file [GPIO_interface.h](#).

5.26.2.11 GPIOx_AF10

```
#define GPIOx_AF10 (10)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 10.

Definition at line 534 of file [GPIO_interface.h](#).

5.26.2.12 GPIOx_AF11

```
#define GPIOx_AF11 (11)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 11.

Definition at line 541 of file [GPIO_interface.h](#).

5.26.2.13 GPIOx_AF12

```
#define GPIOx_AF12 (12)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 12.

Definition at line 548 of file [GPIO_interface.h](#).

5.26.2.14 GPIOx_AF13

```
#define GPIOx_AF13 (13)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 13.

Definition at line 555 of file [GPIO_interface.h](#).

5.26.2.15 GPIOx_AF14

```
#define GPIOx_AF14 (14)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 14.

Definition at line 562 of file [GPIO_interface.h](#).

5.26.2.16 GPIOx_AF15

```
#define GPIOx_AF15 (15)

#include <COTS/MCAL/GPIO/GPIO_interface.h>
```

GPIO Alternate function 15.

Definition at line 569 of file [GPIO_interface.h](#).

5.27 GPIO Addresses

Macros

- #define GPIOA_BASE_ADDRESS (0x40020000)
- #define GPIOB_BASE_ADDRESS (0x40020400)
- #define GPIOC_BASE_ADDRESS (0x40020800)

5.27.1 Detailed Description

5.27.2 Macro Definition Documentation

5.27.2.1 GPIOA_BASE_ADDRESS

```
#define GPIOA_BASE_ADDRESS (0x40020000)

#include <COTS/MCAL/GPIO/GPIO_private.h>
```

Port A Base address

Definition at line 83 of file [GPIO_private.h](#).

5.27.2.2 GPIOB_BASE_ADDRESS

```
#define GPIOB_BASE_ADDRESS (0x40020400)

#include <COTS/MCAL/GPIO/GPIO_private.h>
```

Port B Base address

Definition at line 90 of file [GPIO_private.h](#).

5.27.2.3 GPIOC_BASE_ADDRESS

```
#define GPIOC_BASE_ADDRESS (0x40020800)

#include <COTS/MCAL/GPIO/GPIO_private.h>

Port C Base address

Definition at line 97 of file GPIO\_private.h.
```

5.28 GPIO Registers

Macros

- #define GPIOA ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOA_BASE_ADDRESS))
- #define GPIOB ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOB_BASE_ADDRESS))
- #define GPIOC ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOC_BASE_ADDRESS))

5.28.1 Detailed Description

5.28.2 Macro Definition Documentation

5.28.2.1 GPIOA

```
#define GPIOA ((volatile P2VAR(GPIOx_MemoryMapType)) (GPIOA_BASE_ADDRESS))

#include <COTS/MCAL/GPIO/GPIO_private.h>

GPIO register for port A

Definition at line 112 of file GPIO\_private.h.
```

5.28.2.2 GPIOB

```
#define GPIOB ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOB_BASE_ADDRESS))
```

```
#include <COTS/MCAL/GPIO/GPIO_private.h>
```

GPIO register for port B

Definition at line 119 of file [GPIO_private.h](#).

5.28.2.3 GPIOC

```
#define GPIOC ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOC_BASE_ADDRESS))
```

```
#include <COTS/MCAL/GPIO/GPIO_private.h>
```

GPIO register for port C

Definition at line 126 of file [GPIO_private.h](#).

5.29 Interrupt priority grouping

Interrupt priority grouping values.

Macros

- `#define _16GROUP_NoSub_Priorities (0b011)`
16 groups and 0 sub-groups
- `#define _8GROUP_2Sub_Priorities (0b100)`
8 groups and 2 sub-groups
- `#define _4GROUP_4Sub_Priorities (0b101)`
4 groups and 4 sub-groups
- `#define _2GROUP_8Sub_Priorities (0b110)`
2 groups and 8 sub-groups
- `#define NoGROUP_16Sub_Priorities (0b111)`
0 groups and 16 sub-groups
- `#define GROUP_4BITS (0b011)`
16 groups and 0 sub-groups
- `#define GROUP_3BITS (0b100)`
8 groups and 2 sub-groups
- `#define GROUP_2BITS (0b101)`
4 groups and 4 sub-groups
- `#define GROUP_1BITS (0b110)`
2 groups and 8 sub-groups
- `#define GROUP_0BITS (0b111)`
0 groups and 16 sub-groups

5.29.1 Detailed Description

Interrupt priority grouping values.

Interrupt priority grouping values for PRIGROUP in AIRCR register

See also

[SCB_MemoryMapType::AIRCR](#)

5.29.2 Macro Definition Documentation

5.29.2.1 _16GROUP_NoSub_Priorities

```
#define _16GROUP_NoSub_Priorities (0b011)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>  
  
16 groups and 0 sub-groups
```

This tells the system there is going to be 16 groups and 0 sub-groups

See also

[GROUP_4BITS](#)

Definition at line [97](#) of file [NVIC_interface.h](#).

5.29.2.2 _8GROUP_2Sub_Priorities

```
#define _8GROUP_2Sub_Priorities (0b100)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>  
  
8 groups and 2 sub-groups
```

This tells the system there is going to be 8 groups and 2 sub-groups

See also

[GROUP_3BITS](#)

Definition at line [105](#) of file [NVIC_interface.h](#).

5.29.2.3 _4GROUP_4Sub_Priorities

```
#define _4GROUP_4Sub_Priorities (0b101)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>  
  
4 groups and 4 sub-groups
```

This tells the system there is going to be 4 groups and 4 sub-groups

See also

[GROUP_2BITS](#)

Definition at line 113 of file [NVIC_interface.h](#).

5.29.2.4 _2GROUP_8Sub_Priorities

```
#define _2GROUP_8Sub_Priorities (0b110)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>  
  
2 groups and 8 sub-groups
```

This tells the system there is going to be 2 groups and 8 sub-groups

See also

[GROUP_1BITS](#)

Definition at line 121 of file [NVIC_interface.h](#).

5.29.2.5 NoGROUP_16Sub_Priorities

```
#define NoGROUP_16Sub_Priorities (0b111)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>  
  
0 groups and 16 sub-groups
```

This tells the system there is going to be 0 groups and 16 sub-groups

See also

[GROUP_0BITS](#)

Definition at line 129 of file [NVIC_interface.h](#).

5.29.2.6 GROUP_4BITS

```
#define GROUP_4BITS (0b011)

#include <COTS/MCAL/NVIC/NVIC_interface.h>

16 groups and 0 sub-groups

This tells the system there is going to be 16 groups and 0 sub-groups

Definition at line 147 of file NVIC\_interface.h.
```

5.29.2.7 GROUP_3BITS

```
#define GROUP_3BITS (0b100)

#include <COTS/MCAL/NVIC/NVIC_interface.h>

8 groups and 2 sub-groups

This tells the system there is going to be 8 groups and 2 sub-groups

Definition at line 154 of file NVIC\_interface.h.
```

5.29.2.8 GROUP_2BITS

```
#define GROUP_2BITS (0b101)

#include <COTS/MCAL/NVIC/NVIC_interface.h>

4 groups and 4 sub-groups

This tells the system there is going to be 4 groups and 4 sub-groups

Definition at line 161 of file NVIC\_interface.h.
```

5.29.2.9 GROUP_1BITS

```
#define GROUP_1BITS (0b110)

#include <COTS/MCAL/NVIC/NVIC_interface.h>

2 groups and 8 sub-groups

This tells the system there is going to be 2 groups and 8 sub-groups

Definition at line 168 of file NVIC\_interface.h.
```

5.29.2.10 GROUP_0BITS

```
#define GROUP_0BITS (0b111)

#include <COTS/MCAL/NVIC/NVIC_interface.h>

0 groups and 16 sub-groups
```

This tells the system there is going to be 0 groups and 16 sub-groups

Definition at line 175 of file [NVIC_interface.h](#).

5.30 Group priorities

The group priority values.

Macros

- #define NO_GROUP_PRIORITY (0)
No group priority.
- #define GROUP_PRIORITY_0 (0)
Priority: 0.
- #define GROUP_PRIORITY_1 (1)
Priority: 1.
- #define GROUP_PRIORITY_2 (2)
Priority: 2.
- #define GROUP_PRIORITY_3 (3)
Priority: 3.
- #define GROUP_PRIORITY_4 (4)
Priority: 4.
- #define GROUP_PRIORITY_5 (5)
Priority: 5.
- #define GROUP_PRIORITY_6 (6)
Priority: 6.
- #define GROUP_PRIORITY_7 (7)
Priority: 7.
- #define GROUP_PRIORITY_8 (8)
Priority: 8.
- #define GROUP_PRIORITY_9 (9)
Priority: 9.
- #define GROUP_PRIORITY_10 (10)
Priority: 10.
- #define GROUP_PRIORITY_11 (11)
Priority: 11.
- #define GROUP_PRIORITY_12 (12)
Priority: 12.
- #define GROUP_PRIORITY_13 (13)
Priority: 13.
- #define GROUP_PRIORITY_14 (14)
Priority: 14.
- #define GROUP_PRIORITY_15 (15)
Priority: 15.

5.30.1 Detailed Description

The group priority values.

See also

[MNVIC_vSetPriority](#)

[Sub-Group priorities](#)

5.30.2 Macro Definition Documentation

5.30.2.1 NO_GROUP_PRIORITY

```
#define NO_GROUP_PRIORITY (0)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

No group priority.

Definition at line [189](#) of file [NVIC_interface.h](#).

5.30.2.2 GROUP_PRIORITY_0

```
#define GROUP_PRIORITY_0 (0)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 0.

Definition at line [194](#) of file [NVIC_interface.h](#).

5.30.2.3 GROUP_PRIORITY_1

```
#define GROUP_PRIORITY_1 (1)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 1.

Definition at line [199](#) of file [NVIC_interface.h](#).

5.30.2.4 GROUP_PRIORITY_2

```
#define GROUP_PRIORITY_2 (2)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 2.

Definition at line 204 of file [NVIC_interface.h](#).

5.30.2.5 GROUP_PRIORITY_3

```
#define GROUP_PRIORITY_3 (3)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 3.

Definition at line 209 of file [NVIC_interface.h](#).

5.30.2.6 GROUP_PRIORITY_4

```
#define GROUP_PRIORITY_4 (4)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 4.

Definition at line 214 of file [NVIC_interface.h](#).

5.30.2.7 GROUP_PRIORITY_5

```
#define GROUP_PRIORITY_5 (5)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 5.

Definition at line 219 of file [NVIC_interface.h](#).

5.30.2.8 GROUP_PRIORITY_6

```
#define GROUP_PRIORITY_6 (6)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 6.

Definition at line 224 of file [NVIC_interface.h](#).

5.30.2.9 GROUP_PRIORITY_7

```
#define GROUP_PRIORITY_7 (7)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 7.

Definition at line 229 of file [NVIC_interface.h](#).

5.30.2.10 GROUP_PRIORITY_8

```
#define GROUP_PRIORITY_8 (8)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 8.

Definition at line 234 of file [NVIC_interface.h](#).

5.30.2.11 GROUP_PRIORITY_9

```
#define GROUP_PRIORITY_9 (9)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 9.

Definition at line 239 of file [NVIC_interface.h](#).

5.30.2.12 GROUP_PRIORITY_10

```
#define GROUP_PRIORITY_10 (10)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 10.

Definition at line [244](#) of file [NVIC_interface.h](#).

5.30.2.13 GROUP_PRIORITY_11

```
#define GROUP_PRIORITY_11 (11)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 11.

Definition at line [249](#) of file [NVIC_interface.h](#).

5.30.2.14 GROUP_PRIORITY_12

```
#define GROUP_PRIORITY_12 (12)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 12.

Definition at line [254](#) of file [NVIC_interface.h](#).

5.30.2.15 GROUP_PRIORITY_13

```
#define GROUP_PRIORITY_13 (13)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 13.

Definition at line [259](#) of file [NVIC_interface.h](#).

5.30.2.16 GROUP_PRIORITY_14

```
#define GROUP_PRIORITY_14 (14)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 14.

Definition at line 264 of file [NVIC_interface.h](#).

5.30.2.17 GROUP_PRIORITY_15

```
#define GROUP_PRIORITY_15 (15)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 15.

Definition at line 269 of file [NVIC_interface.h](#).

5.31 Sub-Group priorities

The sub-group priority values.

Macros

- `#define NO_SUB_PRIORITY (0)`
No sub-group priority.
- `#define SUB_PRIORITY_0 (0)`
Priority: 0.
- `#define SUB_PRIORITY_1 (1)`
Priority: 1.
- `#define SUB_PRIORITY_2 (2)`
Priority: 2.
- `#define SUB_PRIORITY_3 (3)`
Priority: 3.
- `#define SUB_PRIORITY_4 (4)`
Priority: 4.
- `#define SUB_PRIORITY_5 (5)`
Priority: 5.
- `#define SUB_PRIORITY_6 (6)`
Priority: 6.
- `#define SUB_PRIORITY_7 (7)`
Priority: 7.
- `#define SUB_PRIORITY_8 (8)`
Priority: 8.
- `#define SUB_PRIORITY_9 (9)`

- #define SUB_PRIORITY_10 (10)
Priority: 10.
- #define SUB_PRIORITY_11 (11)
Priority: 11.
- #define SUB_PRIORITY_12 (12)
Priority: 12.
- #define SUB_PRIORITY_13 (13)
Priority: 13.
- #define SUB_PRIORITY_14 (14)
Priority: 14.
- #define SUB_PRIORITY_15 (15)
Priority: 15.

5.31.1 Detailed Description

The sub-group priority values.

See also

- [MNVIC_vSetPriority](#)
- [Group priorities](#)

5.31.2 Macro Definition Documentation

5.31.2.1 NO_SUB_PRIORITY

```
#define NO_SUB_PRIORITY (0)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

No sub-group priority.

Definition at line 285 of file [NVIC_interface.h](#).

5.31.2.2 SUB_PRIORITY_0

```
#define SUB_PRIORITY_0 (0)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 0.

Definition at line 290 of file [NVIC_interface.h](#).

5.31.2.3 SUB_PRIORITY_1

```
#define SUB_PRIORITY_1 (1)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 1.

Definition at line 295 of file [NVIC_interface.h](#).

5.31.2.4 SUB_PRIORITY_2

```
#define SUB_PRIORITY_2 (2)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 2.

Definition at line 300 of file [NVIC_interface.h](#).

5.31.2.5 SUB_PRIORITY_3

```
#define SUB_PRIORITY_3 (3)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 3.

Definition at line 305 of file [NVIC_interface.h](#).

5.31.2.6 SUB_PRIORITY_4

```
#define SUB_PRIORITY_4 (4)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 4.

Definition at line 310 of file [NVIC_interface.h](#).

5.31.2.7 SUB_PRIORITY_5

```
#define SUB_PRIORITY_5 (5)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 5.

Definition at line 315 of file [NVIC_interface.h](#).

5.31.2.8 SUB_PRIORITY_6

```
#define SUB_PRIORITY_6 (6)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 6.

Definition at line 320 of file [NVIC_interface.h](#).

5.31.2.9 SUB_PRIORITY_7

```
#define SUB_PRIORITY_7 (7)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 7.

Definition at line 325 of file [NVIC_interface.h](#).

5.31.2.10 SUB_PRIORITY_8

```
#define SUB_PRIORITY_8 (8)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 8.

Definition at line 330 of file [NVIC_interface.h](#).

5.31.2.11 SUB_PRIORITY_9

```
#define SUB_PRIORITY_9 (9)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 9.

Definition at line 335 of file [NVIC_interface.h](#).

5.31.2.12 SUB_PRIORITY_10

```
#define SUB_PRIORITY_10 (10)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 10.

Definition at line 340 of file [NVIC_interface.h](#).

5.31.2.13 SUB_PRIORITY_11

```
#define SUB_PRIORITY_11 (11)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 11.

Definition at line 345 of file [NVIC_interface.h](#).

5.31.2.14 SUB_PRIORITY_12

```
#define SUB_PRIORITY_12 (12)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 12.

Definition at line 350 of file [NVIC_interface.h](#).

5.31.2.15 SUB_PRIORITY_13

```
#define SUB_PRIORITY_13 (13)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 13.

Definition at line 355 of file [NVIC_interface.h](#).

5.31.2.16 SUB_PRIORITY_14

```
#define SUB_PRIORITY_14 (14)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 14.

Definition at line 360 of file [NVIC_interface.h](#).

5.31.2.17 SUB_PRIORITY_15

```
#define SUB_PRIORITY_15 (15)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Priority: 15.

Definition at line 365 of file [NVIC_interface.h](#).

5.32 NVIC Vector Table

NVIC vector table.

Macros

- #define **WWDG** (0)
Window Watchdog (WWDG) Interrupt.
- #define **EXTI16** (1)
EXTI Line 16 interrupt / PVD through EXTI line detection interrupt.
- #define **EXTI21** (2)
EXTI Line 21 interrupt / Tamper andTimeStamp interrupts through the EXTI line.
- #define **EXTI22** (3)
EXTI Line 22 interrupt / RTC (Real-time clock) wakeup interrupt through the EXTI line.
- #define **FLASH** (4)
Flash global interrupt.
- #define **RCC** (5)
RCC global interrupt.
- #define **EXTI0** (6)
EXTI Line 0 interrupt.
- #define **EXTI1** (7)
EXTI Line 1 interrupt.
- #define **EXTI2** (8)
EXTI Line 2 interrupt.
- #define **EXTI3** (9)
EXTI Line 3 interrupt.
- #define **EXTI4** (10)
EXTI Line 4 interrupt.
- #define **DMA1_STREAM0** (11)
DMA1 (Direct Memory Access) Stream 0 global interrupt.
- #define **DMA1_STREAM1** (12)
DMA1 (Direct Memory Access) Stream 1 global interrupt.
- #define **DMA1_STREAM2** (13)
DMA1 (Direct Memory Access) Stream 2 global interrupt.
- #define **DMA1_STREAM3** (14)
DMA1 (Direct Memory Access) Stream 3 global interrupt.
- #define **DMA1_STREAM4** (15)
DMA1 (Direct Memory Access) Stream 4 global interrupt.
- #define **DMA1_STREAM5** (16)
DMA1 (Direct Memory Access) Stream 5 global interrupt.
- #define **DMA1_STREAM6** (17)
DMA1 (Direct Memory Access) Stream 6 global interrupt.
- #define **ADC** (18)
ADC (Analog-To-Digital Converter) ADC1 global interrupt.
- #define **EXTI9** (23)
EXTI Lines [9:5] interrupts.
- #define **TIM1_BRK_TIM9** (24)
TIM1 (Timer 1) break interrupt and TIM9 (Timer 9) global interrupt.
- #define **TIM1_UP_TIM10** (25)
TIM1 (Timer 1) update interrupt and TIM10 (Timer 10) global interrupt.
- #define **TIM1_TRG_COM_TIM11** (26)
TIM1 (Timer 1) trigger and commutation itnerrups and TIM11 (Timer 11) global interrupts.
- #define **TIM1_CC** (27)
TIM1 (Timer 1) capture compare interrupt.
- #define **TIM2** (28)

- #define **TIM2** (29)
TIM2 (Timer 2) global interrupt.
- #define **TIM3** (29)
TIM3 (Timer 3) global interrupt.
- #define **TIM4** (30)
TIM4 (Timer 4) global interrupt.
- #define **I2C1_EV** (31)
I2C1 (Inter-integrated Circuit 1) event interrupt.
- #define **I2C1_ER** (32)
I2C1 (Inter-integrated Circuit 1) error interrupt.
- #define **I2C2_EV** (33)
I2C2 (Inter-integrated Circuit 2) event interrupt.
- #define **I2C2_ER** (34)
I2C2 (Inter-integrated Circuit 2) error interrupt.
- #define **SPI1** (35)
SPI1 (Serial Peripheral Interface 1) global interrupt.
- #define **SPI2** (36)
SPI2 (Serial Peripheral Interface 2) global interrupt.
- #define **USART1** (37)
USART1 (Universal Synchronous/Asynchronous Receiver/Transmitter 1) global interrupt.
- #define **USART2** (38)
USART2 (Universal Synchronous/Asynchronous Receiver/Transmitter 2) global interrupt.
- #define **EXTI15_10** (30)
EXTI Line[15:10] interrupts.
- #define **EXTI17** (41)
EXTI Line 17 interrupt / RTC Alarms (A and B) through EXTI line interrupt.
- #define **EXTI18** (42)
EXTI Line 18 interrupt / USB On-The-Go FS Wakeup through EXTI line interrupt.
- #define **DMA1_STREAM7** (47)
DMA1 (Direct Memory Access) Stream 7 global interrupt.
- #define **SDIO** (49)
SDIO (Secure Digital Input Output) global interrupt.
- #define **TIM5** (50)
TIM5 (Timer 5) global interrupt.
- #define **SPI3** (51)
SPI3 (Serial Peripheral Interface 3) global interrupt.
- #define **DMA2_STREAM0** (56)
DMA2 (Direct Memory Access 2) Stream 0 global interrupt.
- #define **DMA2_STREAM1** (57)
DMA2 (Direct Memory Access 2) Stream 1 global interrupt.
- #define **DMA2_STREAM2** (58)
DMA2 (Direct Memory Access 2) Stream 2 global interrupt.
- #define **DMA2_STREAM3** (59)
DMA2 (Direct Memory Access 2) Stream 3 global interrupt.
- #define **DMA2_STREAM4** (60)
DMA2 (Direct Memory Access 2) Stream 4 global interrupt.
- #define **OTG_FS** (67)
USB On The Go FS global interrupt.
- #define **DMA2_STREAM5** (68)
DMA2 (Direct Memory Access 2) Stream 5 global interrupt.
- #define **DMA2_STREAM6** (69)
DMA2 (Direct Memory Access 2) Stream 6 global interrupt.

- #define DMA2_STREAM7 (70)
DMA2 (Direct Memory Access 2) Steam 7 global interrupt.
- #define USART6 (71)
USART6 (Universal Synchronous/Asynchronous Receiver/Transmitter 6) global interrupt.
- #define I2C3_EV (72)
I2C3 (Inter-integrated Circuit 3) event interrupt.
- #define I2C3_ER (73)
I2C3 (Inter-integrated Circuit 3) error interrupt.
- #define FPU (81)
FPU (Floating Point Unit) global interrupt.
- #define SPI4 (84)
SPI4 (Serial Peripheral Interface 4) global interrupt.

5.32.1 Detailed Description

NVIC vector table.

NVIC vector table that contains each peripheral's priority.

The lower the value, the higher the priority

5.32.2 Macro Definition Documentation

5.32.2.1 WWDG

```
#define WWDG (0)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Window Watchdog (WWDG) Interrupt.

Definition at line 379 of file [NVIC_interface.h](#).

5.32.2.2 EXTI16

```
#define EXTI16 (1)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 16 interrupt / PVD through EXTI line detection interrupt.

Definition at line 384 of file [NVIC_interface.h](#).

5.32.2.3 EXTI21

```
#define EXTI21 (2)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 21 interrupt / Tamper and TimeStamp interrupts through the EXTI line.

Definition at line [389](#) of file [NVIC_interface.h](#).

5.32.2.4 EXTI22

```
#define EXTI22 (3)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 22 interrupt / RTC (Real-time clock) wakeup interrupt through the EXTI line.

Definition at line [394](#) of file [NVIC_interface.h](#).

5.32.2.5 FLASH

```
#define FLASH (4)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

Flash global interrupt.

Definition at line [399](#) of file [NVIC_interface.h](#).

5.32.2.6 RCC

```
#define RCC (5)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

RCC global interrupt.

Definition at line [404](#) of file [NVIC_interface.h](#).

5.32.2.7 EXTI0

```
#define EXTI0 (6)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 0 interrupt.

Definition at line [409](#) of file [NVIC_interface.h](#).

5.32.2.8 EXTI1

```
#define EXTI1 (7)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 1 interrupt.

Definition at line [414](#) of file [NVIC_interface.h](#).

5.32.2.9 EXTI2

```
#define EXTI2 (8)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 2 interrupt.

Definition at line [419](#) of file [NVIC_interface.h](#).

5.32.2.10 EXTI3

```
#define EXTI3 (9)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 3 interrupt.

Definition at line [424](#) of file [NVIC_interface.h](#).

5.32.2.11 EXTI4

```
#define EXTI4 (10)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 4 interrupt.

Definition at line [429](#) of file [NVIC_interface.h](#).

5.32.2.12 DMA1_STREAM0

```
#define DMA1_STREAM0 (11)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 0 global interrupt.

Definition at line [434](#) of file [NVIC_interface.h](#).

5.32.2.13 DMA1_STREAM1

```
#define DMA1_STREAM1 (12)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 1 global interrupt.

Definition at line [439](#) of file [NVIC_interface.h](#).

5.32.2.14 DMA1_STREAM2

```
#define DMA1_STREAM2 (13)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 2 global interrupt.

Definition at line [444](#) of file [NVIC_interface.h](#).

5.32.2.15 DMA1_STREAM3

```
#define DMA1_STREAM3 (14)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 3 global interrupt.

Definition at line [449](#) of file [NVIC_interface.h](#).

5.32.2.16 DMA1_STREAM4

```
#define DMA1_STREAM4 (15)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 4 global interrupt.

Definition at line [454](#) of file [NVIC_interface.h](#).

5.32.2.17 DMA1_STREAM5

```
#define DMA1_STREAM5 (16)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 5 global interrupt.

Definition at line [459](#) of file [NVIC_interface.h](#).

5.32.2.18 DMA1_STREAM6

```
#define DMA1_STREAM6 (17)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Steam 6 global interrupt.

Definition at line [464](#) of file [NVIC_interface.h](#).

5.32.2.19 ADC

```
#define ADC (18)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

ADC (Analog-To-Digital Converter) ADC1 global interrupt.

Definition at line [469](#) of file [NVIC_interface.h](#).

5.32.2.20 EXTI9

```
#define EXTI9 (23)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Lines [9:5] interrupts.

Definition at line [474](#) of file [NVIC_interface.h](#).

5.32.2.21 TIM1_BRK_TIM9

```
#define TIM1_BRK_TIM9 (24)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM1 (Timer 1) break interrupt and TIM9 (Timer 9) global interrupt.

Definition at line [479](#) of file [NVIC_interface.h](#).

5.32.2.22 TIM1_UP_TIM10

```
#define TIM1_UP_TIM10 (25)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM1 (Timer 1) update interrupt and TIM10 (Timer 10) global interrupt.

Definition at line [484](#) of file [NVIC_interface.h](#).

5.32.2.23 TIM1_TRG_COM_TIM11

```
#define TIM1_TRG_COM_TIM11 (26)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM1 (Timer 1) trigger and commutation itnerrupts and TIM11 (Timer 11) global interrupts.

Definition at line [489](#) of file [NVIC_interface.h](#).

5.32.2.24 TIM1_CC

```
#define TIM1_CC (27)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM1 (Timer 1) capture compare interrupt.

Definition at line [494](#) of file [NVIC_interface.h](#).

5.32.2.25 TIM2

```
#define TIM2 (28)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM2 (Timer 2) global interrupt.

Definition at line [499](#) of file [NVIC_interface.h](#).

5.32.2.26 TIM3

```
#define TIM3 (29)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM3 (Timer 3) global interrupt.

Definition at line [504](#) of file [NVIC_interface.h](#).

5.32.2.27 TIM4

```
#define TIM4 (30)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM4 (Timer 4) global interrupt.

Definition at line [509](#) of file [NVIC_interface.h](#).

5.32.2.28 I2C1_EV

```
#define I2C1_EV (31)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C1 (Inter-integrated Circuit 1) event interrupt.

Definition at line [514](#) of file [NVIC_interface.h](#).

5.32.2.29 I2C1_ER

```
#define I2C1_ER (32)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C1 (Inter-integrated Circuit 1) error interrupt.

Definition at line [519](#) of file [NVIC_interface.h](#).

5.32.2.30 I2C2_EV

```
#define I2C2_EV (33)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C2 (Inter-integrated Circuit 2) event interrupt.

Definition at line [524](#) of file [NVIC_interface.h](#).

5.32.2.31 I2C2_ER

```
#define I2C2_ER (34)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C2 (Inter-integrated Circuit 2) error interrupt.

Definition at line [529](#) of file [NVIC_interface.h](#).

5.32.2.32 SPI1

```
#define SPI1 (35)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

SPI1 (Serial Peripheral Interface 1) global interrupt.

Definition at line [534](#) of file [NVIC_interface.h](#).

5.32.2.33 SPI2

```
#define SPI2 (36)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

SPI2 (Serial Peripheral Interface 2) global interrupt.

Definition at line [539](#) of file [NVIC_interface.h](#).

5.32.2.34 USART1

```
#define USART1 (37)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

USART1 (Universal Synchronous/Asynchronous Receiver/Transmitter 1) global interrupt.

Definition at line [544](#) of file [NVIC_interface.h](#).

5.32.2.35 USART2

```
#define USART2 (38)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

USART2 (Universal Synchronous/Asynchronous Receiver/Transmitter 2) global interrupt.

Definition at line [549](#) of file [NVIC_interface.h](#).

5.32.2.36 EXTI15_10

```
#define EXTI15_10 (30)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line[15:10] interrupts.

Definition at line [554](#) of file [NVIC_interface.h](#).

5.32.2.37 EXTI17

```
#define EXTI17 (41)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 17 interrupt / RTC Alarms (A and B) through EXTI line interrupt.

Definition at line [559](#) of file [NVIC_interface.h](#).

5.32.2.38 EXTI18

```
#define EXTI18 (42)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

EXTI Line 18 interrupt / USB On-The-Go FS Wakeup through EXTI line interrupt.

Definition at line [564](#) of file [NVIC_interface.h](#).

5.32.2.39 DMA1_STREAM7

```
#define DMA1_STREAM7 (47)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA1 (Direct Memory Access) Stream 7 global interrupt.

Definition at line [569](#) of file [NVIC_interface.h](#).

5.32.2.40 SDIO

```
#define SDIO (49)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

SDIO (Secure Digital Input Output) global interrupt.

Definition at line [574](#) of file [NVIC_interface.h](#).

5.32.2.41 TIM5

```
#define TIM5 (50)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

TIM5 (Timer 5) global interrupt.

Definition at line [579](#) of file [NVIC_interface.h](#).

5.32.2.42 SPI3

```
#define SPI3 (51)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

SPI3 (Serial Peripheral Interface 3) global interrupt.

Definition at line [584](#) of file [NVIC_interface.h](#).

5.32.2.43 DMA2_STREAM0

```
#define DMA2_STREAM0 (56)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 0 global interrupt.

Definition at line 589 of file [NVIC_interface.h](#).

5.32.2.44 DMA2_STREAM1

```
#define DMA2_STREAM1 (57)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 1 global interrupt.

Definition at line 594 of file [NVIC_interface.h](#).

5.32.2.45 DMA2_STREAM2

```
#define DMA2_STREAM2 (58)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 2 global interrupt.

Definition at line 599 of file [NVIC_interface.h](#).

5.32.2.46 DMA2_STREAM3

```
#define DMA2_STREAM3 (59)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 3 global interrupt.

Definition at line 604 of file [NVIC_interface.h](#).

5.32.2.47 DMA2_STREAM4

```
#define DMA2_STREAM4 (60)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 4 global interrupt.

Definition at line [609](#) of file [NVIC_interface.h](#).

5.32.2.48 OTG_FS

```
#define OTG_FS (67)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

USB On The Go FS global interrupt.

Definition at line [614](#) of file [NVIC_interface.h](#).

5.32.2.49 DMA2_STREAM5

```
#define DMA2_STREAM5 (68)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 5 global interrupt.

Definition at line [619](#) of file [NVIC_interface.h](#).

5.32.2.50 DMA2_STREAM6

```
#define DMA2_STREAM6 (69)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Steam 6 global interrupt.

Definition at line [624](#) of file [NVIC_interface.h](#).

5.32.2.51 DMA2_STREAM7

```
#define DMA2_STREAM7 (70)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

DMA2 (Direct Memory Access 2) Stream 7 global interrupt.

Definition at line [629](#) of file [NVIC_interface.h](#).

5.32.2.52 USART6

```
#define USART6 (71)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

USART6 (Universal Synchronous/Asynchronous Receiver/Transmitter 6) global interrupt.

Definition at line [634](#) of file [NVIC_interface.h](#).

5.32.2.53 I2C3_EV

```
#define I2C3_EV (72)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C3 (Inter-integrated Circuit 3) event interrupt.

Definition at line [639](#) of file [NVIC_interface.h](#).

5.32.2.54 I2C3_ER

```
#define I2C3_ER (73)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C3 (Inter-integrated Circuit 3) error interrupt.

Definition at line [644](#) of file [NVIC_interface.h](#).

5.32.2.55 FPU

```
#define FPU (81)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

FPU (Floating Point Unit) global interrupt.

Definition at line [649](#) of file [NVIC_interface.h](#).

5.32.2.56 SPI4

```
#define SPI4 (84)

#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

SPI4 (Serial Peripheral Interface 4) global interrupt.

Definition at line [654](#) of file [NVIC_interface.h](#).

5.33 NVIC Addresses

NVIC registers base addresses.

Macros

- #define [NVIC_BASE_ADDRESS](#) (0xE000E100)
NVIC base address.
- #define [SCB_BASE_ADDRESS](#) (0xE000ED00)
SCB base address.

5.33.1 Detailed Description

NVIC registers base addresses.

5.33.2 Macro Definition Documentation

5.33.2.1 NVIC_BASE_ADDRESS

```
#define NVIC_BASE_ADDRESS (0xE000E100)  
#include <COTS/MCAL/NVIC/NVIC_private.h>
```

NVIC base address.

Definition at line 212 of file [NVIC_private.h](#).

5.33.2.2 SCB_BASE_ADDRESS

```
#define SCB_BASE_ADDRESS (0xE000ED00)  
#include <COTS/MCAL/NVIC/NVIC_private.h>
```

SCB base address.

Definition at line 219 of file [NVIC_private.h](#).

5.34 NVIC Registers

NVIC registers.

Macros

- #define MSCB ((volatile P2VAR(SCB_MemoryMapType))(SCB_BASE_ADDRESS))
- #define MNVIC ((volatile P2VAR(NVIC_MemoryMapType))(NVIC_BASE_ADDRESS))

5.34.1 Detailed Description

NVIC registers.

5.34.2 Macro Definition Documentation

5.34.2.1 MSCB

```
#define MSCB ((volatile P2VAR(SCB_MemoryMapType)) (SCB_BASE_ADDRESS))  
#include <COTS/MCAL/NVIC/NVIC_private.h>
```

MSCB register

Definition at line 234 of file [NVIC_private.h](#).

5.34.2.2 MNVIC

```
#define MNVIC ((volatile P2VAR(NVIC_MemoryMapType))(NVIC_BASE_ADDRESS))
```

```
#include <COTS/MCAL/NVIC/NVIC_private.h>
```

MNVIC register

Definition at line [240](#) of file [NVIC_private.h](#).

5.35 NVIC Settable Priorities

NVIC Settable Priorities for the hardware interrupts.

Macros

- #define PEND_SV (-6)
- #define SYSTICK (-5)
- #define SV_CALL (-4)
- #define MEMORY_MANAGE (-3)
- #define BUS_FAULT (-2)
- #define USAGE_FAULT (-1)

5.35.1 Detailed Description

NVIC Settable Priorities for the hardware interrupts.

5.35.2 Macro Definition Documentation

5.35.2.1 PEND_SV

```
#define PEND_SV (-6)
```

```
#include <COTS/MCAL/NVIC/NVIC_private.h>
```

Pendable request for system service

Definition at line [266](#) of file [NVIC_private.h](#).

5.35.2.2 SYSTICK

```
#define SYSTICK (-5)

#include <COTS/MCAL/NVIC/NVIC_private.h>
```

System tick timer

Definition at line [272](#) of file [NVIC_private.h](#).

5.35.2.3 SV_CALL

```
#define SV_CALL (-4)

#include <COTS/MCAL/NVIC/NVIC_private.h>
```

System service call via SWI instruction

Definition at line [278](#) of file [NVIC_private.h](#).

5.35.2.4 MEMORY_MANAGE

```
#define MEMORY_MANAGE (-3)

#include <COTS/MCAL/NVIC/NVIC_private.h>
```

Memory management

Definition at line [284](#) of file [NVIC_private.h](#).

5.35.2.5 BUSFAULT

```
#define BUS_FAULT (-2)

#include <COTS/MCAL/NVIC/NVIC_private.h>
```

Pre-fetch fault, memory access fault

Definition at line [290](#) of file [NVIC_private.h](#).

5.35.2.6 USAGE_FAULT

```
#define USAGE_FAULT (-1)

#include <COTS/MCAL/NVIC/NVIC_private.h>
```

Undefined instruction or illegal state

Definition at line 296 of file [NVIC_private.h](#).

5.36 ID options

Macros

- #define RCC_AHB1 1
AHB1 Bus.
- #define RCC_AHB2 2
AHB2 Bus.
- #define RCC_APB1 3
APB1 Bus.
- #define RCC_APB2 4
APB2 Bus.
- #define RCC_AHB1LPENR 5
peripheral clock enable in low power mode register

5.36.1 Detailed Description

5.36.2 Macro Definition Documentation

5.36.2.1 RCC_AHB1

```
#define RCC_AHB1 1

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

AHB1 Bus.

Definition at line 51 of file [MRCC_interface.h](#).

5.36.2.2 RCC_AHB2

```
#define RCC_AHB2 2

#include <COTS/MCAL/RCC/MRCC_interface.h>

AHB2 Bus.
```

Definition at line [58](#) of file [MRCC_interface.h](#).

5.36.2.3 RCC_APB1

```
#define RCC_APB1 3

#include <COTS/MCAL/RCC/MRCC_interface.h>

APB1 Bus.
```

Definition at line [65](#) of file [MRCC_interface.h](#).

5.36.2.4 RCC_APB2

```
#define RCC_APB2 4

#include <COTS/MCAL/RCC/MRCC_interface.h>

APB2 Bus.
```

Definition at line [72](#) of file [MRCC_interface.h](#).

5.36.2.5 RCC_AHB1LPENR

```
#define RCC_AHB1LPENR 5

#include <COTS/MCAL/RCC/MRCC_interface.h>

peripheral clock enable in low power mode register

Definition at line 79 of file MRCC\_interface.h.
```

5.37 ID options

Macros

- #define AHB1ENR_DMA2EN 22
Enable CLK on DMA2 peripheral.
- #define AHB1ENR_DMA1EN 21
Enable CLK on DMA1 peripheral.
- #define AHB1ENR_CRCEN 12
Enable CLK on CRC peripheral.
- #define AHB1ENR_GPIOHEN 7
Enable CLK on GPIOH peripheral.
- #define AHB1ENR_GPIOEEN 4
Enable CLK on GPIOE peripheral.
- #define AHB1ENR_GPIODEN 3
Enable CLK on GPIOD peripheral.
- #define AHB1ENR_GPIOCEN 2
Enable CLK on GPIOC peripheral.
- #define AHB1ENR_GPIOBEN 1
Enable CLK on GPIOB peripheral.
- #define AHB1ENR_GPIOAEN 0
Enable CLK on GPIOA peripheral.
- #define AHB2ENR_OTGFSEN 7
Enable CLK on OTGFS peripheral.
- #define APB1ENR_PWREN 28
Enable CLK on PWR peripheral.
- #define APB1ENR_I2C3EN 23
Enable CLK on I2C3 peripheral.
- #define APB1ENR_I2C2EN 22
Enable CLK on I2C2 peripheral.
- #define APB1ENR_I2C1EN 21
Enable CLK on I2C1 peripheral.
- #define APB1ENR_USART2EN 17
Enable CLK on USART2 peripheral.
- #define APB1ENR_SPI3EN 15
Enable CLK on SPI3 peripheral.
- #define APB1ENR_SPI2EN 14
Enable CLK on SPI2 peripheral.
- #define APB1ENR_WWDGEN 11
Enable CLK on WWD peripheral.
- #define APB1ENR_TIM5EN 3
Enable CLK on TIM5 peripheral.
- #define APB1ENR_TIM4EN 2
Enable CLK on TIM4 peripheral.
- #define APB1ENR_TIM3EN 1
Enable CLK on TIM3 peripheral.
- #define APB1ENR_TIM2EN 0
Enable CLK on TIM2 peripheral.
- #define APB2ENR_TIM11EN 18
Enable CLK on TIM11 peripheral.

- #define APB2ENR_TIM10EN 17
Enable CLK on TIM10 peripheral.
- #define APB2ENR_TIM9EN 16
Enable CLK on TIM9 peripheral.
- #define APB2ENR_SYSCFGEN 14
Enable CLK on SYSCFG peripheral.
- #define APB2ENR_SPI4EN 13
Enable CLK on SPI4 peripheral.
- #define APB2ENR_SPI1EN 12
Enable CLK on SPI1 peripheral.
- #define APB2ENR_SDIOEN 11
Enable CLK on SDIO peripheral.
- #define APB2ENR_ADC1EN 8
Enable CLK on ADC1 peripheral.
- #define APB2ENR_USART6EN 5
Enable CLK on USART6 peripheral.
- #define APB2ENR_USART1EN 4
Enable CLK on USART1 peripheral.
- #define APB2ENR_TIM1EN 0
Enable CLK on TIM1 peripheral.
- #define AHB1LPENR_FLITFLPEN 15
Enable CLK on FLITFLP peripheral.

5.37.1 Detailed Description

5.37.2 Macro Definition Documentation

5.37.2.1 AHB1ENR_DMA2EN

```
#define AHB1ENR_DMA2EN 22

#include <COTS/MCAL/RCC/MRCC_interface.h>

Enable CLK on DMA2 peripheral.

Definition at line 94 of file MRCC_interface.h.
```

5.37.2.2 AHB1ENR_DMA1EN

```
#define AHB1ENR_DMA1EN 21

#include <COTS/MCAL/RCC/MRCC_interface.h>

Enable CLK on DMA1 peripheral.

Definition at line 101 of file MRCC_interface.h.
```

5.37.2.3 AHB1ENR_CRCEN

```
#define AHB1ENR_CRCEN 12

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on CRC peripheral.

Definition at line 108 of file [MRCC_interface.h](#).

5.37.2.4 AHB1ENR_GPIOHEN

```
#define AHB1ENR_GPIOHEN 7

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOH peripheral.

Definition at line 115 of file [MRCC_interface.h](#).

5.37.2.5 AHB1ENR_GPIOEEN

```
#define AHB1ENR_GPIOEEN 4

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOE peripheral.

Definition at line 122 of file [MRCC_interface.h](#).

5.37.2.6 AHB1ENR_GPIODEN

```
#define AHB1ENR_GPIODEN 3

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOD peripheral.

Definition at line 129 of file [MRCC_interface.h](#).

5.37.2.7 AHB1ENR_GPIOCEN

```
#define AHB1ENR_GPIOCEN 2

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOC peripheral.

Definition at line 136 of file [MRCC_interface.h](#).

5.37.2.8 AHB1ENR_GPIOBEN

```
#define AHB1ENR_GPIOBEN 1

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOB peripheral.

Definition at line 143 of file [MRCC_interface.h](#).

5.37.2.9 AHB1ENR_GPIOAEN

```
#define AHB1ENR_GPIOAEN 0

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on GPIOA peripheral.

Definition at line 150 of file [MRCC_interface.h](#).

5.37.2.10 AHB2ENR_OTGFSEN

```
#define AHB2ENR_OTGFSEN 7

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on OTGFS peripheral.

Definition at line 157 of file [MRCC_interface.h](#).

5.37.2.11 APB1ENR_PWREN

```
#define APB1ENR_PWREN 28  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on PWR peripheral.

Definition at line 164 of file [MRCC_interface.h](#).

5.37.2.12 APB1ENR_I2C3EN

```
#define APB1ENR_I2C3EN 23  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on I2C3 peripheral.

Definition at line 171 of file [MRCC_interface.h](#).

5.37.2.13 APB1ENR_I2C2EN

```
#define APB1ENR_I2C2EN 22  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on I2C2 peripheral.

Definition at line 178 of file [MRCC_interface.h](#).

5.37.2.14 APB1ENR_I2C1EN

```
#define APB1ENR_I2C1EN 21  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on I2C1 peripheral.

Definition at line 185 of file [MRCC_interface.h](#).

5.37.2.15 APB1ENR_USART2EN

```
#define APB1ENR_USART2EN 17  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on USART2 peripheral.

Definition at line 192 of file [MRCC_interface.h](#).

5.37.2.16 APB1ENR_SPI3EN

```
#define APB1ENR_SPI3EN 15  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SPI3 peripheral.

Definition at line 199 of file [MRCC_interface.h](#).

5.37.2.17 APB1ENR_SPI2EN

```
#define APB1ENR_SPI2EN 14  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SPI2 peripheral.

Definition at line 206 of file [MRCC_interface.h](#).

5.37.2.18 APB1ENR_WWDGEN

```
#define APB1ENR_WWDGEN 11  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on WWD peripheral.

Definition at line 213 of file [MRCC_interface.h](#).

5.37.2.19 APB1ENR_TIM5EN

```
#define APB1ENR_TIM5EN 3

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM5 peripheral.

Definition at line [220](#) of file [MRCC_interface.h](#).

5.37.2.20 APB1ENR_TIM4EN

```
#define APB1ENR_TIM4EN 2

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM4 peripheral.

Definition at line [227](#) of file [MRCC_interface.h](#).

5.37.2.21 APB1ENR_TIM3EN

```
#define APB1ENR_TIM3EN 1

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM3 peripheral.

Definition at line [234](#) of file [MRCC_interface.h](#).

5.37.2.22 APB1ENR_TIM2EN

```
#define APB1ENR_TIM2EN 0

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM2 peripheral.

Definition at line [241](#) of file [MRCC_interface.h](#).

5.37.2.23 APB2ENR_TIM11EN

```
#define APB2ENR_TIM11EN 18

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM11 peripheral.

Definition at line [248](#) of file [MRCC_interface.h](#).

5.37.2.24 APB2ENR_TIM10EN

```
#define APB2ENR_TIM10EN 17

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM10 peripheral.

Definition at line [255](#) of file [MRCC_interface.h](#).

5.37.2.25 APB2ENR_TIM9EN

```
#define APB2ENR_TIM9EN 16

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM9 peripheral.

Definition at line [262](#) of file [MRCC_interface.h](#).

5.37.2.26 APB2ENR_SYSCFGGEN

```
#define APB2ENR_SYSCFGGEN 14

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SYSCFG peripheral.

Definition at line [269](#) of file [MRCC_interface.h](#).

5.37.2.27 APB2ENR_SPI4EN

```
#define APB2ENR_SPI4EN 13

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SPI4 peripheral.

Definition at line [276](#) of file [MRCC_interface.h](#).

5.37.2.28 APB2ENR_SPI1EN

```
#define APB2ENR_SPI1EN 12

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SPI1 peripheral.

Definition at line [283](#) of file [MRCC_interface.h](#).

5.37.2.29 APB2ENR_SDIOEN

```
#define APB2ENR_SDIOEN 11

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SDIO peripheral.

Definition at line [290](#) of file [MRCC_interface.h](#).

5.37.2.30 APB2ENR_ADC1EN

```
#define APB2ENR_ADC1EN 8

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on ADC1 peripheral.

Definition at line [297](#) of file [MRCC_interface.h](#).

5.37.2.31 APB2ENR_USART6EN

```
#define APB2ENR_USART6EN 5

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on USART6 peripheral.

Definition at line 304 of file [MRCC_interface.h](#).

5.37.2.32 APB2ENR_USART1EN

```
#define APB2ENR_USART1EN 4

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on USART1 peripheral.

Definition at line 311 of file [MRCC_interface.h](#).

5.37.2.33 APB2ENR_TIM1EN

```
#define APB2ENR_TIM1EN 0

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on TIM1 peripheral.

Definition at line 318 of file [MRCC_interface.h](#).

5.37.2.34 AHB1LPENR_FLITFLPEN

```
#define AHB1LPENR_FLITFLPEN 15

#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on FLITFLP peripheral.

Definition at line 325 of file [MRCC_interface.h](#).

5.38 SPI Peripherals

Modules

- [SPI Addresses](#)
- [SPI Registers](#)

5.38.1 Detailed Description

5.39 SPI Addresses

Macros

- #define SPI1_BASE_ADDRESS 0x40013000
- #define SPI2_BASE_ADDRESS 0x40003800
- #define SPI3_BASE_ADDRESS 0x40003C00
- #define SPI4_BASE_ADDRESS 0x40013400

5.39.1 Detailed Description

5.39.2 Macro Definition Documentation

5.39.2.1 SPI1_BASE_ADDRESS

```
#define SPI1_BASE_ADDRESS 0x40013000  
#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI1 Base Address

Definition at line 27 of file [SPI_interface.h](#).

5.39.2.2 SPI2_BASE_ADDRESS

```
#define SPI2_BASE_ADDRESS 0x40003800  
#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI2 Base Address

Definition at line 33 of file [SPI_interface.h](#).

5.39.2.3 SPI3_BASE_ADDRESS

```
#define SPI3_BASE_ADDRESS 0x40003C00  
#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI3 Base Address

Definition at line 39 of file [SPI_interface.h](#).

5.39.2.4 SPI4_BASE_ADDRESS

```
#define SPI4_BASE_ADDRESS 0x40013400

#include <COTS/MCAL/SPI/SPI_interface.h>
```

Definition at line 45 of file SPI_interface.h.

5.40 SPI Registers

Macros

- #define SPI1 ((SPI_MemoryMapType *)SPI1_BASE_ADDRESS)
 - #define SPI2 ((SPI_MemoryMapType *)SPI2_BASE_ADDRESS)
 - #define SPI3 ((SPI_MemoryMapType *)SPI3_BASE_ADDRESS)

5.40.1 Detailed Description

5.40.2 Macro Definition Documentation

5.40.2.1 SPI1

```
#define SPI1 ((SPI_MemoryMapType *)SPI1_BASE_ADDRESS)

#include <COTS/MCAL/SPI/SPI_interface.h>
```

Definition at line 58 of file [SPI_interface.h](#).

5.40.2.2 SPI2

```
#define SPI2 ((SPI_MemoryMapType *)SPI2_BASE_ADDRESS)

#include <COTS/MCAL/SPI/SPI_interface.h>

SPI2 Peripheral

Definition at line 64 of file SPI_interface.h
```

5.40.2.3 SPI3

```
#define SPI3 ((SPI_MemoryMapType *)SPI3_BASE_ADDRESS)

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI3 Peripheral

Definition at line 70 of file [SPI_interface.h](#).

5.41 SPI Interfacing Macros

Macros

- #define SPIx_MSTR 1
- #define SPIx_SLAVE 2
- #define SPIx_SIMPLEX 1
- #define SPIx_HALF_DUPLEX 2
- #define SPIx_FULL_DUPLEX 3

5.41.1 Detailed Description

5.41.2 Macro Definition Documentation

5.41.2.1 SPIx_MSTR

```
#define SPIx_MSTR 1

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI Master

Definition at line 217 of file [SPI_interface.h](#).

5.41.2.2 SPIx_SLAVE

```
#define SPIx_SLAVE 2

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI Slave

Definition at line 223 of file [SPI_interface.h](#).

5.41.2.3 SPIx_SIMPLEX

```
#define SPIx_SIMPLEX 1

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI Simplex

Definition at line 229 of file [SPI_interface.h](#).

5.41.2.4 SPIx_HALF_DUPLEX

```
#define SPIx_HALF_DUPLEX 2

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI Half Duplex

Definition at line 235 of file [SPI_interface.h](#).

5.41.2.5 SPIx_FULL_DUPLEX

```
#define SPIx_FULL_DUPLEX 3

#include <COTS/MCAL/SPI/SPI_interface.h>
```

SPI Full Duplex

Definition at line 241 of file [SPI_interface.h](#).

5.42 Systick Configuration

Macros

- #define CLK_SOURCE AHB_DividedBy8
- #define Exception_Request Dont AssertRequest
- #define SINGLE_INTERVAL_MODE (1)
- #define PERIODIC_INTERVAL_MODE (2)
- #define MAX_TICKS (16777216)

5.42.1 Detailed Description

5.42.2 Macro Definition Documentation

5.42.2.1 CLK_SOURCE

```
#define CLK_SOURCE AHB_DividedBy8

#include <COTS/MCAL/SysTick/SysTick_config.h>

Clock source for SysTick
```

Note

Accepted values: AHB_DividedBy8, AHB

Definition at line 24 of file [SysTick_config.h](#).

5.42.2.2 Exception_Request

```
#define Exception_Request Dont AssertRequest

#include <COTS/MCAL/SysTick/SysTick_config.h>

Exception request for SysTick
```

Note

Accepted values: Dont AssertRequest, AssertRequest

Definition at line 33 of file [SysTick_config.h](#).

5.42.2.3 SINGLE_INTERVAL_MODE

```
#define SINGLE_INTERVAL_MODE (1)

#include <COTS/MCAL/SysTick/SysTick_config.h>

Single interval mode
```

Definition at line 41 of file [SysTick_config.h](#).

5.42.2.4 PERIODIC_INTERVAL_MODE

```
#define PERIODIC_INTERVAL_MODE (2)

#include <COTS/MCAL/SysTick/SysTick_config.h>

Periodic interval mode
```

Definition at line 47 of file [SysTick_config.h](#).

5.42.2.5 MAX_TICKS

```
#define MAX_TICKS (16777216)

#include <COTS/MCAL/SysTick/SysTick_config.h>

Maximum number of ticks
```

Definition at line 53 of file [SysTick_config.h](#).

5.43 Delay Units

Macros

- #define **MILLI_SEC** (1)
Delay in milli seconds.
- #define **MICRO_SEC** (2)
Delay in micro seconds.
- #define **SEC** (3)
Delay in seconds.

5.43.1 Detailed Description

5.43.2 Macro Definition Documentation

5.43.2.1 MILLI_SEC

```
#define MILLI_SEC (1)

#include <COTS/MCAL/SysTick/SysTick_interface.h>
```

Delay in milli seconds.

Definition at line 142 of file [SysTick_interface.h](#).

5.43.2.2 MICRO_SEC

```
#define MICRO_SEC (2)

#include <COTS/MCAL/SysTick/SysTick_interface.h>
```

Delay in micro seconds.

Definition at line 148 of file [SysTick_interface.h](#).

5.43.2.3 SEC

```
#define SEC (3)

#include <COTS/MCAL/SysTick/SysTick_interface.h>
```

Delay in seconds.

Definition at line 154 of file [SysTick_interface.h](#).

5.44 Systick Addresses

Macros

- `#define SysTick_BASE_ADDRESS (0xE000E010)`

5.44.1 Detailed Description

5.44.2 Macro Definition Documentation

5.44.2.1 SysTick_BASE_ADDRESS

```
#define SysTick_BASE_ADDRESS (0xE000E010)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Systick Base address

Definition at line 55 of file [SysTick_private.h](#).

5.45 Systick Registers

Macros

- `#define SysTick ((volatile P2VAR(SysTick_Type))(SysTick_BASE_ADDRESS))`

5.45.1 Detailed Description

5.45.2 Macro Definition Documentation

5.45.2.1 SysTick

```
#define SysTick ((volatile P2VAR(SysTick_Type))(SysTick_BASE_ADDRESS))
```

```
#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Systick register

Definition at line 70 of file [SysTick_private.h](#).

5.46 Systick Register Bits Positions

Macros

- #define COUNTFLAG (16)
- #define CLKSOURCE (2)
- #define TICKINT (1)
- #define COUNTER_ENABLE (0)

5.46.1 Detailed Description

5.46.2 Macro Definition Documentation

5.46.2.1 COUNTFLAG

```
#define COUNTFLAG (16)
```

```
#include <COTS/MCAL/SysTick/SysTick_private.h>
```

This bit is responsible for keeping an eye on the timer whether it has counted to 0 or no, since the last time this was read It returns 1 if the timer counted to 0, since last time this was read

Definition at line 85 of file [SysTick_private.h](#).

5.46.2.2 CLKSOURCE

```
#define CLKSOURCE (2)
```

```
#include <COTS/MCAL/SysTick/SysTick_private.h>
```

This bit is responsible for selecting the clock source for the Systick peripheral

Definition at line 91 of file [SysTick_private.h](#).

5.46.2.3 TICKINT

```
#define TICKINT (1)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

This bit is responsible for controlling the exception request whether to enable or disable the request when reaching zero

Definition at line 97 of file [SysTick_private.h](#).

5.46.2.4 COUNTER_ENABLE

```
#define COUNTER_ENABLE (0)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

This bit is responsible for whether to enable the counter or no When this is enabled, the counter loads the reload value from the load register and then starts countdownning. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. Then, it loads the reload value again and begins counting.

See also

[SysTick_Type::VAL](#)
[TICKINT](#)

Definition at line 108 of file [SysTick_private.h](#).

5.47 Systick Clock Sources

Macros

- #define AHB_DividedBy8 (1)
- #define AHB (2)

5.47.1 Detailed Description

5.47.2 Macro Definition Documentation

5.47.2.1 AHB_DividedBy8

```
#define AHB_DividedBy8 (1)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Set the clock source to divide the clock output of RCC by 8

Definition at line 120 of file [SysTick_private.h](#).

5.47.2.2 AHB

```
#define AHB (2)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Set the clock source to the same as the clock output of RCC

Definition at line 126 of file [SysTick_private.h](#).

5.48 Systick Exception (Interrupt) Status

Macros

- #define Dont AssertRequest (1)
- #define AssertRequest (2)

5.48.1 Detailed Description

5.48.2 Macro Definition Documentation

5.48.2.1 Dont AssertRequest

```
#define Dont AssertRequest (1)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Don't raise exception (interrupt) when the counter reaches 0

Definition at line 138 of file [SysTick_private.h](#).

5.48.2.2 AssertRequest

```
#define AssertRequest (2)

#include <COTS/MCAL/SysTick/SysTick_private.h>
```

Raise exception (interrupt) when the counter reaches 0

Definition at line 144 of file [SysTick_private.h](#).

5.49 UART Addresses

Macros

- #define [USART1_BASE_ADDRESS](#) (0x40011000)
- #define [USART2_BASE_ADDRESS](#) (0x40004400)
- #define [USART6_BASE_ADDRESS](#) (0x40011400)

5.49.1 Detailed Description

5.49.2 Macro Definition Documentation

5.49.2.1 USART1_BASE_ADDRESS

```
#define USART1_BASE_ADDRESS (0x40011000)

#include <COTS/MCAL/UART/UART_interface.h>

USART1 base address in the memory

Definition at line 26 of file UART\_interface.h.
```

5.49.2.2 USART2_BASE_ADDRESS

```
#define USART2_BASE_ADDRESS (0x40004400)

#include <COTS/MCAL/UART/UART_interface.h>

USART2 base address in the memory

Definition at line 32 of file UART\_interface.h.
```

5.49.2.3 USART6_BASE_ADDRESS

```
#define USART6_BASE_ADDRESS (0x40011400)

#include <COTS/MCAL/UART/UART_interface.h>
```

USART6 base address in the memory

Definition at line 38 of file [UART_interface.h](#).

5.50 UART Registers

Macros

- #define USART1_REG ((USART_MemoryMapType *)USART1_BASE_ADDRESS)
- #define USART2_REG ((USART_MemoryMapType *)USART2_BASE_ADDRESS)
- #define USART6_REG ((USART_MemoryMapType *)USART6_BASE_ADDRESS)

5.50.1 Detailed Description

5.50.2 Macro Definition Documentation

5.50.2.1 USART1_REG

```
#define USART1_REG ((USART_MemoryMapType *)USART1_BASE_ADDRESS)

#include <COTS/MCAL/UART/UART_interface.h>

USART1 registers map
```

Definition at line 51 of file [UART_interface.h](#).

5.50.2.2 USART2_REG

```
#define USART2_REG ((USART_MemoryMapType *)USART2_BASE_ADDRESS)

#include <COTS/MCAL/UART/UART_interface.h>

USART2 registers map
```

Definition at line 57 of file [UART_interface.h](#).

5.50.2.3 USART6_REG

```
#define USART6_REG ((USART_MemoryMapType *)USART6_BASE_ADDRESS)
```

```
#include <COTS/MCAL/UART/UART_interface.h>
```

USART6 registers map

Definition at line 63 of file [UART_interface.h](#).

5.51 UART Oversampling

Macros

- #define OVER_SAMPLING_16 (0)
- #define OVER_SAMPLING_8 (1)

5.51.1 Detailed Description

5.51.2 Macro Definition Documentation

5.51.2.1 OVER_SAMPLING_16

```
#define OVER_SAMPLING_16 (0)
```

```
#include <COTS/MCAL/UART/UART_interface.h>
```

Oversampling by 16

Definition at line 296 of file [UART_interface.h](#).

5.51.2.2 OVER_SAMPLING_8

```
#define OVER_SAMPLING_8 (1)
```

```
#include <COTS/MCAL/UART/UART_interface.h>
```

Oversampling by 8

Definition at line 302 of file [UART_interface.h](#).

5.52 UART Operating Modes

Macros

- #define TX_ONLY (0)
- #define RX_ONLY (1)
- #define TX_RX (2)

5.52.1 Detailed Description

5.52.2 Macro Definition Documentation

5.52.2.1 TX_ONLY

```
#define TX_ONLY (0)

#include <COTS/MCAL/UART/UART_interface.h>
```

Transmit only mode

Definition at line [314](#) of file [UART_interface.h](#).

5.52.2.2 RX_ONLY

```
#define RX_ONLY (1)

#include <COTS/MCAL/UART/UART_interface.h>
```

Receive only mode

Definition at line [320](#) of file [UART_interface.h](#).

5.52.2.3 TX_RX

```
#define TX_RX (2)

#include <COTS/MCAL/UART/UART_interface.h>
```

Transmit and Receive mode

Definition at line [326](#) of file [UART_interface.h](#).

5.53 UART Parity

Macros

- #define EVEN_PARITY (0)
- #define ODD_PARITY (1)

5.53.1 Detailed Description

5.53.2 Macro Definition Documentation

5.53.2.1 EVEN_PARITY

```
#define EVEN_PARITY (0)

#include <COTS/MCAL/UART/UART_interface.h>

Even parity

Definition at line 338 of file UART\_interface.h.
```

5.53.2.2 ODD_PARITY

```
#define ODD_PARITY (1)

#include <COTS/MCAL/UART/UART_interface.h>

Odd parity

Definition at line 344 of file UART\_interface.h.
```

5.54 UART Bit Sizes

Macros

- #define MODE_8BIT (0)
- #define MODE_9BIT (1)

5.54.1 Detailed Description

5.54.2 Macro Definition Documentation

5.54.2.1 MODE_8BIT

```
#define MODE_8BIT (0)

#include <COTS/MCAL/UART/UART_interface.h>
```

8-bit mode for UART

Definition at line 356 of file [UART_interface.h](#).

5.54.2.2 MODE_9BIT

```
#define MODE_9BIT (1)

#include <COTS/MCAL/UART/UART_interface.h>
```

9-bit mode for UART

Definition at line 362 of file [UART_interface.h](#).

5.55 UART Stop Bits

Macros

- #define STOP_BIT_1 (0)
- #define STOP_BIT_0_5 (1)
- #define STOP_BIT_2 (2)
- #define STOP_BIT_1_5 (3)

5.55.1 Detailed Description

5.55.2 Macro Definition Documentation

5.55.2.1 STOP_BIT_1

```
#define STOP_BIT_1 (0)

#include <COTS/MCAL/UART/UART_interface.h>
```

1 stop bit

Definition at line 374 of file [UART_interface.h](#).

5.55.2.2 STOP_BIT_0_5

```
#define STOP_BIT_0_5 (1)

#include <COTS/MCAL/UART/UART_interface.h>

0.5 stop bits
```

Definition at line 380 of file [UART_interface.h](#).

5.55.2.3 STOP_BIT_2

```
#define STOP_BIT_2 (2)

#include <COTS/MCAL/UART/UART_interface.h>

2 stop bits
```

Definition at line 386 of file [UART_interface.h](#).

5.55.2.4 STOP_BIT_1_5

```
#define STOP_BIT_1_5 (3)

#include <COTS/MCAL/UART/UART_interface.h>

1.5 stop bits
```

Definition at line 392 of file [UART_interface.h](#).

Chapter 6

Data Structure Documentation

6.1 ADC_MemoryMapType Struct Reference

ADC declaration structure for its registers.

```
#include "COTS/MCAL/ADC/ADC_private.h"
```

Data Fields

- **u32_t SR**
clock control register
- **u32_t CR1**
PLL configuration register.
- **u32_t CR2**
clock configuration register
- **u32_t SMPR1**
clock interrupt register
- **u32_t SMPR2**
AHB1 peripheral reset register.
- **u32_t JOFR1**
AHB2 peripheral reset register.
- **u32_t JOFR2**
Reserved register.
- **u32_t JOFR3**
Reserved register.
- **u32_t JOFR4**
APB1 peripheral reset register.
- **u32_t HTR**
APB2 peripheral reset register.
- **u32_t LTR**
Reserved register.
- **u32_t SQR1**
Reserved register.
- **u32_t SQR2**
AHB1 peripheral clock enable register.

- [`u32_t SQR3`](#)
AHB2 peripheral clock enable register.
- [`u32_t JSQR`](#)
Reserved register.
- [`u32_t JDR1`](#)
Reserved register.
- [`u32_t JDR2`](#)
APB1 peripheral clock enable register.
- [`u32_t JDR3`](#)
APB2 peripheral clock enable register.
- [`u32_t JDR4`](#)
Reserved register.
- [`u32_t DR`](#)
Reserved register.
- [`u32_t CCR`](#)
AHB1 peripheral clock enable in low power mode register.

6.1.1 Detailed Description

ADC declaration structure for its registers.

Definition at line 20 of file [ADC_private.h](#).

6.1.2 Field Documentation

6.1.2.1 SR

[`u32_t SR`](#)

clock control register

Definition at line 26 of file [ADC_private.h](#).

6.1.2.2 CR1

[`u32_t CR1`](#)

PLL configuration register.

Definition at line 31 of file [ADC_private.h](#).

6.1.2.3 CR2

`u32_t` CR2

clock configuration register

Definition at line 36 of file [ADC_private.h](#).

6.1.2.4 SMPR1

`u32_t` SMPR1

clock interrupt register

Definition at line 41 of file [ADC_private.h](#).

6.1.2.5 SMPR2

`u32_t` SMPR2

AHB1 peripheral reset register.

Definition at line 46 of file [ADC_private.h](#).

6.1.2.6 JOFR1

`u32_t` JOFR1

AHB2 peripheral reset register.

Definition at line 51 of file [ADC_private.h](#).

6.1.2.7 JOFR2

`u32_t` JOFR2

Reserved register.

Definition at line 56 of file [ADC_private.h](#).

6.1.2.8 JOFR3

`u32_t` JOFR3

Reserved register.

Definition at line 61 of file [ADC_private.h](#).

6.1.2.9 JOFR4

`u32_t` JOFR4

APB1 peripheral reset register.

Definition at line 66 of file [ADC_private.h](#).

6.1.2.10 HTR

`u32_t` HTR

APB2 peripheral reset register.

Definition at line 71 of file [ADC_private.h](#).

6.1.2.11 LTR

`u32_t` LTR

Reserved register.

Definition at line 76 of file [ADC_private.h](#).

6.1.2.12 SQR1

`u32_t` SQR1

Reserved register.

Definition at line 81 of file [ADC_private.h](#).

6.1.2.13 SQR2

`u32_t SQR2`

AHB1 peripheral clock enable register.

Definition at line [86](#) of file [ADC_private.h](#).

6.1.2.14 SQR3

`u32_t SQR3`

AHB2 peripheral clock enable register.

Definition at line [91](#) of file [ADC_private.h](#).

6.1.2.15 JSQR

`u32_t JSQR`

Reserved register.

Definition at line [96](#) of file [ADC_private.h](#).

6.1.2.16 JDR1

`u32_t JDR1`

Reserved register.

Definition at line [101](#) of file [ADC_private.h](#).

6.1.2.17 JDR2

`u32_t JDR2`

APB1 peripheral clock enable register.

Definition at line [106](#) of file [ADC_private.h](#).

6.1.2.18 JDR3

`u32_t` JDR3

APB2 peripheral clock enable register.

Definition at line 111 of file [ADC_private.h](#).

6.1.2.19 JDR4

`u32_t` JDR4

Reserved register.

Definition at line 116 of file [ADC_private.h](#).

6.1.2.20 DR

`u32_t` DR

Reserved register.

Definition at line 121 of file [ADC_private.h](#).

6.1.2.21 CCR

`u32_t` CCR

AHB1 peripheral clock enable in low power mode register.

Definition at line 126 of file [ADC_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/ADC/ADC_private.h

6.2 BUZZER_BuzzerConfiguration Struct Reference

Buzzer configuration structure for Buzzer initialization.

```
#include "COTS/HAL/BUZZER/BUZZER_interface.h"
```

Data Fields

- `u8_t u8Port`
Initialize the Buzzer on a certain port.
- `u8_t u8Pin`
Initialize the Buzzer on a certain pin.

6.2.1 Detailed Description

Buzzer configuration structure for Buzzer initialization.

This structure is used to initialize the Buzzer with a certain port and pin

Definition at line 25 of file [BUZZER_interface.h](#).

6.2.2 Field Documentation

6.2.2.1 u8Port

`u8_t u8Port`

Initialize the Buzzer on a certain port.

Definition at line 30 of file [BUZZER_interface.h](#).

Referenced by [HBUZZER_vInit\(\)](#), [HBUZZER_vSoundOff\(\)](#), [HBUZZER_vSoundOn\(\)](#), and [HBUZZER_vToggleSound\(\)](#).

6.2.2.2 u8Pin

`u8_t u8Pin`

Initialize the Buzzer on a certain pin.

Definition at line 34 of file [BUZZER_interface.h](#).

Referenced by [HBUZZER_vInit\(\)](#), [HBUZZER_vSoundOff\(\)](#), [HBUZZER_vSoundOn\(\)](#), and [HBUZZER_vToggleSound\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/HAL/BUZZER/[BUZZER_interface.h](#)

6.3 DCM_MotorConfiguration Struct Reference

DC Motor configuration structure for motor initialization.

```
#include "COTS/HAL/DCMOTOR/DCM_interface.h"
```

Data Fields

- `u8_t u8Port`
Initialize the DC Motor on a certain port.
- `u8_t u8Pin1`
Initialize the DC Motor on a certain pin.
- `u8_t u8Pin2`
Initialize the DC Motor on a certain pin.
- `u8_t u8Direction`
Initialize the DC Motor with a certain direction.
- `u8_t u8SpeedPin`
Initialize the DC Motor with a certain speed pin.
- `u32_t u32SpeedRatio`
Initialize the DC Motor with a certain speed ratio.

6.3.1 Detailed Description

DC Motor configuration structure for motor initialization.

DC Motor configuration for motor initialization for a certain port and pin.

Since DC Motors have 2 sides to be controller, we will initialize each motor with 2 pins.

Taking into consideration that the pins could be on different ports.

Definition at line 28 of file [DCM_interface.h](#).

6.3.2 Field Documentation

6.3.2.1 u8Port

```
u8_t u8Port
```

Initialize the DC Motor on a certain port.

Definition at line 33 of file [DCM_interface.h](#).

Referenced by [HDCM_vInitMotor\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), and [HDCM_vStopMotor\(\)](#).

6.3.2.2 u8Pin1

`u8_t u8Pin1`

Initialize the DC Motor on a certain pin.

Definition at line 37 of file [DCM_interface.h](#).

Referenced by [HDCM_vInitMotor\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), and [HDCM_vStopMotor\(\)](#).

6.3.2.3 u8Pin2

`u8_t u8Pin2`

Initialize the DC Motor on a certain pin.

Definition at line 41 of file [DCM_interface.h](#).

Referenced by [HDCM_vInitMotor\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), and [HDCM_vStopMotor\(\)](#).

6.3.2.4 u8Direction

`u8_t u8Direction`

Initialize the DC Motor with a certain direction.

Definition at line 45 of file [DCM_interface.h](#).

Referenced by [HDCM_vInitMotor\(\)](#).

6.3.2.5 u8SpeedPin

`u8_t u8SpeedPin`

Initialize the DC Motor with a certain speed pin.

Definition at line 49 of file [DCM_interface.h](#).

Referenced by [HDCM_vGetSpeedValue\(\)](#), and [HDCM_vMotorSpeedCntrl\(\)](#).

6.3.2.6 u32SpeedRatio

`u32_t u32SpeedRatio`

Initialize the DC Motor with a certain speed ratio.

Definition at line 53 of file [DCM_interface.h](#).

Referenced by [HDCM_vInitMotor\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/HAL/DCMOTOR/[DCM_interface.h](#)

6.4 EXTI_ConfigType Struct Reference

Interrupt configuration structure to initialize the interrupt with.

```
#include "COTS/MCAL/EXTI/EXTI_interface.h"
```

Data Fields

- `u8_t LineNum`
Initialize the interrupt with a certain line ID.
- `u8_t PortNum`
Initialize the interrupt with a certain PORT.
- `u8_t TriggerStatus`
The trigger status to trigger when the interrupt occurs.

6.4.1 Detailed Description

Interrupt configuration structure to initialize the interrupt with.

Definition at line 18 of file [EXTI_interface.h](#).

6.4.2 Field Documentation

6.4.2.1 LineNum

`u8_t LineNum`

Initialize the interrupt with a certain line ID.

Definition at line 23 of file [EXTI_interface.h](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

6.4.2.2 PortNum

`u8_t PortNum`

Initialize the interrupt with a certain PORT.

Definition at line 28 of file [EXTI_interface.h](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

6.4.2.3 TriggerStatus

`u8_t TriggerStatus`

The trigger status to trigger when the interrupt occurs.

Definition at line 33 of file [EXTI_interface.h](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/EXTI/[EXTI_interface.h](#)

6.5 EXTI_Type Struct Reference

EXTI Configuration structure for interrupt initialization process.

```
#include "COTS/MCAL/EXTI/EXTI_private.h"
```

Data Fields

- `u32_t IMR`
Interrupt mask register.
- `u32_t EMR`
Event mask register.
- `u32_t RTSR`
Rising trigger status register.
- `u32_t FTSR`
Falling trigger status register.
- `u32_t SWIER`
Software interrupt event register.
- `u32_t PR`
Pending register.

6.5.1 Detailed Description

EXTI Configuration structure for interrupt initialization process.

Definition at line 18 of file [EXTI_private.h](#).

6.5.2 Field Documentation

6.5.2.1 IMR

`u32_t IMR`

Interrupt mask register.

Definition at line 23 of file [EXTI_private.h](#).

6.5.2.2 EMR

`u32_t EMR`

Event mask register.

Definition at line 27 of file [EXTI_private.h](#).

6.5.2.3 RTSR

`u32_t RTSR`

Rising trigger status register.

Definition at line 31 of file [EXTI_private.h](#).

6.5.2.4 FTSR

`u32_t FTSR`

Falling trigger status register.

Definition at line 35 of file [EXTI_private.h](#).

6.5.2.5 SWIER

`u32_t SWIER`

Software interrupt event register.

Definition at line 39 of file [EXTI_private.h](#).

6.5.2.6 PR

`u32_t PR`

Pending register.

Definition at line 43 of file [EXTI_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/EXTI/[EXTI_private.h](#)

6.6 GPIOx_MemoryMapType Struct Reference

GPIO declaration structure for its registers.

```
#include "COTS/MCAL/GPIO/GPIO_private.h"
```

Data Fields

- `u32_t MODERx`
*Control PIN mode whether **INPUT** or **OUTPUT***
- `u32_t OTYPERx`
Select the output type between push-pull or open-drain.
- `u32_t OSPEEDRx`
Configure the speed that the PIN is connected to.
- `u32_t PUPDRx`
Control the pull-up or pull-down of the pin, despite its direction.
- `u32_t IDRx`
Read the input data.
- `u32_t ODRx`
Write the output data.
- `u32_t BSRRx`
(Re)set each bit in the output data register
- `u32_t LCKRx`
Lock the GPIO control registers.
- `u32_t AFRLx`
*Control alternate function **LOW** register.*
- `u32_t AFRHx`
*Control alternate function **HIGH** register.*

6.6.1 Detailed Description

GPIO declaration structure for its registers.

Definition at line 18 of file [GPIO_private.h](#).

6.6.2 Field Documentation

6.6.2.1 MODERx

`u32_t MODERx`

Control PIN mode whether **INPUT** or **OUTPUT**

Definition at line 23 of file [GPIO_private.h](#).

6.6.2.2 OTYPERx

`u32_t OTYPERx`

Select the output type between push-pull or open-drain.

Definition at line 28 of file [GPIO_private.h](#).

6.6.2.3 OSPEEDRx

`u32_t OSPEEDRx`

Configure the speed that the PIN is connected to.

Definition at line 33 of file [GPIO_private.h](#).

6.6.2.4 PUPDRx

`u32_t PUPDRx`

Control the pull-up or pull-down of the pin, despite its direction.

Definition at line 38 of file [GPIO_private.h](#).

6.6.2.5 IDR_x

`u32_t IDRx`

Read the input data.

Definition at line 43 of file [GPIO_private.h](#).

6.6.2.6 ODR_x

`u32_t ODRx`

Write the output data.

Definition at line 48 of file [GPIO_private.h](#).

6.6.2.7 BSRR_x

`u32_t BSRRx`

(Re)set each bit in the output data register

Definition at line 53 of file [GPIO_private.h](#).

6.6.2.8 LCKR_x

`u32_t LCKRx`

Lock the GPIO control registers.

Definition at line 58 of file [GPIO_private.h](#).

6.6.2.9 AFRL_x

`u32_t AFRLx`

Control alternate function **LOW** register.

Definition at line 63 of file [GPIO_private.h](#).

6.6.2.10 AFRHx

`u32_t AFRHx`

Control alternate function **HIGH** register.

Definition at line 68 of file [GPIO_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/GPIO/[GPIO_private.h](#)

6.7 HULTSNC_ConfigType Struct Reference

Ultrasonic Configurations.

```
#include "COTS/HAL/UltraSonic/UltraSonic_interface.h"
```

Data Fields

- `u8_t u8Port`
Trigger Port.
- `u8_t u8Pin`
Trigger Pin.

6.7.1 Detailed Description

Ultrasonic Configurations.

Definition at line 20 of file [UltraSonic_interface.h](#).

6.7.2 Field Documentation

6.7.2.1 u8Port

`u8_t u8Port`

Trigger Port.

Definition at line 25 of file [UltraSonic_interface.h](#).

Referenced by [HULTSNC_vInit\(\)](#), and [HULTSNC_vTrigger\(\)](#).

6.7.2.2 u8Pin

`u8_t u8Pin`

Trigger Pin.

Definition at line 29 of file [UltraSonic_interface.h](#).

Referenced by [HULTSNC_vInit\(\)](#), and [HULTSNC_vTrigger\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/HAL/UltraSonic/[UltraSonic_interface.h](#)

6.8 LED_LEDConfiguration Struct Reference

LED configuration structure for LED initialization.

```
#include "COTS/HAL/LED/LED_interface.h"
```

Data Fields

- `u8_t u8Port`
Initialize the LED on a certain port.
- `u8_t u8Pin`
Initialize the LED on a certain pin.

6.8.1 Detailed Description

LED configuration structure for LED initialization.

This structure is used to initialize the LED with a certain port and pin

Definition at line 17 of file [LED_interface.h](#).

6.8.2 Field Documentation

6.8.2.1 u8Port

`u8_t u8Port`

Initialize the LED on a certain port.

Definition at line 22 of file [LED_interface.h](#).

6.8.2.2 u8Pin

`u8_t u8Pin`

Initialize the LED on a certain pin.

Definition at line 26 of file [LED_interface.h](#).

The documentation for this struct was generated from the following file:

- COTS/HAL/LED/LED_interface.h

6.9 MGPIox_ConfigType Struct Reference

MDIO Configuration structure for a specific PIN initialization.

```
#include "COTS/MCAL/GPIO/GPIO_interface.h"
```

Data Fields

- `u8_t Port`
Configures a specific GPIO port.
- `u8_t Pin`
Configures a specific GPIO pin.
- `u8_t Mode`
Configures a specific GPIO mode.
- `u8_t OutputType`
Configures a specific GPIO output's type.
- `u8_t OutputSpeed`
Configures a specific GPIO output's speed.
- `u8_t InputType`
Configures a specific GPIO input type.
- `u8_t AF_Type`
Configures a specific GPIO Alternative function.

6.9.1 Detailed Description

MDIO Configuration structure for a specific PIN initialization.

Definition at line 23 of file [GPIO_interface.h](#).

6.9.2 Field Documentation

6.9.2.1 Port

`u8_t Port`

Configures a specific GPIO port.

Definition at line 28 of file [GPIO_interface.h](#).

Referenced by [MGPIox_vInit\(\)](#), and [MUSART_vInit\(\)](#).

6.9.2.2 Pin

`u8_t Pin`

Configures a specific GPIO pin.

Definition at line 32 of file [GPIO_interface.h](#).

Referenced by [MGPIox_vInit\(\)](#).

6.9.2.3 Mode

`u8_t Mode`

Configures a specific GPIO mode.

Definition at line 36 of file [GPIO_interface.h](#).

Referenced by [MGPIox_vInit\(\)](#).

6.9.2.4 OutputType

`u8_t OutputType`

Configures a specific GPIO output's type.

Definition at line 40 of file [GPIO_interface.h](#).

Referenced by [MGPIox_vInit\(\)](#).

6.9.2.5 OutputSpeed

`u8_t` `OutputSpeed`

Configures a specific GPIO output's speed.

Definition at line 44 of file [GPIO_interface.h](#).

Referenced by [MGPIOx_vInit\(\)](#).

6.9.2.6 InputType

`u8_t` `InputType`

Configures a specific GPIO input type.

Definition at line 48 of file [GPIO_interface.h](#).

Referenced by [MGPIOx_vInit\(\)](#).

6.9.2.7 AF_Type

`u8_t` `AF_Type`

Configures a specific GPIO Alternative function.

Definition at line 52 of file [GPIO_interface.h](#).

Referenced by [MGPIOx_vInit\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/GPIO/[GPIO_interface.h](#)

6.10 MSYSCFG_MemMap_t Struct Reference

System configuration structure to initialize the SYSCFG module with.

```
#include "COTS/MCAL/EXTI/SYSCFG_private.h"
```

Data Fields

- `u32_t MEMRMP`
Memory remap register.
- `u32_t PMC`
Peripheral mode configuration register.
- `u32_t EXTICR [4]`
External interrupt configuration 1-4 registers.
- `u32_t CMPCR`
Compensation cell control register.

6.10.1 Detailed Description

System configuration structure to initialize the SYSCFG module with.

Definition at line 19 of file [SYSCFG_private.h](#).

6.10.2 Field Documentation

6.10.2.1 MEMRMP

`u32_t MEMRMP`

Memory remap register.

This register is used for specific configurations on memory map

Definition at line 25 of file [SYSCFG_private.h](#).

6.10.2.2 PMC

`u32_t PMC`

Peripheral mode configuration register.

Definition at line 29 of file [SYSCFG_private.h](#).

6.10.2.3 EXTICR

`u32_t EXTICR [4]`

External interrupt configuration 1-4 registers.

Definition at line 33 of file [SYSCFG_private.h](#).

6.10.2.4 CMPCR

`u32_t CMPCR`

Compensation cell control register.

Definition at line 37 of file [SYSCFG_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/EXTI/[SYSCFG_private.h](#)

6.11 NVIC_MemoryMapType Struct Reference

NVIC configuration structure for NVIC memory map.

```
#include "COTS/MCAL/NVIC/NVIC_private.h"
```

Data Fields

- `u32_t ISERx [8]`
Interrupt set-enable register.
- `u32_t RESERVED0x [24]`
- `u32_t ICERx [8]`
Interrupt clear-enable register.
- `u32_t RSERVED1x [24]`
- `u32_t ISPRx [8]`
Interrupt set-pending register.
- `u32_t RESERVED2x [24]`
- `u32_t ICPRx [8]`
Interrupt clear-pending register.
- `u32_t RESERVED3x [24]`
- `u32_t IABRx [8]`
Interrupt active bit register.
- `u32_t RESERVED4x [56]`
- `u8_t IPRx [240]`
Interrupt priority register.

6.11.1 Detailed Description

NVIC configuration structure for NVIC memory map.

Definition at line 18 of file [NVIC_private.h](#).

6.11.2 Field Documentation

6.11.2.1 ISERx

`u32_t ISERx[8]`

Interrupt set-enable register.

This register is responsible for enabling the interrupt

Note

Writing 0 has no effect at all

Writing 1 enables the interrupt

See also

[ICERx](#)

Definition at line [27](#) of file [NVIC_private.h](#).

6.11.2.2 RESERVED0x

`u32_t RESERVED0x[24]`

Note

This is a reserved register

Definition at line [31](#) of file [NVIC_private.h](#).

6.11.2.3 ICERx

`u32_t ICERx[8]`

Interrupt clear-enable register.

This register is responsible for disabling the interrupt

Note

Writing 0 has no effect at all

Writing 1 disables the interrupt

See also

[ISERx](#)

Definition at line [39](#) of file [NVIC_private.h](#).

6.11.2.4 RSERVED1x

`u32_t RSERVED1x[24]`

Note

This is a reserved register

Definition at line 43 of file [NVIC_private.h](#).

6.11.2.5 ISPRx

`u32_t ISPRx[8]`

Interrupt set-pending register.

Mostly used in testing the NVIC module

Note

Writing 0 has no effect at all

Writing 1 changes the interrupt state to pending

Writing 1 on a certain bit that already has a value of 1, it stays pending and the value doesn't change

See also

[ICPRx](#)

Definition at line 53 of file [NVIC_private.h](#).

6.11.2.6 RESERVED2x

`u32_t RESERVED2x[24]`

Note

This is a reserved register

Definition at line 57 of file [NVIC_private.h](#).

6.11.2.7 ICPRx

`u32_t ICPRx [8]`

Interrupt clear-pending register.

Mostly used in testing the NVIC module

Note

Writing 0 has no effect at all

Writing 1 removes the pending interrupt state

Writing 1 on a certain bit that already has a value of 1, it stays pending and the value doesn't change

See also

[ISPRx](#)

Definition at line [67](#) of file [NVIC_private.h](#).

6.11.2.8 RESERVED3x

`u32_t RESERVED3x [24]`

Note

This is a reserved register

Definition at line [71](#) of file [NVIC_private.h](#).

6.11.2.9 IABRx

`u32_t IABRx [8]`

Interrupt active bit register.

Note

This is a read-only register

Definition at line [77](#) of file [NVIC_private.h](#).

6.11.2.10 RESERVED4x

`u32_t` RESERVED4x[56]

Note

This is a reserved register

Definition at line 81 of file [NVIC_private.h](#).

6.11.2.11 IPRx

`u8_t` IPRx[240]

Interrupt priority register.

See also

[Group priorities](#)

[Sub-Group priorities](#)

Note

Interrupt priority register

Definition at line 89 of file [NVIC_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/NVIC/[NVIC_private.h](#)

6.12 RCC_MemoryMapType Struct Reference

RCC declaration structure for its registers.

```
#include "COTS/MCAL/RCC/MRCC_private.h"
```

Data Fields

- **u32_t CR**
clock control register
- **u32_t PLLCFGR**
PLL configuration register.
- **u32_t CFGR**
clock configuration register
- **u32_t CIR**
clock interrupt register
- **u32_t AHB1RSTR**
AHB1 peripheral reset register.
- **u32_t AHB2RSTR**
AHB2 peripheral reset register.
- **u32_t Reserved1**
Reserved register.
- **u32_t Reserved2**
Reserved register.
- **u32_t APB1RSTR**
APB1 peripheral reset register.
- **u32_t APB2RSTR**
APB2 peripheral reset register.
- **u32_t Reserved3**
Reserved register.
- **u32_t Reserved4**
Reserved register.
- **u32_t AHB1ENR**
AHB1 peripheral clock enable register.
- **u32_t AHB2ENR**
AHB2 peripheral clock enable register.
- **u32_t Reserved5**
Reserved register.
- **u32_t Reserved6**
Reserved register.
- **u32_t APB1ENR**
APB1 peripheral clock enable register.
- **u32_t APB2ENR**
APB2 peripheral clock enable register.
- **u32_t Reserved7**
Reserved register.
- **u32_t Reserved8**
Reserved register.
- **u32_t AHB1LPENR**
AHB1 peripheral clock enable in low power mode register.
- **u32_t AHB2LPENR**
AHB2 peripheral clock enable in low power mode register.
- **u32_t Reserved9**
Reserved register.
- **u32_t Reserved10**
Reserved register.
- **u32_t APB1LPENR**

- **u32_t APB2LPENR**
APB2 peripheral clock enable in low power mode register.
- **u32_t Reserved11**
Reserved register.
- **u32_t Reserved12**
Reserved register.
- **u32_t BDCR**
RCC Backup domain control register.
- **u32_t CSR**
clock control & status register
- **u32_t Reserved13**
Reserved register.
- **u32_t Reserved14**
Reserved register.
- **u32_t SSCGR**
RCC spread spectrum clock generation register.
- **u32_t PLLI2SCFGR**
PLLI2S configuration register.
- **u32_t DCKCFGR**
Dedicated Clocks Configuration Register.

6.12.1 Detailed Description

RCC declaration structure for its registers.

Definition at line 24 of file [MRCC_private.h](#).

6.12.2 Field Documentation

6.12.2.1 CR

u32_t CR

clock control register

Definition at line 30 of file [MRCC_private.h](#).

6.12.2.2 PLLCFG

u32_t PLLCFG

PLL configuration register.

Definition at line 35 of file [MRCC_private.h](#).

6.12.2.3 CFGR

`u32_t` CFGR

clock configuration register

Definition at line 40 of file [MRCC_private.h](#).

6.12.2.4 CIR

`u32_t` CIR

clock interrupt register

Definition at line 45 of file [MRCC_private.h](#).

6.12.2.5 AHB1RSTR

`u32_t` AHB1RSTR

AHB1 peripheral reset register.

Definition at line 50 of file [MRCC_private.h](#).

6.12.2.6 AHB2RSTR

`u32_t` AHB2RSTR

AHB2 peripheral reset register.

Definition at line 55 of file [MRCC_private.h](#).

6.12.2.7 Reserved1

`u32_t` Reserved1

Reserved register.

Definition at line 60 of file [MRCC_private.h](#).

6.12.2.8 Reserved2

`u32_t` Reserved2

Reserved register.

Definition at line 65 of file [MRCC_private.h](#).

6.12.2.9 APB1RSTR

`u32_t` APB1RSTR

APB1 peripheral reset register.

Definition at line 70 of file [MRCC_private.h](#).

6.12.2.10 APB2RSTR

`u32_t` APB2RSTR

APB2 peripheral reset register.

Definition at line 75 of file [MRCC_private.h](#).

6.12.2.11 Reserved3

`u32_t` Reserved3

Reserved register.

Definition at line 80 of file [MRCC_private.h](#).

6.12.2.12 Reserved4

`u32_t` Reserved4

Reserved register.

Definition at line 85 of file [MRCC_private.h](#).

6.12.2.13 AHB1ENR

`u32_t` AHB1ENR

AHB1 peripheral clock enable register.

Definition at line 90 of file [MRCC_private.h](#).

6.12.2.14 AHB2ENR

`u32_t` AHB2ENR

AHB2 peripheral clock enable register.

Definition at line 95 of file [MRCC_private.h](#).

6.12.2.15 Reserved5

`u32_t` Reserved5

Reserved register.

Definition at line 100 of file [MRCC_private.h](#).

6.12.2.16 Reserved6

`u32_t` Reserved6

Reserved register.

Definition at line 105 of file [MRCC_private.h](#).

6.12.2.17 APB1ENR

`u32_t` APB1ENR

APB1 peripheral clock enable register.

Definition at line 110 of file [MRCC_private.h](#).

6.12.2.18 APB2ENR

`u32_t` APB2ENR

APB2 peripheral clock enable register.

Definition at line 115 of file [MRCC_private.h](#).

6.12.2.19 Reserved7

`u32_t` Reserved7

Reserved register.

Definition at line 120 of file [MRCC_private.h](#).

6.12.2.20 Reserved8

`u32_t` Reserved8

Reserved register.

Definition at line 125 of file [MRCC_private.h](#).

6.12.2.21 AHB1LPENR

`u32_t` AHB1LPENR

AHB1 peripheral clock enable in low power mode register.

Definition at line 130 of file [MRCC_private.h](#).

6.12.2.22 AHB2LPENR

`u32_t` AHB2LPENR

AHB2 peripheral clock enable in low power mode register.

Definition at line 135 of file [MRCC_private.h](#).

6.12.2.23 Reserved9

`u32_t` Reserved9

Reserved register.

Definition at line 140 of file [MRCC_private.h](#).

6.12.2.24 Reserved10

`u32_t` Reserved10

Reserved register.

Definition at line 145 of file [MRCC_private.h](#).

6.12.2.25 APB1LPENR

`u32_t` APB1LPENR

APB1 peripheral clock enable in low power mode register.

Definition at line 150 of file [MRCC_private.h](#).

6.12.2.26 APB2LPENR

`u32_t` APB2LPENR

APB2 peripheral clock enable in low power mode register.

Definition at line 155 of file [MRCC_private.h](#).

6.12.2.27 Reserved11

`u32_t` Reserved11

Reserved register.

Definition at line 160 of file [MRCC_private.h](#).

6.12.2.28 Reserved12

`u32_t` Reserved12

Reserved register.

Definition at line 165 of file [MRCC_private.h](#).

6.12.2.29 BDCR

`u32_t` BDCR

RCC Backup domain control register.

Definition at line 170 of file [MRCC_private.h](#).

6.12.2.30 CSR

`u32_t` CSR

clock control & status register

Definition at line 175 of file [MRCC_private.h](#).

6.12.2.31 Reserved13

`u32_t` Reserved13

Reserved register.

Definition at line 180 of file [MRCC_private.h](#).

6.12.2.32 Reserved14

`u32_t` Reserved14

Reserved register.

Definition at line 185 of file [MRCC_private.h](#).

6.12.2.33 SSCGR

`u32_t` `SSCGR`

RCC spread spectrum clock generation register.

Definition at line 190 of file [MRCC_private.h](#).

6.12.2.34 PLLI2SCFGR

`u32_t` `PLLI2SCFGR`

PLLI2S configuration register.

Definition at line 195 of file [MRCC_private.h](#).

6.12.2.35 DCKCFGR

`u32_t` `DCKCFGR`

Dedicated Clocks Configuration Register.

Definition at line 200 of file [MRCC_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/RCC/[MRCC_private.h](#)

6.13 SCB_MemoryMapType Struct Reference

System control block memory map structure.

```
#include "COTS/MCAL/NVIC/NVIC_private.h"
```

Data Fields

- [u32_t CPUID](#)
CPUID base register.
- [u32_t ICSR](#)
Interrupt control and state register.
- [u32_t VTOR](#)
Vector table offset register.
- [u32_t AIRCR](#)
Application interrupt and reset control register.
- [u32_t SCR](#)
System control register.
- [u32_t CCR](#)
configuration and control register
- [u32_t SHPR1](#)
System handler priority register 1.
- [u32_t SHPR2](#)
System handler priority register 2.
- [u32_t SHPR3](#)
System handler priority register 3.
- [u32_t SHCSR](#)
System handler control and state register.
- [u32_t CFSR](#)
Configurable fault status register.
- [u32_t HFSR](#)
Hart fault status register.
- [u32_t RESERVED](#)
- [u32_t MMFAR](#)
Memory management fault address register.
- [u32_t BFAR](#)
Bus fault address register.

6.13.1 Detailed Description

System control block memory map structure.

provides system implementation information, and system control.

This includes configuration, control, and reporting of the system exceptions

Definition at line 100 of file [NVIC_private.h](#).

6.13.2 Field Documentation

6.13.2.1 CPUID

`u32_t CPUID`

CPUID base register.

This register contains the the processor part number, version, and implementation information

Definition at line 106 of file [NVIC_private.h](#).

6.13.2.2 ICSR

`u32_t ICSR`

Interrupt control and state register.

This register is responsible for enabling the interrupt

Definition at line 111 of file [NVIC_private.h](#).

6.13.2.3 VTOR

`u32_t VTOR`

Vector table offset register.

Interrupt control and state register

Definition at line 116 of file [NVIC_private.h](#).

6.13.2.4 AIRCR

`u32_t AIRCR`

Application interrupt and reset control register.

This register provides priority grouping control for the exception model, endian status for data accesses, and reset control for the system

To write on this register, password 0x5FA must be written to the VECTKEY field, otherwise the write attempt is ignored

See also

[VECTKEY_PASSWORD](#) for the VECTKEY field password value

Definition at line 123 of file [NVIC_private.h](#).

6.13.2.5 SCR

`u32_t SCR`

System control register.

The SCR controls features of entry to and exit from low power state

Definition at line 128 of file [NVIC_private.h](#).

6.13.2.6 CCR

`u32_t CCR`

configuration and control register

The CCR controls entry to Thread mode and enables:

- The handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore bus faults
- Trapping of divide by zero and unaligned accesses
- Access to the STIR by unprivileged software

Definition at line 136 of file [NVIC_private.h](#).

6.13.2.7 SHPR1

`u32_t SHPR1`

System handler priority register 1.

The SHPR1-SHPR3 registers set the priority level, 0 to 255 of the exception handlers that have configurable priority.

See also

[SHPR2](#)

[SHPR3](#)

Definition at line 143 of file [NVIC_private.h](#).

6.13.2.8 SHPR2

`u32_t SHPR2`

System handler priority register 2.

The SHPR1-SHPR3 registers set the priority level, 0 to 255 of the exception handlers that have configurable priority.

See also

[SHPR1](#)

[SHPR3](#)

Definition at line 150 of file [NVIC_private.h](#).

6.13.2.9 SHPR3

`u32_t SHPR3`

System handler priority register 3.

The SHPR1-SHPR3 registers set the priority level, 0 to 255 of the exception handlers that have configurable priority.

See also

[SHPR1](#)

[SHPR2](#)

Definition at line 157 of file [NVIC_private.h](#).

6.13.2.10 SHCSR

`u32_t SHCSR`

System handler control and state register.

The SHCSR enables the system handlers, and indicates:

- The pending status of the bus fault, memory management fault, and SVC exceptions
- The active status of the system handlers.

Note

If you disable a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault.

Definition at line 165 of file [NVIC_private.h](#).

6.13.2.11 CFSR

`u32_t` CFSR

Configurable fault status register.

This register consists of 3 sub-registers:

- Usage Fault Status Register (UFSR)
- Bus Fault Status Register (BFSR)
- Memory Management Fault Status Register (MMFSR)
System handler control and state register

Definition at line 176 of file [NVIC_private.h](#).

6.13.2.12 HFSR

`u32_t` HFSR

Hard fault status register.

The HFSR gives information about events that activate the hard fault handler.

This register is read, write to clear.

This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0

Definition at line 183 of file [NVIC_private.h](#).

6.13.2.13 RESERVED

`u32_t` RESERVED

Note

This is a reserved register

Definition at line 187 of file [NVIC_private.h](#).

6.13.2.14 MMFAR

`u32_t MMFAR`

Memory management fault address register.

Memory management fault address register

Definition at line 192 of file [NVIC_private.h](#).

6.13.2.15 BFAR

`u32_t BFAR`

Bus fault address register.

Bus fault address register

Definition at line 197 of file [NVIC_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/NVIC/[NVIC_private.h](#)

6.14 SPI_MemoryMapType Struct Reference

SPI Registers.

```
#include "COTS/MCAL/SPI/SPI_interface.h"
```

Data Fields

- volatile `u32_t CR1`
SPI Control Register 1.
- volatile `u32_t CR2`
SPI Control Register 2.
- volatile `u32_t SR`
SPI Status Register.
- volatile `u32_t DR`
SPI Data Register.
- volatile `u32_t CRCPR`
SPI CRC Polynomial Register.
- volatile `u32_t RXCRCR`
SPI RX CRC Register.
- volatile `u32_t TXCRCR`
SPI TX CRC Register.
- volatile `u32_t I2SCFGR`
SPI I2S Configuration Register.
- volatile `u32_t I2SPR`
SPI I2S Prescaler Register.

6.14.1 Detailed Description

SPI Registers.

Definition at line 80 of file [SPI_interface.h](#).

6.14.2 Field Documentation

6.14.2.1 CR1

```
volatile u32_t CR1
```

SPI Control Register 1.

Definition at line 85 of file [SPI_interface.h](#).

6.14.2.2 CR2

```
volatile u32_t CR2
```

SPI Control Register 2.

Definition at line 89 of file [SPI_interface.h](#).

6.14.2.3 SR

```
volatile u32_t SR
```

SPI Status Register.

Definition at line 93 of file [SPI_interface.h](#).

6.14.2.4 DR

```
volatile u32_t DR
```

SPI Data Register.

Definition at line 97 of file [SPI_interface.h](#).

6.14.2.5 CRCPR

```
volatile u32_t CRCPR
```

SPI CRC Polynomial Register.

Definition at line 101 of file [SPI_interface.h](#).

6.14.2.6 RXCRCR

```
volatile u32_t RXCRCR
```

SPI RX CRC Register.

Definition at line 105 of file [SPI_interface.h](#).

6.14.2.7 TXCRCR

```
volatile u32_t TXCRCR
```

SPI TX CRC Register.

Definition at line 109 of file [SPI_interface.h](#).

6.14.2.8 I2SCFGR

```
volatile u32_t I2SCFGR
```

SPI I2S Configuration Register.

Definition at line 113 of file [SPI_interface.h](#).

6.14.2.9 I2SPR

```
volatile u32_t I2SPR
```

SPI I2S Prescaler Register.

Definition at line 117 of file [SPI_interface.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/SPI/SPI_interface.h

6.15 SysTick_Type Struct Reference

Systick memory map structure declaration for the Systick's registers.

```
#include "COTS/MCAL/SysTick/SysTick_private.h"
```

Data Fields

- `u32_t CTRL`
Systick control and status register.
- `u32_t LOAD`
Systick reload value register.
- `u32_t VAL`
Systick current value register.
- `u32_t CALIB`
Systick calibration value register.

6.15.1 Detailed Description

Systick memory map structure declaration for the Systick's registers.

Definition at line 18 of file [SysTick_private.h](#).

6.15.2 Field Documentation

6.15.2.1 CTRL

`u32_t CTRL`

Systick control and status register.

This register is responsible for enabling the systick features

Definition at line 24 of file [SysTick_private.h](#).

6.15.2.2 LOAD

`u32_t LOAD`

Systick reload value register.

This register is responsible for setting the countdown value

Definition at line 29 of file [SysTick_private.h](#).

6.15.2.3 VAL

`u32_t` `VAL`

Systick current value register.

This register holds the current value for the Systick counter

Definition at line 34 of file [SysTick_private.h](#).

6.15.2.4 CALIB

`u32_t` `CALIB`

Systick calibration value register.

SysTick calibration value register

Definition at line 39 of file [SysTick_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/SysTick/[SysTick_private.h](#)

6.16 Task Struct Reference

Data Fields

- `void(* TaskHandler)(void)`
- `u8_t periodicity`
- `Running_enState Task_RunningState`

6.16.1 Detailed Description

Definition at line 20 of file [MyRTOS_private.h](#).

6.16.2 Field Documentation

6.16.2.1 TaskHandler

`void(* TaskHandler) (void)`

Definition at line 23 of file [MyRTOS_private.h](#).

6.16.2.2 periodicity

`u8_t periodicity`

Definition at line 25 of file [MyRTOS_private.h](#).

6.16.2.3 Task_RunningState

`Running_enState Task_RunningState`

Definition at line 27 of file [MyRTOS_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MyRTOS/MyRTOS_private.h

6.17 TIM1_MemoryMapType Struct Reference

TIM1 declaration structure for its registers.

```
#include "COTS/MCAL/TIM1/TIM1_private.h"
```

Data Fields

- volatile `u32_t CR1`
- volatile `u32_t CR2`
- volatile `u32_t SMCR`
- volatile `u32_t DIER`
- volatile `u32_t SR`
- volatile `u32_t EGR`
- volatile `u32_t CCMR1`
- volatile `u32_t CCMR2`
- volatile `u32_t CCER`
- volatile `u32_t CNT`
- volatile `u32_t PSC`
- volatile `u32_t ARR`
- volatile `u32_t RCR`
- volatile `u32_t CCR1`
- volatile `u32_t CCR2`
- volatile `u32_t CCR3`
- volatile `u32_t CCR4`
- volatile `u32_t BDTR`
- volatile `u32_t DCR`
- volatile `u32_t DMAR`

6.17.1 Detailed Description

TIM1 declaration structure for its registers.

Definition at line 18 of file [TIM1_private.h](#).

6.17.2 Field Documentation

6.17.2.1 CR1

```
volatile u32_t CR1
```

Definition at line 21 of file [TIM1_private.h](#).

6.17.2.2 CR2

```
volatile u32_t CR2
```

Definition at line 23 of file [TIM1_private.h](#).

6.17.2.3 SMCR

```
volatile u32_t SMCR
```

Definition at line 25 of file [TIM1_private.h](#).

6.17.2.4 DIER

```
volatile u32_t DIER
```

Definition at line 27 of file [TIM1_private.h](#).

6.17.2.5 SR

```
volatile u32_t SR
```

Definition at line 29 of file [TIM1_private.h](#).

6.17.2.6 EGR

```
volatile u32_t EGR
```

Definition at line 31 of file [TIM1_private.h](#).

6.17.2.7 CCMR1

```
volatile u32_t CCMR1
```

Definition at line 33 of file [TIM1_private.h](#).

6.17.2.8 CCMR2

```
volatile u32_t CCMR2
```

Definition at line 35 of file [TIM1_private.h](#).

6.17.2.9 CCER

```
volatile u32_t CCER
```

Definition at line 37 of file [TIM1_private.h](#).

6.17.2.10 CNT

```
volatile u32_t CNT
```

Definition at line 39 of file [TIM1_private.h](#).

6.17.2.11 PSC

```
volatile u32_t PSC
```

Definition at line 41 of file [TIM1_private.h](#).

6.17.2.12 ARR

```
volatile u32_t ARR
```

Definition at line 43 of file [TIM1_private.h](#).

6.17.2.13 RCR

```
volatile u32_t RCR
```

Definition at line 45 of file [TIM1_private.h](#).

6.17.2.14 CCR1

```
volatile u32_t CCR1
```

Definition at line 47 of file [TIM1_private.h](#).

6.17.2.15 CCR2

```
volatile u32_t CCR2
```

Definition at line 49 of file [TIM1_private.h](#).

6.17.2.16 CCR3

```
volatile u32_t CCR3
```

Definition at line 51 of file [TIM1_private.h](#).

6.17.2.17 CCR4

```
volatile u32_t CCR4
```

Definition at line 53 of file [TIM1_private.h](#).

6.17.2.18 BDTR

```
volatile u32_t BDTR
```

Definition at line 55 of file [TIM1_private.h](#).

6.17.2.19 DCR

```
volatile u32_t DCR
```

Definition at line 57 of file [TIM1_private.h](#).

6.17.2.20 DMAR

```
volatile u32_t DMAR
```

Definition at line 59 of file [TIM1_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/TIM1/TIM1_private.h

6.18 USART_ClockInitTypeDef Struct Reference

USART Clock declaration structure for its registers.

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- [u8_t ClockOutput](#)
USART Clock Enable.
- [u8_t ClockPolarity](#)
USART Clock Polarity.
- [u8_t ClockPhase](#)
USART Clock Phase.
- [u8_t LastBitClockPulse](#)
USART Last Bit Clock Pulse.

6.18.1 Detailed Description

USART Clock declaration structure for its registers.

Definition at line 149 of file [UART_interface.h](#).

6.18.2 Field Documentation

6.18.2.1 ClockOutput

`u8_t ClockOutput`

USART Clock Enable.

Definition at line 154 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.18.2.2 ClockPolarity

`u8_t ClockPolarity`

USART Clock Polarity.

Definition at line 158 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.18.2.3 ClockPhase

`u8_t ClockPhase`

USART Clock Phase.

Definition at line 162 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.18.2.4 LastBitClockPulse

`u8_t LastBitClockPulse`

USART Last Bit Clock Pulse.

Definition at line 166 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/UART/[UART_interface.h](#)

6.19 USART_InitType Struct Reference

USART Clock declaration structure for.

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- `u32_t BaudRate`
UART Baud Rate.
- `u8_t DataWidth`
UART Data Width.
- `u8_t StopBits`
UART Stop Bits.
- `u8_t Parity_Enable`
UART Parity Enable.
- `u8_t Parity_Selection`
UART Parity Selection.
- `u8_t TransferDirection`
UART Transfer Direction.
- `u8_t HardwareFlowControl`
UART Hardware Flow Control.
- `u8_t Oversampling`
UART Oversampling.

6.19.1 Detailed Description

USART Clock declaration structure for.

Definition at line 108 of file [UART_interface.h](#).

6.19.2 Field Documentation

6.19.2.1 BaudRate

```
u32_t BaudRate
```

UART Baud Rate.

Definition at line 113 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.2 DataWidth

`u8_t` DataWidth

UART Data Width.

Definition at line 117 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.3 StopBits

`u8_t` StopBits

UART Stop Bits.

Definition at line 121 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.4 Parity_Enable

`u8_t` Parity_Enable

UART Parity Enable.

Definition at line 125 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.5 Parity_Selection

`u8_t` Parity_Selection

UART Parity Selection.

Definition at line 129 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.6 TransferDirection

`u8_t TransferDirection`

UART Transfer Direction.

Note

Accepted values are: TX_ONLY, RX_ONLY, TX_RX

Definition at line 134 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.19.2.7 HardwareFlowControl

`u8_t HardwareFlowControl`

UART Hardware Flow Control.

Definition at line 138 of file [UART_interface.h](#).

6.19.2.8 Oversampling

`u8_t Oversampling`

UART Oversampling.

Definition at line 142 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/UART/[UART_interface.h](#)

6.20 USART_MemoryMapType Struct Reference

USART declaration structure for its registers.

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- volatile `u32_t SR_REG`
USART Status Register.
- volatile `u32_t DR_REG`
USART Data Register.
- volatile `u32_t BRR_REG`
USART Baud Rate Register.
- volatile `u32_t CR1_REG`
USART Control Register 1.
- volatile `u32_t CR2_REG`
USART Control Register 2.
- volatile `u32_t CR3_REG`
USART Control Register 3.
- volatile `u32_t GTPR_REG`
USART Guard time and prescaler register.

6.20.1 Detailed Description

USART declaration structure for its registers.

Definition at line 71 of file [UART_interface.h](#).

6.20.2 Field Documentation

6.20.2.1 SR_REG

```
volatile u32_t SR_REG
```

USART Status Register.

Definition at line 76 of file [UART_interface.h](#).

Referenced by [MUSART_u8ReceiveByteSynchBlocking\(\)](#), [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#), [MUSART_vClearFlags\(\)](#), [MUSART_vInit\(\)](#), and [MUSART_vTransmitByte\(\)](#).

6.20.2.2 DR_REG

```
volatile u32_t DR_REG
```

USART Data Register.

Definition at line 80 of file [UART_interface.h](#).

Referenced by [MUSART_u8ReadDataRegister\(\)](#), [MUSART_u8ReceiveByteSynchBlocking\(\)](#), [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#) and [MUSART_vTransmitByte\(\)](#).

6.20.2.3 BRR_REG

```
volatile u32_t BRR_REG
```

USART Baud Rate Register.

Definition at line 84 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.20.2.4 CR1_REG

```
volatile u32_t CR1_REG
```

USART Control Register 1.

Definition at line 88 of file [UART_interface.h](#).

Referenced by [MUSART_vDisable\(\)](#), [MUSART_vEnable\(\)](#), [MUSART_vInit\(\)](#), and [MUSART_vRxIntSetStatus\(\)](#).

6.20.2.5 CR2_REG

```
volatile u32_t CR2_REG
```

USART Control Register 2.

Definition at line 92 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.20.2.6 CR3_REG

```
volatile u32_t CR3_REG
```

USART Control Register 3.

Definition at line 96 of file [UART_interface.h](#).

6.20.2.7 GTPR_REG

```
volatile u32_t GTPR_REG
```

USART Guard time and prescaler register.

Definition at line 100 of file [UART_interface.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/UART/[UART_interface.h](#)

Chapter 7

File Documentation

7.1 ACC_config.h

```
00001 /* FILENAME: ACC_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 04/13/2023
00005 */
00006 #ifndef _ACC_config_H
00007 #define _ACC_config_H
00008
00009 #define ACC_SAFE_DIST 50 // (cm)
00010
00011 #endif // _ACC_config_H
```

7.2 ACC_interface.h

```
00001 /* FILENAME: ACC_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 04/13/2023
00005 */
00006 #ifndef _ACC_interface_H
00007 #define _ACC_interface_H
00008
00009
00010 /***** Functions prototypes *****/
00011 /* Functions prototypes */
00012 /***** */
00013
00014
00015 //void AACC_vInit( void ) ;
00016
00017 void AACC_vModeON( void ) ;
00018
00019
00020 #endif // _ACC_interface_H
```

7.3 ACC_private.h

```
00001 /* FILENAME: ACC_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 04/13/2023
00005 */
00006 #ifndef _ACC_private_H
00007 #define _ACC_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _ACC_private_H
```

7.4 ACC_program.c

```

00001 /* FILENAME: ACC_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 04/13/2023
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/GPIO/GPIO_interface.h"
00017
00018 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00019 #include "../../HAL/DCMOTOR/DCM_interface.h"
00020 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00021 #include "../../HAL/UltraSonic/UltraSonic_interface.h"
00022
00023 #include "../Mob_APP/Mob_APP_interface.h"
00024
00025 #include "ACC_interface.h"
00026 #include "ACC_private.h"
00027 #include "ACC_config.h"
00028
00029 u32_t L_u32SpeedValue          = INITIAL_ZERO ;
00030 f32_t L_f32Distance           = INITIAL_ZERO ;
00031 c8_t volatile G_c8ACCRecievedButton = INITIAL_ZERO ;
00032 u8_t volatile Gv_u8InISR      = FALSE ;
00033
00034 static VAR(HULTSNC_ConfigType)
00035 TRIG =
00036 {
00037     .u8Port = GPIO_PORTB ,
00038     .u8Pin  = GPIOx_PIN8
00039 };
00040
00041 /*****
00042 *****/
00043
00044 /* This is a call back RX function */
00045 static void Button_vCurrentData( void )
00046 {
00047     G_c8ACCRecievedButton = HBluetooth_u8GetDataRegister() ;
00048
00049     Gv_u8InISR           = TRUE
00050 }
00051
00052 /*****
00053 *****/
00054
00055 void AACC_vModeON( void )
00056 {
00057
00058     G_c8ACCRecievedButton = INITIAL_ZERO ;
00059     Gv_u8InISR           = FALSE ;
00060
00061     HBluetooth_vSendString( "\nACC Mode ON\n" );
00062
00063     HBluetooth_vEnableAsynchReceive( ) ;
00064
00065     HBluetooth_u8AsynchReceiveByte( &Button_vCurrentData ) ;
00066
00067 do
00068 {
00069
00070     HULTSNC_vTrigger( &TRIG ) ;
00071
00072     L_f32Distance = HULTSNC_f32GetDistance( ) ;
00073
00074     if( L_f32Distance < ACC_SAFE_DIST )
00075     {
00076         HCarMove_vStop( ) ;
00077
00078         HBluetooth_vSendString( SPEED0_STR ) ;
00079     }
00080     else if( L_f32Distance >= ACC_SAFE_DIST + HYSTRS_VALUE )
00081     {
00082         HCarMove_vForward( ) ;
00083
00084         L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00085

```

```

00086         AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00087     }
00088     else
00089     {
00090         /* Do Nothing */
00091     }
00092
00093     if( Gv_u8InISR == TRUE )
00094     {
00095
00096         Gv_u8InISR = FALSE ;
00097
00098         switch( G_c8ACCRecievedButton )
00099         {
00100
00101             case EXIT_MODE_CHAR:
00102                 HCarMove_vStop( ) ;
00103
00104                 HBluetooth_vSendString( SPEED0_STR ) ;
00105
00106                 HBluetooth_vSendString( "\nACC Mode Off\n" ) ;
00107
00108                 HBluetooth_u8AsynchReceiveByte( NULL ) ;
00109
00110             break ;
00111
00112             case INC_SPEED_CHAR:
00113
00114                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00115
00116                 if( L_u32SpeedValue >= SPEED_100_PERCENT )
00117                 {
00118                     L_u32SpeedValue = SPEED_100_PERCENT ;
00119
00120                     AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00121
00122                 }
00123                 else
00124                 {
00125                     L_u32SpeedValue += SPEED_10_PERCENT ;
00126
00127                     HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00128
00129                     AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00130
00131                 }
00132
00133             break ;
00134
00135             case DEC_SPEED_CHAR:
00136
00137                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00138
00139                 if( L_u32SpeedValue <= SPEED_0_PERCENT )
00140                 {
00141                     L_u32SpeedValue = SPEED_0_PERCENT ;
00142
00143                     AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00144
00145                 }
00146                 else
00147
00148                     L_u32SpeedValue -= SPEED_10_PERCENT ;
00149
00150                     HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00151
00152
00153             break ;
00154
00155             default: /* Do Nothing */ break ;
00156         }
00157
00158     }
00159     else
00160     {
00161         /* Do nothing */
00162     }
00163
00164 }while( G_c8ACCRecievedButton != EXIT_MODE_CHAR ) ;
00165
00166 }
00167
00168 /***** *****/
00169 /***** *****/
00170
00171
00172

```

```

00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193

```

7.5 Exit_State_config.h

```

00001 /* FILENAME: Exit_State_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Exit_State_config_H
00007 #define _Exit_State_config_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Exit_State_config_H

```

7.6 Exit_State_interface.h

```

00001 /* FILENAME: Exit_State_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Exit_State_interface_H
00007 #define _Exit_State_interface_H
00008
00009 /*****
00010 /* Functions prototypes */
00011 *****/
00012
00013 void AExit_vCriteriaON( void ) ;
00014
00015
00016 #endif // _Exit_State_interface_H

```

7.7 Exit_State_private.h

```

00001 /* FILENAME: Exit_State_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Exit_State_private_H
00007 #define _Exit_State_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Exit_State_private_H

```

7.8 Exit_State_program.c

```
00001 /* FILENAME: Exit_State_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/GPIO/GPIO_interface.h"
00017
00018 #include "../../HAL/BUZZER/BUZZER_interface.h"
00019 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00020 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00021
00022 #include "Exit_State_interface.h"
00023 #include "Exit_State_private.h"
00024 #include "Exit_State_config.h"
00025
00026
00027 static VAR(BUZZER_BuzzerConfiguration)
00028     Buzzer =
00029     {
00030         .u8Port = GPIO_PORTB ,
00031         .u8Pin  = GPIOX_PIN5
00032     } ;
00033
00034 /***** Function Implementation *****/
00035 /***** Function Implementation *****/
00036
00037 void AExit_vCriteriaON( void )
00038 {
00039
00040     HBluetooth_vSendString( "\nActivate Exit Criteria\n" ) ;
00041
00042     HCarMove_vStop( ) ;
00043
00044     HBUZZER_vSoundOff( &Buzzer ) ;
00045
00046 }
00047
00048 /***** Function Implementation *****/
00049 /***** Function Implementation *****/
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
```

00086

7.9 FCW_config.h

```
00001 /* FILENAME: FCW_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _FCW_config_H
00007 #define _FCW_config_H
00008
00009
00010 #define FCW_SAFE_DIST 80 // (cm)
00011
00012
00013
00014 #endif //_FCW_config_H
```

7.10 FCW_interface.h

```
00001 /* FILENAME: FCW_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _FCW_interface_H
00007 #define _FCW_interface_H
00008
00009
00010 /***** Functions prototypes *****/
00011 /* Functions prototypes */
00012 /***** *****/
00013
00014 void AFCW_vModeON( void ) ;
00015
00016
00017 #endif //_FCW_interface_H
```

7.11 FCW_private.h

```
00001 /* FILENAME: FCW_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _FCW_private_H
00007 #define _FCW_private_H
00008
00009
00010
00011
00012
00013
00014 #endif //_FCW_private_H
```

7.12 FCW_program.c

```
00001 /* FILENAME: FCW_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006
00007 /***** *****/
00008 /* Include headers */
00009 /***** *****/
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
```

```

00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/GPIO/GPIO_interface.h"
00017
00018 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00019 #include "../../HAL/BUZZER/BUZZER_interface.h"
00020 #include "../../HAL/DCMOTOR/DCM_interface.h"
00021 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00022 #include "../../HAL/UltraSonic/UltraSonic_interface.h"
00023
00024 #include "../Mob_APP/Mob_APP_interface.h"
00025
00026 #include "FCW_interface.h"
00027 #include "FCW_private.h"
00028 #include "FCW_config.h"
00029
00030 c8_t volatile G_c8RecievedButton = INITIAL_ZERO ;
00031 u8_t volatile GV_u8InISR = FALSE ;
00032
00033 static VAR(HULTSNC_ConfigType)
00034 TRIG =
00035 {
00036     .u8Port = GPIO_PORTB ,
00037     .u8Pin = GPIOx_PIN8
00038 };
00039
00040 static VAR(BUZZER_BuzzerConfiguration)
00041 Buzzer =
00042 {
00043     .u8Port = GPIO_PORTB ,
00044     .u8Pin = GPIOx_PIN5
00045 } ;
00046
00047 /*****
00048 *****/
00049
00050 /* This is a call back RX function */
00051 static void Button_vCurrentData( void )
00052 {
00053     G_c8RecievedButton = HBluetooth_u8GetDataRegister( ) ;
00054
00055     GV_u8InISR = TRUE ;
00056 }
00057
00058 /*****
00059 *****/
00060
00061 void AFCW_vModeON( void )
00062 {
00063
00064     u32_t L_u32SpeedValue = INITIAL_ZERO ;
00065     f32_t L_f32Distance = INITIAL_ZERO ;
00066     G_c8RecievedButton = INITIAL_ZERO ;
00067     GV_u8InISR = FALSE ;
00068
00069     HBUZZER_vInit( &Buzzer ) ;
00070
00071     HBluetooth_vEnableAsynchReceive( ) ;
00072
00073     HBluetooth_u8AsynchReceiveByte( &Button_vCurrentData ) ;
00074
00075     HBluetooth_vSendString( "\nFCW Mode ON\n" ) ;
00076
00077     do
00078     {
00079
00080         HULTSNC_vTrigger( &TRIG ) ;
00081
00082         L_f32Distance = HULTSNC_f32GetDistance( ) ;
00083
00084         if( L_f32Distance < FCW_SAFE_DIST )
00085         {
00086             HBUZZER_vSoundOn( &Buzzer ) ;
00087             HBluetooth_vSendString( "\nWarning there's an object 80cm away\n" ) ;
00088         }
00089         else if( L_f32Distance >= FCW_SAFE_DIST + HYSTRS_VALUE )
00090         {
00091             HBUZZER_vSoundOff( &Buzzer ) ;
00092         }
00093         else
00094         {
00095             /* Do Nothing */
00096         }
00097
00098         if( GV_u8InISR == TRUE )
00099         {
00100

```

```

00101     GV_u8InISR = FALSE ;
00102
00103     switch( G_c8RecievedButton )
00104     {
00105
00106         case FORW_CHAR:
00107
00108             HCarMove_vForward( ) ;
00109
00110             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00111
00112             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00113
00114             break ;
00115
00116         case BACKW_CHAR:
00117
00118             HCarMove_vBackward( ) ;
00119
00120             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00121
00122             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00123
00124             break ;
00125
00126         case RIGHT_CHAR:
00127
00128             HCarMove_vRight( ) ;
00129
00130             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00131
00132             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00133
00134             break ;
00135
00136         case LEFT_CHAR:
00137
00138             HCarMove_vLeft( ) ;
00139
00140             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00141
00142             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00143
00144             break ;
00145
00146         case STOP_CHAR:
00147
00148             HCarMove_vStop( ) ;
00149
00150             HBluetooth_vSendString( SPEED0_STR ) ;
00151
00152             break ;
00153
00154         case EXIT_MODE_CHAR:
00155
00156             HBUZZER_vSoundOff( &Buzzer ) ;
00157
00158             HCarMove_vStop( ) ;
00159
00160             HBluetooth_vSendString( SPEED0_STR ) ;
00161
00162             HBluetooth_u8AsynchReceiveByte( NULL ) ;
00163
00164             break ;
00165
00166         case INC_SPEED_CHAR:
00167
00168             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00169
00170             if( L_u32SpeedValue >= SPEED_100_PERCENT )
00171             {
00172                 L_u32SpeedValue = SPEED_100_PERCENT ;
00173
00174                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00175             }
00176             else
00177             {
00178                 L_u32SpeedValue += SPEED_10_PERCENT ;
00179
00180                 HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00181
00182                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00183             }
00184
00185             break ;
00186
00187         case DEC_SPEED_CHAR:

```

```

00188
00189     L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00190
00191     if( L_u32SpeedValue <= SPEED_0_PERCENT )
00192     {
00193         L_u32SpeedValue = SPEED_0_PERCENT ;
00194
00195         AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00196     }
00197     else
00198     {
00199         L_u32SpeedValue -= SPEED_10_PERCENT ;
00200
00201         HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00202
00203         AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00204     }
00205
00206     break ;
00207
00208     default: /* Do Nothing */ break ;
00209 }
00210
00211 }
00212 else
00213 {
00214     /* Do nothing */
00215 }
00216
00217 }while( G_c8RecievedButton != EXIT_MODE_CHAR ) ;
00218
00219 }
00220
00221 *****
00222 *****
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253

```

7.13 Mob_APP_config.h

```

00001 /* FILENAME: Mob_APP_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Sun 03/26/2023
00005 */
00006 #ifndef _Mob_APP_Config_H
00007 #define _Mob_APP_Config_H
00008
00009
00010
00011
00012 #endif // _Mob_APP_Config_H

```

7.14 Mob_APP_interface.h

```

00001 /* FILENAME: Mob_APP_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Sun 03/26/2023
00005 */
00006 #ifndef _Mob_APP_interface_H
00007 #define _Mob_APP_interface_H
00008
00009
00010 #define DECIMAL 10
00011 #define SPEED0_STR "*S0*"
00012
00013 #define TRAD_MODE_CHAR 'T'
00014 #define NCC_MODE_CHAR 'N'
00015 #define ACC_MODE_CHAR 'A'
00016 #define FCW_MODE_CHAR 'F'
00017 #define EXIT_SYS_CHAR 'E'
00018
00019 #define FORW_CHAR 'f'
00020 #define BACKW_CHAR 'b'
00021 #define RIGHT_CHAR 'r'
00022 #define LEFT_CHAR 'l'
00023 #define STOP_CHAR 's'
00024 #define EXIT_MODE_CHAR 'e'
00025 #define INC_SPEED_CHAR '+'
00026 #define DEC_SPEED_CHAR '-'
00027
00028 #define HYSTRS_VALUE 3
00029
00030 /***** Functions prototypes *****/
00031 /* */
00032 /***** */
00033
00034 void AMobApp_vInit( void ) ;
00035
00036 void AMobApp_vSendSpeedValue( u32_t A_u32SpeedValue ) ;
00037
00038 void AMobApp_vCntrlCar( void ) ;
00039
00040
00041
00042 #endif // _Mob_APP_interface_H

```

7.15 Mob_APP_private.h

```

00001 /* FILENAME: Mob_APP_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Sun 03/26/2023
00005 */
00006 #ifndef _Mob_APP_private_H
00007 #define _Mob_APP_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Mob_APP_private_H

```

7.16 Mob_APP_program.c

```

00001 /* FILENAME: Mob_APP_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Sun 03/26/2023
00005 */
00006
00007 /***** Include headers *****/
00008 /* */
00009 /***** */
00010
00011 #include "stdlib.h"
00012 #include "string.h"
00013
00014 #include "../../LIB/LSTD_TYPES.h"
00015 #include "../../LIB/LSTD_COMPILER.h"

```

```

00016 #include "../../LIB/LSTD_VALUES.h"
00017 #include "../../LIB/LSTD_BITMATH.h"
00018
00019 #include "../../MCAL/GPIO/GPIO_interface.h"
00020
00021 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00022 #include "../../HAL/DCMOTOR/DCM_interface.h"
00023 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00024 #include "../../HAL/UltraSonic/UltraSonic_interface.h"
00025 #include "../../HAL/BUZZER/BUZZER_interface.h"
00026
00027 #include "../Traditional_Mode/Traditional_Mode_interface.h"
00028 #include "../NCC/NCC_interface.h"
00029 #include "../ACC/ACC_interface.h"
00030 #include "../FCW/FCW_interface.h"
00031 #include "../Exit_State/Exit_State_interface.h"
00032
00033 #include "Mob_APP_interface.h"
00034 #include "Mob_APP_private.h"
00035 #include "Mob_APP_config.h"
00036
00037 /***** Functions implementations *****/
00038 /*          Functions implementations          */
00039 /***** Functions implementations *****/
00040
00041 void AMobApp_vInit( void )
00042 {
00043
00044     // Initialization of Bluetooth Module
00045     HBluetooth_vInit( ) ;
00046
00047     HBluetooth_vSendString( "\nSuccessfully connected\n" ) ;
00048
00049 }
00050
00051 /***** Functions implementations *****/
00052 /***** Functions implementations *****/
00053 void AMobApp_vSendSpeedValue( u32_t A_u32SpeedValue )
00054 {
00055
00056     char L_strSpeedIndc[9] = "*S" ;
00057
00058     char L_strSpeedStr[6] ;
00059
00060     // Convert the speed value from integer to string and store it in speedStr.
00061     itoa( A_u32SpeedValue, L_strSpeedStr, DECIMAL ) ;
00062
00063     // Concatenate "*S" with the speed value string to result a string like that "*S100".
00064     strcat( L_strSpeedIndc, L_strSpeedStr ) ;
00065
00066     // Concatenate speed value string with "*" to result a string like that "*S100*"
00067     strcat( L_strSpeedIndc, "*" ) ;
00068
00069     // Then send this string through bluetooth to the speed indicator.
00070     HBluetooth_vSendString( L_strSpeedIndc ) ;
00071
00072 }
00073
00074 /***** Functions implementations *****/
00075 /***** Functions implementations *****/
00076
00077 void AMobApp_vCntrlCar( void )
00078 {
00079
00080     c8_t L_c8RecievedState = TRAD_MODE_CHAR ;
00081
00082     HBluetooth_vSendString( "\nCar is ready to be controlled\n" ) ;
00083
00084     while( TRUE )
00085     {
00086
00087         HBluetooth_vDisableAsynchReceive( ) ;
00088
00089         HBluetooth_vSendString( "\nSelect one of these modes:\n" ) ;
00090         HBluetooth_vSendString( "\n1-Traditional Mode\n" ) ;
00091         HBluetooth_vSendString( "\n2-NCC Mode\n" ) ;
00092         HBluetooth_vSendString( "\n3-ACC Mode\n" ) ;
00093         HBluetooth_vSendString( "\n4-FCW Mode\n" ) ;
00094         HBluetooth_vSendString( "\n5-Exit Mode\n" ) ;
00095
00096         L_c8RecievedState = HBluetooth_u8ReceiveByte( ) ;
00097
00098         switch( L_c8RecievedState )
00099         {
00100             case TRAD_MODE_CHAR : ATraditional_vModeON( ) ; break ;
00101

```

```

00103     case NCC_MODE_CHAR : ANCC_vModeON( ) ; break ;
00104     case ACC_MODE_CHAR : AACC_vModeON( ) ; break ;
00105     case FCW_MODE_CHAR : AFCW_vModeON( ) ; break ;
00106     default           : ATraditional_vModeON( ) ; break ;
00107
00108 }
00109
00110 }
00111
00112 }
00113 }
00114
00115 }
00116
00117 /******
00118 *****/
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158

```

7.17 NCC_config.h

```

00001 /* FILENAME: NCC_config
00002 * Author:    Ali El Bana
00003 * Version:   V1.0
00004 * DATE:     Mon 04/17/2023
00005 */
00006 #ifndef _NCC_config_H
00007 #define _NCC_config_H
00008
00009
00010
00011
00012
00013
00014 #endif // _NCC_config_H

```

7.18 NCC_interface.h

```

00001 /* FILENAME: NCC_interface
00002 * Author:    Ali El Bana
00003 * Version:   V1.0

```

```

00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _NCC_interface_H
00007 #define _NCC_interface_H
00008
00009
00010 /***** Functions prototypes *****/
00011 /* Functions prototypes */
00012 /***** Functions prototypes *****/
00013
00014
00015 void ANCC_vModeON( void ) ;
00016
00017
00018 #endif // _NCC_interface_H

```

7.19 NCC_private.h

```

00001 /* FILENAME: NCC_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _NCC_private_H
00007 #define _NCC_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _NCC_private_H

```

7.20 NCC_program.c

```

00001 /* FILENAME: NCC_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006
00007
00008 /***** Include headers *****/
00009 /* Include headers */
00010 /***** Include headers *****/
00011
00012 #include "../../LIB/LSTD_TYPES.h"
00013 #include "../../LIB/LSTD_COMPILER.h"
00014 #include "../../LIB/LSTD_VALUES.h"
00015 #include "../../LIB/LSTD_BITMATH.h"
00016
00017 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00018 #include "../../HAL/DCMOTOR/DCM_interface.h"
00019 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00020
00021 #include "../Mob_APP/Mob_APP_interface.h"
00022
00023 #include "NCC_interface.h"
00024 #include "NCC_private.h"
00025 #include "NCC_config.h"
00026
00027 /***** Function definitions *****/
00028 /***** Function definitions *****/
00029
00030 void ANCC_vModeON( void )
00031 {
00032
00033     u32_t L_u32SpeedValue      = INITIAL_ZERO ;
00034     c8_t   L_c8RecievedButton  = INITIAL_ZERO ;
00035
00036     HBluetooth_vSendString( "\nNCC Mode ON\n" ) ;
00037
00038     do
00039     {
00040         HCarMove_vForward( ) ;
00041
00042         L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00043
00044

```

```

00045     AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00046
00047     L_c8RecievedButton = HBluetooth_u8ReceiveByte( ) ;
00048
00049     switch( L_c8RecievedButton )
00050     {
00051
00052         case EXIT_MODE_CHAR:
00053
00054             HCarMove_vStop( ) ;
00055
00056             HBluetooth_vSendString( SPEED0_STR ) ;
00057
00058             HBluetooth_vSendString( "\nNCC Mode Off\n" ) ;
00059
00060             break ;
00061
00062         case INC_SPEED_CHAR:
00063
00064             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00065
00066             if( L_u32SpeedValue >= SPEED_100_PERCENT )
00067             {
00068                 L_u32SpeedValue = SPEED_100_PERCENT ;
00069
00070                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00071             }
00072             else
00073             {
00074                 L_u32SpeedValue += SPEED_10_PERCENT ;
00075
00076                 HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00077
00078                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00079             }
00080
00081             break ;
00082
00083         case DEC_SPEED_CHAR:
00084
00085             L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00086
00087             if( L_u32SpeedValue <= SPEED_0_PERCENT )
00088             {
00089                 L_u32SpeedValue = SPEED_0_PERCENT ;
00090
00091                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00092             }
00093             else
00094             {
00095                 L_u32SpeedValue -= SPEED_10_PERCENT ;
00096
00097                 HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00098
00099                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00100             }
00101
00102             break ;
00103
00104             default: /* Do Nothing */ break ;
00105     }
00106
00107 }while( L_c8RecievedButton != EXIT_MODE_CHAR ) ;
00108
00109 }

00110 ****
00111 ****
00112 ****
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131

```

```
00132
00133
00134
00135
00136
00137
00138
00139
00140
```

7.21 Traditional_Mode_config.h

```
00001 /* FILENAME: Traditional_Mode_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Traditional_Mode_config_H
00007 #define _Traditional_Mode_config_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Traditional_Mode_config_H
```

7.22 Traditional_Mode_interface.h

```
00001 /* FILENAME: Traditional_Mode_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Traditional_Mode_interface_H
00007 #define _Traditional_Mode_interface_H
00008
00009
00010 /***** Functions prototypes *****/
00011 /* Functions prototypes */
00012 /***** *****/
00013
00014 void ATraditional_vModeON( void ) ;
00015
00016
00017 #endif // _Traditional_Mode_interface_H
00018
00019
00020
```

7.23 Traditional_Mode_private.h

```
00001 /* FILENAME: Traditional_Mode_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006 #ifndef _Traditional_Mode_private_H
00007 #define _Traditional_Mode_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Traditional_Mode_private_H
```

7.24 Traditional_Mode_program.c

```
00001 /* FILENAME: Traditional_Mode_program
00002 * Author: Ali El Bana
```

```

00003 * Version: V1.0
00004 * DATE: Mon 04/17/2023
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00017 #include "../../HAL/DCMOTOR/DCM_interface.h"
00018 #include "../../HAL/Car_Movement/Car_Movement_interface.h"
00019
00020 #include "../Mob_APP/Mob_APP_interface.h"
00021
00022 #include "Traditional_Mode_interface.h"
00023 #include "Traditional_Mode_private.h"
00024 #include "Traditional_Mode_config.h"
00025
00026 /***** */
00027 /***** */
00028
00029 void ATraditional_vModeON( void )
00030 {
00031
00032     u32_t L_u32SpeedValue      = INITIAL_ZERO ;
00033     c8_t  L_c8RecievedButton   = INITIAL_ZERO ;
00034
00035     HBluetooth_vSendString( "\nTraditional Mode ON\n" ) ;
00036
00037     do
00038     {
00039
00040         L_c8RecievedButton = HBluetooth_u8ReceiveByte( ) ;
00041
00042         switch( L_c8RecievedButton )
00043         {
00044
00045             case FORW_CHAR:
00046
00047                 HCarMove_vForward( ) ;
00048
00049                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00050
00051                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00052
00053                 break ;
00054
00055             case BACKW_CHAR:
00056
00057                 HCarMove_vBackward( ) ;
00058
00059                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00060
00061                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00062
00063                 break ;
00064
00065             case RIGHT_CHAR:
00066
00067                 HCarMove_vRight( ) ;
00068
00069                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00070
00071                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00072
00073                 break ;
00074
00075             case LEFT_CHAR:
00076
00077                 HCarMove_vLeft( ) ;
00078
00079                 L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00080
00081                 AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00082
00083                 break ;
00084
00085             case STOP_CHAR:
00086
00087                 HCarMove_vStop( ) ;
00088
00089                 HBluetooth_vSendString( SPEED0_STR ) ;

```

```

00090         break ;
00091
00092     case EXIT_MODE_CHAR:
00093         HCarMove_vStop( ) ;
00094
00095         HBluetooth_vSendString( SPEED0_STR ) ;
00096
00097         HBluetooth_vSendString( "\nTraditional Mode Off\n" ) ;
00098
00099
00100     break ;
00101
00102     case INC_SPEED_CHAR:
00103
00104         L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00105
00106         if( L_u32SpeedValue >= SPEED_100_PERCENT )
00107         {
00108             L_u32SpeedValue = SPEED_100_PERCENT ;
00109
00110             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00111
00112         }
00113         else
00114         {
00115             L_u32SpeedValue += SPEED_10_PERCENT ;
00116
00117             HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00118
00119             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00120
00121         }
00122
00123     break ;
00124
00125     case DEC_SPEED_CHAR:
00126
00127         L_u32SpeedValue = HCarMove_u32GetCarSpeed( ) ;
00128
00129         if( L_u32SpeedValue <= SPEED_0_PERCENT )
00130         {
00131             L_u32SpeedValue = SPEED_0_PERCENT ;
00132
00133             AMobApp_vSendSpeedValue( L_u32SpeedValue ) ;
00134
00135         }
00136         else
00137         {
00138             L_u32SpeedValue -= SPEED_10_PERCENT ;
00139
00140             HCarMove_vSpeedRatio( L_u32SpeedValue ) ;
00141
00142         }
00143
00144     break ;
00145     default: /* Do Nothing */ break ;
00146 }
00147
00148 }while( L_c8RecievedButton != EXIT_MODE_CHAR ) ;
00149
00150 }
00151
00152 /***** */
00153 /***** */
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168

```

7.25 Bluetooth_config.c

```
00001 /*
```

```

00002 * Bluetooth_config.c
00003 *
00004 * Created on: Jan 5, 2023
00005 * Author: Ali El Bana
00006 */
00007 /***** Include headers *****/
00008 /*           Configuration parameters           */
00009 /*****
0010
0011 #include "../../LIB/LSTD_TYPES.h"
0012 #include "../../LIB/LSTD_COMPILER.h"
0013 #include "../../LIB/LSTD_VALUES.h"
0014 #include "../../LIB/LSTD_BITMATH.h"
0015
0016 #include "../../MCAL/RCC/MRCC_interface.h"
0017 #include "../../MCAL/GPIO/GPIO_interface.h"
0018 #include "../../MCAL/UART/UART_interface.h"
0019
0020 #include "Bluetooth_config.h"
0021
0022 /***** USART_InitStructure definition *****/
0023 /*           Configuration parameters           */
0024 /*****
0025
0026 USART_InitType Bluetooth_Config =
0027 {
0028     .BaudRate      = __BAUDRATE__ ,     .DataWidth      = MODE_8BIT ,
0029
0030     .StopBits       = STOP_BIT_1      ,     .Parity_Enable   = DISABLE ,
0031
0032     .Parity_Selection = EVEN_PARITY ,     .TransferDirection = TX_RX ,
0033
0034     .HardwareFlowControl = DISABLE ,     .Oversampling    = OVER_SAMPLING_16
0035 } ;
0036
0037 USART_ClockInitTypeDef Bluetooth_CLK = { DISABLE, 0, 0, 0 } ;
0038
0039
0040 /*****
0041
0042 USART_MemoryMapType * Bluetooth_UART_ID = USART1_REG ;
0043
0044
0045 /*****
0046 /*****
0047

```

7.26 Bluetooth_config.h

```

00001 /* FILENAME: Bluetooth_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 01/05/2023
00005 */
00006 #ifndef _Bluetooth_config_H
00007 #define _Bluetooth_config_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Bluetooth_config_H

```

7.27 Bluetooth_interface.h

```

00001 /* FILENAME: Bluetooth_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 01/05/2023
00005 */
00006 #ifndef _Bluetooth_interface_H
00007 #define _Bluetooth_interface_H
00008
00009 /*****

```

```

00010 /* Functions prototypes */
00011 /***** */
00012
00013 void HBluetooth_vInit( void ) ;
00014
00015 void HBluetooth_vSendByte( u8_t A_u8Byte ) ;
00016
00017 u8_t HBluetooth_u8ReceiveByte( void ) ;
00018
00019 void HBluetooth_u8AsynchReceiveByte( void (*Fptr) (void) ) ;
00020
00021 void HBluetooth_vSendString( c8_t *A_ptrc8String ) ;
00022
00023 void HBluetooth_vReceiveString( c8_t A_c8YourString[] ) ;
00024
00025 u8_t HBluetooth_u8CompStrings( c8_t *String1, c8_t *String2 ) ;
00026
00027 void HBluetooth_vEnable( void ) ;
00028
00029 void HBluetooth_vDisable( void ) ;
00030
00031 void HBluetooth_vEnableAsynchReceive( void ) ;
00032
00033 void HBluetooth_vDisableAsynchReceive( void ) ;
00034
00035 u8_t HBluetooth_u8GetDataRegister( void ) ;
00036
00037
00038 #endif // _Bluetooth_interface_H

```

7.28 Bluetooth_private.h

```

00001 /* FILENAME: Bluetooth_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 01/05/2023
00005 */
00006 #ifndef _Bluetooth_private_H
00007 #define _Bluetooth_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Bluetooth_private_H

```

7.29 Bluetooth_program.c

```

00001 /* FILENAME: Bluetooth_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 01/05/2023
00005 */
00006
00007 /***** */
00008 /* Include headers */ */
00009 /***** */
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/RCC/MRCC_interface.h"
00017 #include "../../MCAL/NVIC/NVIC_interface.h"
00018 #include "../../MCAL/GPIO/GPIO_interface.h"
00019 #include "../../MCAL/UART/UART_interface.h"
00020
00021 #include "Bluetooth_interface.h"
00022 #include "Bluetooth_private.h"
00023 #include "Bluetooth_config.h"
00024
00025 /***** */
00026 /* Externed variables */ */
00027 /***** */
00028
00029 extern USART_InitType Bluetooth_Config ;
00030

```

```

00031 extern USART_ClockInitTypeDef      Bluetooth_CLK      ;
00032
00033 extern USART_MemoryMapType *      Bluetooth_UART_ID   ;
00034
00035 /****** Global variables *****/
00036 /****** Functions' implementations *****/
00037 /****** */
00038
00039
00040
00041 /****** */
00042 /****** Functions' implementations *****/
00043 /****** */
00044
00045 void HBluetooth_vInit( void )
00046 {
00047
00048     // Initialization of USART1:
00049     MUSART_vInit( &Bluetooth_Config, &Bluetooth_CLK, Bluetooth_UART_ID ) ;
00050
00051 }
00052
00053
00054
00055
00056 void HBluetooth_vSendByte( u8_t A_u8Byte )
00057 {
00058
00059     MUSART_vTransmitByte( Bluetooth_UART_ID, A_u8Byte ) ;
00060
00061 }
00062
00063
00064
00065
00066 u8_t HBluetooth_u8ReceiveByte( void )
00067 {
00068
00069     u8_t L_u8ReceivedByte = INITIAL_ZERO ;
00070
00071     L_u8ReceivedByte = MUSART_u8ReceiveByteSyncBlocking( Bluetooth_UART_ID ) ;
00072
00073     return L_u8ReceivedByte ;
00074
00075 }
00076
00077
00078
00079
00080 void HBluetooth_u8AsynchReceiveByte( void (*Fptr) (void) )
00081 {
00082
00083     MUSART1_vSetCallBack( Fptr ) ;
00084
00085 }
00086
00087
00088
00089
00090 void HBluetooth_vSendString( c8_t *A_ptrc8String )
00091 {
00092
00093     MUSART_vTransmitString( Bluetooth_UART_ID, A_ptrc8String ) ;
00094
00095 }
00096
00097
00098
00099
00100 void HBluetooth_vReceiveString( c8_t A_c8YourString[] )
00101 {
00102
00103     MUSART_vRecieveString( Bluetooth_UART_ID, A_c8YourString ) ;
00104
00105 }
00106
00107

```

```
00108 //*****
00109 //*****
00110 u8_t HBluetooth_u8CompStrings( c8_t *String1, c8_t *String2 )
00111 {
00112     u8_t L_u8Result = INITIAL_ZERO ;
00113     L_u8Result = MUSART_u8CompareString( String1, String2 ) ;
00114
00115     return L_u8Result ;
00116
00117 }
00118
00119 }
00120
00121 //*****
00122 //*****
00123 void HBluetooth_vEnable( void )
00124 {
00125
00126     MUSART_vEnable( Bluetooth_UART_ID ) ;
00127
00128 }
00129
00130
00131 //*****
00132 //*****
00133 void HBluetooth_vDisable( void )
00134 {
00135
00136
00137     MUSART_vDisable( Bluetooth_UART_ID ) ;
00138
00139 }
00140
00141 //*****
00142 //*****
00143 u8_t HBluetooth_u8GetDataRegister( void )
00144 {
00145
00146     u8_t volatile L_u8ReceivedByte = INITIAL_ZERO ;
00147
00148     L_u8ReceivedByte = MUSART_u8ReadDataRegister( USART1_REG ) ;
00149
00150     return L_u8ReceivedByte ;
00151
00152 }
00153
00154
00155 //*****
00156 //*****
00157 void HBluetooth_vEnableAsynchReceive( void )
00158 {
00159
00160
00161     // RXNEIE Pending Flag Enabled:
00162     MUSART_vRxIntSetStatus( Bluetooth_UART_ID, ENABLE ) ;
00163
00164     // USART1 IRQ Active Flag Enabled:
00165     MNVIC_vEnablePeripheral( USART1 ) ;
00166
00167 }
00168
00169 //*****
00170 //*****
00171 void HBluetooth_vDisableAsynchReceive( void )
00172 {
00173
00174
00175     // RXNEIE Pending Flag Disabled:
00176     MUSART_vRxIntSetStatus( Bluetooth_UART_ID, DISABLE ) ;
00177
00178     // USART1 IRQ Active Flag Disabled:
00179     MNVIC_vDisablePeripheral( USART1 ) ;
00180
00181 }
00182
```

```

00183
00184 /*****
00185 ****
00186 ****
00187 ****
00188 ****
00189 ****
00190 ****
00191 ****

```

7.30 BUZZER_config.h

```

00001 /* FILENAME: BUZZER_config.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00006 #ifndef _BUZZER_CONFIG_H
00007 #define _BUZZER_CONFIG_H
00008
00009 #include "../../MCAL/GPIO/GPIO_interface.h"
00010
00011 /*****
00012 * Configuration macros
00013 *****/
00014
00015 #define BUZZER_PORT GPIO_PORTA
00016
00017 #define BUZZER_PIN GPIOx_PIN1
00018
00019 #endif // _BUZZER_CONFIG_H

```

7.31 COTS/HAL/BUZZER/BUZZER_interface.h File Reference

This file contains the interface information for the Buzzer module.

Data Structures

- struct [BUZZER_BuzzerConfiguration](#)
Buzzer configuration structure for Buzzer initialization.

Functions

- void [HBUZZER_vInit \(BUZZER_BuzzerConfiguration *pBuzzer_Cfg\)](#)
This function is used to initialize the Buzzer.
- void [HBUZZER_vSoundOn \(BUZZER_BuzzerConfiguration *pBuzzer_Cfg\)](#)
This function is used to sound the Buzzer on.
- void [HBUZZER_vSoundOff \(BUZZER_BuzzerConfiguration *pBuzzer_Cfg\)](#)
This function is used to sound the Buzzer off.
- void [HBUZZER_vToggleSound \(BUZZER_BuzzerConfiguration *pBuzzer_Cfg\)](#)
This function is used to toggle the sound of the buzzer.

7.31.1 Detailed Description

This file contains the interface information for the Buzzer module.

Author

Mohamed Alaa

Version

1.0

Date

1/27/2022

Definition in file [BUZZER_interface.h](#).

7.31.2 Function Documentation

7.31.2.1 HBUZZER_vInit()

```
void HBUZZER_vInit (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to initialize the Buzzer.

Parameters

in	<code>pBuzzer_Cfg</code>	Pointer to Buzzer configuration structure to initialize the Buzzer with a certain port and pin
----	--------------------------	--

Definition at line 26 of file [BUZZER_program.c](#).

```
00027 {
00028     VAR(MGPIoX_ConfigType) Buzzer_Cfg =
00029     {
00030         .Port = pBuzzer_Cfg->u8Port,
00031         .Pin = pBuzzer_Cfg->u8Pin,
00032         .Mode = GPIOx_MODE_OUTPUT,
00033         .OutputType = GPIOx_PUSH_PULL,
00034         .OutputSpeed = GPIOx_LowSpeed,
00035         .InputType = GPIOx_NoPull};
00036
00037     MGPIoX_vInit(&Buzzer_Cfg);
00038 }
```

References [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSH_PULL](#), [MGPIoX_vInit\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), [BUZZER_BuzzerConfiguration::u8Port](#), and [VAR](#).

7.31.2.2 HBUZZER_vSoundOn()

```
void HBUZZER_vSoundOn (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to sound the Buzzer on.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to sound on the Buzzer
----	--------------------	--

Definition at line 43 of file [BUZZER_program.c](#).

```
00044 {
00045     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_HIGH);
00046 }
```

References [GPIOx_HIGH](#), [MGPIOx_vSetValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Po](#)

7.31.2.3 HBUZZER_vSoundOff()

```
void HBUZZER_vSoundOff (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to sound the Buzzer off.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to sound off the Buzzer
----	--------------------	---

Definition at line 51 of file [BUZZER_program.c](#).

```
00052 {
00053     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_LOW);
00054 }
```

References [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Po](#)

7.31.2.4 HBUZZER_vToggleSound()

```
void HBUZZER_vToggleSound (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to toggle the sound of the buzzer.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to toggle the sound of the Buzzer
----	--------------------	---

Definition at line 59 of file [BUZZER_program.c](#).

```

00060 {
00061     MGPIOx_vTogglePinValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin);
00062 }

```

References [MGPIOx_vTogglePinValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Port](#).

7.32 BUZZER_interface.h

[Go to the documentation of this file.](#)

```

00001 /* FILENAME: BUZZER_interface.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00014 #ifndef _BUZZER_INTERFACE_H
00015 #define _BUZZER_INTERFACE_H
00016
00017 /*****
00018 ****
00019
00025 typedef struct
00026 {
00030     u8_t u8Port;
00034     u8_t u8Pin;
00035 } BUZZER_BuzzerConfiguration;
00036
00037 /*****
00038 /*          Functions prototypes          */
00039 ****/
00040
00045 void HBUZZER_vInit(BUZZER_BuzzerConfiguration *pBuzzer_Cfg);
00046
00051 void HBUZZER_vSoundOn(BUZZER_BuzzerConfiguration *pBuzzer_Cfg);
00052
00057 void HBUZZER_vSoundOff(BUZZER_BuzzerConfiguration *pBuzzer_Cfg);
00058
00063 void HBUZZER_vToggleSound(BUZZER_BuzzerConfiguration *pBuzzer_Cfg);
00064
00065 #endif // _BUZZER_INTERFACE_H

```

7.33 BUZZER_private.h

```

00001 /* FILENAME: BUZZER_private.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00006 #ifndef _BUZZER_PRIVATE_H
00007 #define _BUZZER_PRIVATE_H
00008
00009 #endif // _BUZZER_PRIVATE_H

```

7.34 COTS/HAL/BUZZER/BUZZER_program.c File Reference

This file contains the logical operations of the buzzer module.

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../MCAL/GPIO/GPIO_interface.h"
#include "BUZZER_private.h"
#include "BUZZER_interface.h"
#include "BUZZER_config.h"

```

Functions

- void **HBUZZER_vInit** (BUZZER_BuzzerConfiguration *pBuzzer_Cfg)
This function is used to initialize the Buzzer.
- void **HBUZZER_vSoundOn** (BUZZER_BuzzerConfiguration *pBuzzer_Cfg)
This function is used to sound the Buzzer on.
- void **HBUZZER_vSoundOff** (BUZZER_BuzzerConfiguration *pBuzzer_Cfg)
This function is used to sound the Buzzer off.
- void **HBUZZER_vToggleSound** (BUZZER_BuzzerConfiguration *pBuzzer_Cfg)
This function is used to toggle the sound of the buzzer.

7.34.1 Detailed Description

This file contains the logical operations of the buzzer module.

Author

Mohamed Alaa

Version

1.0

Date

12/22/2022

Definition in file [BUZZER_program.c](#).

7.34.2 Function Documentation

7.34.2.1 HBUZZER_vInit()

```
void HBUZZER_vInit (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to initialize the Buzzer.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration structure to initialize the Buzzer with a certain port and pin
----	--------------------	--

Definition at line 26 of file [BUZZER_program.c](#).

```
00027 {
00028     VAR(MGPIOx_ConfigType) Buzzer_Cfg =
00029     {
```

```

00030     .Port = pBuzzer_Cfg->u8Port,
00031     .Pin = pBuzzer_Cfg->u8Pin,
00032     .Mode = GPIOx_MODE_OUTPUT,
00033     .OutputType = GPIOx_PUSH_PULL,
00034     .OutputSpeed = GPIOx_LowSpeed,
00035     .InputType = GPIOx_NoPull};
00036
00037     MGPIOx_vInit(&Buzzer_Cfg);
00038 }
```

References [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSH_PULL](#), [MGPIOx_vInit\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), [BUZZER_BuzzerConfiguration::u8Port](#), and [VAR](#).

7.34.2.2 HBUZZER_vSoundOn()

```
void HBUZZER_vSoundOn (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to sound the Buzzer on.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to sound on the Buzzer
----	--------------------	--

Definition at line 43 of file [BUZZER_program.c](#).

```

00044 {
00045     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_HIGH);
00046 }
```

References [GPIOx_HIGH](#), [MGPIOx_vSetValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Port](#).

7.34.2.3 HBUZZER_vSoundOff()

```
void HBUZZER_vSoundOff (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to sound the Buzzer off.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to sound off the Buzzer
----	--------------------	---

Definition at line 51 of file [BUZZER_program.c](#).

```

00052 {
00053     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_LOW);
00054 }
```

References [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Port](#).

7.34.2.4 HBUZZER_vToggleSound()

```
void HBUZZER_vToggleSound (
    BUZZER_BuzzerConfiguration * pBuzzer_Cfg )
```

This function is used to toggle the sound of the buzzer.

Parameters

in	<i>pBuzzer_Cfg</i>	Pointer to Buzzer configuration to toggle the sound of the Buzzer
----	--------------------	---

Definition at line 59 of file [BUZZER_program.c](#).

```
00060 {
00061     MGPIOx_vTogglePinValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin);
00062 }
```

References [MGPIOx_vTogglePinValue\(\)](#), [BUZZER_BuzzerConfiguration::u8Pin](#), and [BUZZER_BuzzerConfiguration::u8Port](#).

7.35 BUZZER_program.c

[Go to the documentation of this file.](#)

```
00001 /*****
00002  *          Include headers
00003  */
00004
00005 #include "../../../LIB/LSTD_TYPES.h"
00006 #include "../../../LIB/LSTD_COMPILER.h"
00007
00008 #include "../../../MCAL/GPIO/GPIO_interface.h"
00009
00010 #include "BUZZER_private.h"
00011 #include "BUZZER_interface.h"
00012 #include "BUZZER_config.h"
00013
00014 /*****
00015  *          Functions' implementations
00016  */
00017
00018 void HBUZZER_vInit(BUZZER_BuzzerConfiguration * pBuzzer_Cfg)
00019 {
00020     VAR(MGPIOx_ConfigType) Buzzer_Cfg =
00021     {
00022         .Port = pBuzzer_Cfg->u8Port,
00023         .Pin = pBuzzer_Cfg->u8Pin,
00024         .Mode = GPIOx_MODE_OUTPUT,
00025         .OutputType = GPIOx_PUSHPULL,
00026         .OutputSpeed = GPIOx_LowSpeed,
00027         .InputType = GPIOx_NoPull};
00028
00029     MGPIOx_vInit(&Buzzer_Cfg);
00030 }
00031
00032 /*****
00033  */
00034 void HBUZZER_vSoundOn(BUZZER_BuzzerConfiguration * pBuzzer_Cfg)
00035 {
00036     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_HIGH);
00037 }
00038
00039 /*****
00040  */
00041 /*****
00042  */
00043 void HBUZZER_vSoundOff(BUZZER_BuzzerConfiguration * pBuzzer_Cfg)
00044 {
00045     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_LOW);
00046 }
00047
00048 /*****
00049  */
00050 /*****
00051  */
00052 void HBUZZER_vToggleSound(BUZZER_BuzzerConfiguration * pBuzzer_Cfg)
00053 {
00054     MGPIOx_vSetValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin, GPIOx_LOW);
00055 }
```

```

00055
00056    /*****
00057    ****
00058
00059 void HBUZZER_vToggleSound(BUZZER_BuzzerConfiguration * pBuzzer_Cfg)
00060 {
00061     MGPIOx_vTogglePinValue(pBuzzer_Cfg->u8Port, pBuzzer_Cfg->u8Pin);
00062 }
00063
00064
00065    ****
00066    ****

```

7.36 Car_Movement_config.h

```

00001 /* FILENAME: Car_Movement_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 03/14/2023
00005 */
00006 #ifndef _Car_Movement_config_H
00007 #define _Car_Movement_config_H
00008
00009
00010 #define MOTOR1_PORT GPIO_PORTA
00011 #define MOTOR2_PORT GPIO_PORTA
00012
00013 #define IN1_PIN      GPIOx_PIN0
00014 #define IN2_PIN      GPIOx_PIN1
00015
00016 #define IN3_PIN      GPIOx_PIN2
00017 #define IN4_PIN      GPIOx_PIN3
00018
00019
00020 // Initialize motor 1
00021 VAR(DCM_MotorConfiguration) Motor1 =
00022 {
00023     .u8Port      = MOTOR1_PORT  ,
00024     .u8Pin1     = IN1_PIN      ,
00025     .u8Pin2     = IN2_PIN      ,
00026     .u8SpeedPin = TIM1_CH4   ,
00027     .u32SpeedRatio = SPEED_0_PERCENT
00028 };
00029
00030 // Initialize motor 3
00031 VAR(DCM_MotorConfiguration) Motor3 =
00032 {
00033     .u8Port      = MOTOR2_PORT  ,
00034     .u8Pin1     = IN3_PIN      ,
00035     .u8Pin2     = IN4_PIN      ,
00036     .u8SpeedPin = TIM1_CH4   ,
00037     .u32SpeedRatio = SPEED_0_PERCENT
00038 };
00039
00040
00041 #endif // _Car_Movement_config_H

```

7.37 Car_Movement_interface.h

```

00001 /* FILENAME: Car_Movement_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 03/14/2023
00005 */
00006 #ifndef _Car_Movement_interface_H
00007 #define _Car_Movement_interface_H
00008
00009 /*****
00010 *          Functions prototypes
00011 *****/
00012
00013 void HCarMove_vInit(void);
00014
00015 void HCarMove_vSpeedRatio(u16_t A_u16SpeedRatio);
00016
00017 void HCarMove_vForward(void);

```

```

00018
00019 void HCarMove_vBackward(void);
00020
00021 void HCarMove_vRight(void);
00022
00023 void HCarMove_vLeft(void);
00024
00025 void HCarMove_vStop(void);
00026
00027 u32_t HCarMove_u32GetCarSpeed(void);
00028
00029 #endif // _Car_Movement_interface_H

```

7.38 Car_Movement_private.h

```

00001 /* FILENAME: Car_Movement_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 03/14/2023
00005 */
00006 #ifndef _Car_Movement_private_H
00007 #define _Car_Movement_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _Car_Movement_private_H

```

7.39 Car_Movement_program.c

```

00001 /* FILENAME: Car_Movement_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 03/14/2023
00005 */
00006
00007 /***** Include headers ****/
00008 /*           Include headers           */
00009 /***** Include headers ****/
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/GPIO/GPIO_interface.h"
00017 #include "../../MCAL/Systick/SysTick_interface.h"
00018 #include "../../MCAL/TIM1/TIM1_interface.h"
00019
00020 #include "../DCMOTOR/DCM_interface.h"
00021
00022 #include "Car_Movement_interface.h"
00023 #include "Car_Movement_private.h"
00024 #include "Car_Movement_config.h"
00025
00026 /***** Functions implementations ****/
00027 /*           Functions implementations           */
00028 /***** Functions implementations ****/
00029
00030 void HCarMove_vInit( void )
00031 {
00032
00033     HDCM_vInitMotor( &Motor1 ) ;
00034     HDCM_vInitMotor( &Motor3 ) ;
00035
00036 }
00037
00038
00039 /***** ****/
00040
00041 void HCarMove_vSpeedRatio( u16_t A_u16SpeedRatio )
00042 {
00043
00044     HDCM_vMotorSpeedCntrl( &Motor1, A_u16SpeedRatio ) ;
00045     HDCM_vMotorSpeedCntrl( &Motor3, A_u16SpeedRatio ) ;

```

```

00046
00047 }
00048
00049
00050 /******
00051 *****/
00052 void HCarMove_vForward( void )
00053 {
00054
00055     /* IN1 = 1, IN2 = 0, IN3 = 1, IN4 = 0 */
00056
00057     HDCM_vMoveForward( &Motor1 ) ;
00058     HDCM_vMoveForward( &Motor3 ) ;
00059
00060 }
00061
00062
00063 /******
00064 *****/
00065 void HCarMove_vBackward( void )
00066 {
00067
00068     /* IN1 = 0, IN2 = 1, IN3 = 0, IN4 = 1 */
00069
00070     HDCM_vMoveBackward( &Motor1 ) ;
00071     HDCM_vMoveBackward( &Motor3 ) ;
00072
00073 }
00074
00075
00076 /******
00077 *****/
00078 void HCarMove_vRight( void )
00079 {
00080
00081     /* IN1 = 1, IN2 = 0, IN3 = 0, IN4 = 0 */
00082
00083     HDCM_vMoveForward( &Motor1 ) ;
00084     HDCM_vStopMotor    ( &Motor3 ) ;
00085
00086 }
00087
00088
00089 /******
00090 *****/
00091 void HCarMove_vLeft( void )
00092 {
00093
00094     /* IN1 = 0, IN2 = 0, IN3 = 1, IN4 = 0 */
00095
00096     HDCM_vStopMotor    ( &Motor1 ) ;
00097     HDCM_vMoveForward( &Motor3 ) ;
00098
00099 }
00100
00101
00102 /******
00103 *****/
00104 void HCarMove_vStop( void )
00105 {
00106
00107     /* IN1 = 0, IN2 = 0, IN3 = 0, IN4 = 0 */
00108
00109     HDCM_vStopMotor( &Motor1 ) ;
00110     HDCM_vStopMotor( &Motor3 ) ;
00111
00112 }
00113
00114
00115 /******
00116 *****/
00117 u32_t HCarMove_u32GetCarSpeed( void )
00118 {
00119
00120     u32_t L_u32SpeedValue = INITIAL_ZERO ;

```

```
00121     L_u32SpeedValue = HDCM\_vGetSpeedValue\( &Motor1 \) ;
00122 
00123     return L_u32SpeedValue ;
00124 }
00125
00126 }
00127
00128 *****
00129 *****
00130
00131
00132
```

7.40 COTS/HAL/DCMOTOR/DCM_config.h File Reference

This file contains the configuration information for the DC Motor module.

Macros

- `#define MAX_SPEED (10000)`

7.40.1 Detailed Description

This file contains the configuration information for the DC Motor module.

Author

Mohamed Alaa

Version

1.0

Date

8/12/2022

Definition in file [DCM_config.h](#).

7.40.2 Macro Definition Documentation

7.40.2.1 MAX_SPEED

```
#define MAX_SPEED (10000)
```

Maximum speed of the DC motor

Definition at line 17 of file [DCM_config.h](#).

7.41 DCM_config.h

Go to the documentation of this file.

```
00001 /* Header file guard */  
00002 #ifndef _DCM_config_H  
00003 #define _DCM_config_H  
00004  
00017 #define MAX_SPEED (10000)  
00018  
00019 #endif // _DCM_config_H
```

7.42 COTS/HAL/DCMOTOR/DCM_interface.h File Reference

This file contains the interfacing information for the DC Motor module.

```
#include "DCM_private.h"
```

Data Structures

- struct [DCM_MotorConfiguration](#)
DC Motor configuration structure for motor initialization.

Macros

- [#define FORWARD \(CW\)](#)
Motor forward direction.
- [#define BACKWARD \(CCW\)](#)
Motor backward direction.
- [#define SPEED_0_PERCENT 0](#)
Motor speed 0%.
- [#define SPEED_10_PERCENT 1000U](#)
Motor speed 10%.
- [#define SPEED_20_PERCENT 2000U](#)
Motor speed 20%.
- [#define SPEED_30_PERCENT 3000U](#)
Motor speed 30%.
- [#define SPEED_40_PERCENT 4000U](#)
Motor speed 40%.
- [#define SPEED_50_PERCENT 5000U](#)
Motor speed 50%.
- [#define SPEED_60_PERCENT 6000U](#)
Motor speed 60%.
- [#define SPEED_70_PERCENT 7000U](#)
Motor speed 70%.
- [#define SPEED_80_PERCENT 8000U](#)
Motor speed 80%.
- [#define SPEED_90_PERCENT 9000U](#)
Motor speed 90%.
- [#define SPEED_100_PERCENT 10000U](#)
Motor speed 100%.

Functions

- void [HDCM_vInitMotor \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Initialize a motor Initialize a motor with a certain GPIO Port & Pin based on the passed motor configuration.
- void [HDCM_vMoveForward \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Move a certain motor forward.
- void [HDCM_vMoveBackward \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Move a certain motor backward.
- void [HDCM_vStopMotor \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Stop a certain motor's movement completely.
- void [HDCM_vMotorSpeedCntrl \(DCM_MotorConfiguration *pMotorConfiguration, u16_t A_u16SpeedValue\)](#)
Set a certain motor's speed.
- [u32_t HDCM_vGetSpeedValue \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Get a certain motor's speed.

7.42.1 Detailed Description

This file contains the interfacing information for the DC Motor module.

Author

Mohamed Alaa

Version

1.0

Date

8/12/2022

Definition in file [DCM_interface.h](#).

7.42.2 Function Documentation

7.42.2.1 HDCM_vInitMotor()

```
void HDCM_vInitMotor (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Initialize a motor Initialize a motor with a certain GPIO Port & Pin based on the passed motor configuration.

Parameters

in	<i>pMotorConfiguration</i>	The motor configuration to be initialized
----	----------------------------	---

See also

[DCM_MotorConfiguration](#)

Definition at line 32 of file [DCM_program.c](#).

```

00033 {
00034
00035     VAR(MGPIOx_ConfigType) DCMotorPin1Init =
00036     {
00037         .Port = pMotorConfiguration->u8Port,
00038         .Pin = pMotorConfiguration->u8Pin1,
00039         .Mode = GPIOx_MODE_OUTPUT,
00040         .OutputType = GPIOx_PUSH_PULL,
00041         .OutputSpeed = GPIOx_LowSpeed,
00042         .InputType = GPIOx_NoPull};
00043
00044     VAR(MGPIOx_ConfigType) DCMotorPin2Init =
00045     {
00046         .Port = pMotorConfiguration->u8Port,
00047         .Pin = pMotorConfiguration->u8Pin2,
00048         .Mode = GPIOx_MODE_OUTPUT,
00049         .OutputType = GPIOx_PUSH_PULL,
00050         .OutputSpeed = GPIOx_LowSpeed,
00051         .InputType = GPIOx_NoPull};
00052
00053     MGPIOx_vInit(&DCMotorPin1Init);
00054     MGPIOx_vInit(&DCMotorPin2Init);
00055
00056     // Give an initial direction to the DC Motor:
00057     switch (pMotorConfiguration->u8Direction)
00058     {
00059
00060         case FORWARD:
00061             HDCM_vMoveForward(pMotorConfiguration);
00062             break;
00063
00064         case BACKWARD:
00065             HDCM_vMoveBackward(pMotorConfiguration);
00066             break;
00067
00068         /* Default case: stop the motor */
00069         default:
00070             HDCM_vStopMotor(pMotorConfiguration);
00071             break;
00072     }
00073
00074     // Give an initial speed to the DC Motor:
00075     HDCM_vMotorSpeedCntrl(pMotorConfiguration, pMotorConfiguration->u32SpeedRatio);
00076 }
```

References [BACKWARD](#), [FORWARD](#), [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSH_PULL](#), [HDCM_vMotorSpeedCntrl\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), [HDCM_vStopMotor\(\)](#), [MGPIOx_vInit\(\)](#), [DCM_MotorConfiguration::u32SpeedRatio](#), [DCM_MotorConfiguration::u8Direction](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), [DCM_MotorConfiguration::u8Port](#), and [VAR](#).

7.42.2.2 HDCM_vMoveForward()

```
void HDCM_vMoveForward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Move a certain motor forward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move forward
----	----------------------------	---

Definition at line 81 of file [DCM_program.c](#).

```
00082 {
```

```
00083     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_HIGH);
00084     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00085 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.42.2.3 HDCM_vMoveBackward()

```
void HDCM_vMoveBackward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Move a certain motor backward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move backward
----	----------------------------	--

Definition at line 90 of file [DCM_program.c](#).

```
00091 {
00092     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00093     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_HIGH);
00094 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.42.2.4 HDCM_vStopMotor()

```
void HDCM_vStopMotor (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Stop a certain motor's movement completely.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to stop
----	----------------------------	-----------------------------------

Definition at line 99 of file [DCM_program.c](#).

```
00100 {
00101     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00102     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00103 }
```

References [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.42.2.5 HDCM_vMotorSpeedCntrl()

```
void HDCM_vMotorSpeedCntrl (
    DCM_MotorConfiguration * pMotorConfiguration,
    u16_t A_u16SpeedValue )
```

Set a certain motor's speed.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to set its speed
in	<i>A_u16SpeedValue</i>	The speed value to set the motor to

Definition at line 108 of file [DCM_program.c](#).

```
00109 {
00110
00111     VAR(HULTSNC_ConfigType) TRIG =
00112     {
00113         .u8Port = GPIO_PORTB,
00114         .u8Pin = GPIOx_PIN8};
00115
00116     MTIM1_vGeneratePWM(pMotorConfiguration->u8SpeedPin, PWM1, CENTER1,
00117                         PSC_VALUE, MAX_SPEED, CR_VALUE);
00118
00119     HULTSNC_vInit(&TRIG);
00120
00121     switch (pMotorConfiguration->u8SpeedPin)
00122     {
00123
00124         case TIM1_CH1:
00125             MTIM1_vSetCompareReg1Value(A_u16SpeedValue);
00126             break;
00127
00128         case TIM1_CH2:
00129             MTIM1_vSetCompareReg2Value(A_u16SpeedValue);
00130             break;
00131
00132         case TIM1_CH3:
00133             MTIM1_vSetCompareReg3Value(A_u16SpeedValue);
00134             break;
00135
00136         case TIM1_CH4:
00137             MTIM1_vSetCompareReg4Value(A_u16SpeedValue);
00138             break;
00139
00140     default:
00141         break;
00142     }
00143 }
```

References [GPIO_PORTB](#), [GPIOx_PIN8](#), [HULTSNC_vInit\(\)](#), [MAX_SPEED](#), [DCM_MotorConfiguration::u8SpeedPin](#), and [VAR](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.42.2.6 HDCM_vGetSpeedValue()

```
u32_t HDCM_vGetSpeedValue (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Get a certain motor's speed.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to get its speed
----	----------------------------	--

Returns

The speed value of the motor

Definition at line 148 of file [DCM_program.c](#).

```
00149 {
00150
00151     u32_t L_u32SpeedValue = INITIAL_ZERO;
00152
00153     switch (pMotorConfiguration->u8SpeedPin)
00154     {
00155
00156         case TIM1_CH1:
00157             L_u32SpeedValue = MTIM1_u16GetCaptureReg1Value();
00158             break;
00159
00160         case TIM1_CH2:
00161             L_u32SpeedValue = MTIM1_u16GetCaptureReg2Value();
00162             break;
00163
00164         case TIM1_CH3:
00165             L_u32SpeedValue = MTIM1_u16GetCaptureReg3Value();
00166             break;
00167
00168         case TIM1_CH4:
00169             L_u32SpeedValue = MTIM1_u16GetCaptureReg4Value();
00170             break;
00171
00172         default: /* Do nothing */
00173             break;
00174     }
00175
00176     return L_u32SpeedValue;
00177 }
```

References [INITIAL_ZERO](#), and [DCM_MotorConfiguration::u8SpeedPin](#).

7.43 DCM_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _DCM_interface_H
00011 #define _DCM_interface_H
00012
00013 /*****
00014 *           Include headers
00015 *****/
00016 #include "DCM_private.h"
00017
00018 /*****
00019 *****/
00020
00028 typedef struct
00029 {
00033     u8_t u8Port;
00037     u8_t u8Pin1;
00041     u8_t u8Pin2;
00045     u8_t u8Direction;
00049     u8_t u8SpeedPin;
00053     u32_t u32SpeedRatio;
00054 } DCM_MotorConfiguration;
00055
00056 /*****
00057 *           Functions prototypes
00058 *****/
00059
00066 void HDCM_vInitMotor(DCM_MotorConfiguration *pMotorConfiguration);
00067
```

```
00072 void HDCM_vMoveForward(DCM_MotorConfiguration *pMotorConfiguration);
00073
00078 void HDCM_vMoveBackward(DCM_MotorConfiguration *pMotorConfiguration);
00079
00084 void HDCM_vStopMotor(DCM_MotorConfiguration *pMotorConfiguration);
00085
00091 void HDCM_vMotorSpeedCntrl(DCM_MotorConfiguration *pMotorConfiguration, ul6_t A_ul6SpeedValue);
00092
00098 u32_t HDCM_vGetSpeedValue(DCM_MotorConfiguration *pMotorConfiguration);
00099
00100 /***** Interfacing macros *****/
00101 /*                                         */
00102 /*****                                         *****/
00103
00115 #define FORWARD (CW)
00116
00122 #define BACKWARD (CCW)
00123
00136 #define SPEED_0_PERCENT 0
00137
00142 #define SPEED_10_PERCENT 1000U
00143
00148 #define SPEED_20_PERCENT 2000U
00149
00154 #define SPEED_30_PERCENT 3000U
00155
00160 #define SPEED_40_PERCENT 4000U
00161
00166 #define SPEED_50_PERCENT 5000U
00167
00172 #define SPEED_60_PERCENT 6000U
00173
00178 #define SPEED_70_PERCENT 7000U
00179
00184 #define SPEED_80_PERCENT 8000U
00185
00190 #define SPEED_90_PERCENT 9000U
00191
00196 #define SPEED_100_PERCENT 10000U
00197
00200 #endif // _DCM_interface_H
```

7.44 COTS/HAL/DCMOTOR/DCM_private.h File Reference

This file contains the private information for the DC Motor module.

Macros

- #define **CW** (1)
Motor clockwise rotation direction.
- #define **CCW** (2)
Motor counter-clockwise rotation direction.

7.44.1 Detailed Description

This file contains the private information for the DC Motor module.

Author

Mohamed Alaa

Version

1.0

Date

8/12/2022

Definition in file [DCM_private.h](#).

7.45 DCM_private.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _DCM_private_H
00011 #define _DCM_private_H
00012
00024 #define CW (1)
00025
00031 #define CCW (2)
00034 #endif // _DCM_private_H
```

7.46 COTS/HAL/DCMOTOR/DCM_program.c File Reference

This file contains the source code of the interfacing information for the DC Motor module.

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../MCAL/GPIO/GPIO_interface.h"
#include "../../MCAL/SysTick/SysTick_interface.h"
#include "../../MCAL/TIM1/TIM1_interface.h"
#include "../../HAL/UltraSonic/UltraSonic_interface.h"
#include "DCM_interface.h"
#include "DCM_private.h"
#include "DCM_config.h"
```

Functions

- void [HDCM_vInitMotor \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Initialize a motor Initialize a motor with a certain GPIO Port & Pin based on the passed motor configuration.
- void [HDCM_vMoveForward \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Move a certain motor forward.
- void [HDCM_vMoveBackward \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Move a certain motor backward.
- void [HDCM_vStopMotor \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Stop a certain motor's movement completely.
- void [HDCM_vMotorSpeedCntrl \(DCM_MotorConfiguration *pMotorConfiguration, u16_t A_u16SpeedValue\)](#)
Set a certain motor's speed.
- [u32_t HDCM_vGetSpeedValue \(DCM_MotorConfiguration *pMotorConfiguration\)](#)
Get a certain motor's speed.

7.46.1 Detailed Description

This file contains the source code of the interfacing information for the DC Motor module.

Author

Mohamed Alaa

Version

1.0

Date

8/12/2022

Definition in file [DCM_program.c](#).**7.46.2 Function Documentation****7.46.2.1 HDCM_vInitMotor()**

```
void HDCM_vInitMotor (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Initialize a motor Initialize a motor with a certain GPIO Port & Pin based on the passed motor configuration.

Parameters

in	<i>pMotorConfiguration</i>	The motor configuration to be initialized
----	----------------------------	---

See also

[DCM_MotorConfiguration](#)

Definition at line 32 of file [DCM_program.c](#).

```
00033 {
00034
00035     VAR(MGPIOx_ConfigType) DCMotorPin1Init =
00036     {
00037         .Port = pMotorConfiguration->u8Port,
00038         .Pin = pMotorConfiguration->u8Pin1,
00039         .Mode = GPIOx_MODE_OUTPUT,
00040         .OutputType = GPIOx_PUSH_PULL,
00041         .OutputSpeed = GPIOx_LowSpeed,
00042         .InputType = GPIOx_NoPull};
00043
00044     VAR(MGPIOx_ConfigType) DCMotorPin2Init =
00045     {
00046         .Port = pMotorConfiguration->u8Port,
00047         .Pin = pMotorConfiguration->u8Pin2,
00048         .Mode = GPIOx_MODE_OUTPUT,
00049         .OutputType = GPIOx_PUSH_PULL,
00050         .OutputSpeed = GPIOx_LowSpeed,
00051         .InputType = GPIOx_NoPull};
00052
00053     MGPIOx_vInit(&DCMotorPin1Init);
00054     MGPIOx_vInit(&DCMotorPin2Init);
00055
00056     // Give an initial direction to the DC Motor:
00057     switch (pMotorConfiguration->u8Direction)
00058     {
00059
00060         case FORWARD:
00061             HDCM_vMoveForward(pMotorConfiguration);
00062             break;
00063
00064         case BACKWARD:
```

```

00065     HDCM_vMoveBackward(pMotorConfiguration);
00066     break;
00067
00068     /* Default case: stop the motor */
00069     default:
00070         HDCM_vStopMotor(pMotorConfiguration);
00071         break;
00072     }
00073
00074     // Give an initial speed to the DC Motor:
00075     HDCM_vMotorSpeedCntrl(pMotorConfiguration, pMotorConfiguration->u32SpeedRatio);
00076 }
```

References [BACKWARD](#), [FORWARD](#), [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSHPULL](#), [HDCM_vMotorSpeedCntrl\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), [HDCM_vStopMotor\(\)](#), [MGPIOx_vInit\(\)](#), [DCM_MotorConfiguration::u32SpeedRatio](#), [DCM_MotorConfiguration::u8Direction](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), [DCM_MotorConfiguration::u8Port](#), and [VAR](#).

7.46.2.2 [HDCM_vMoveForward\(\)](#)

```
void HDCM_vMoveForward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Move a certain motor forward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move forward
----	----------------------------	---

Definition at line 81 of file [DCM_program.c](#).

```

00082 {
00083     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_HIGH);
00084     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00085 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.46.2.3 [HDCM_vMoveBackward\(\)](#)

```
void HDCM_vMoveBackward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Move a certain motor backward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move backward
----	----------------------------	--

Definition at line 90 of file [DCM_program.c](#).

```
00091 {
```

```

00092     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00093     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_HIGH);
00094 }

```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.46.2.4 HDCM_vStopMotor()

```

void HDCM_vStopMotor (
    DCM_MotorConfiguration * pMotorConfiguration )

```

Stop a certain motor's movement completely.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to stop
----	----------------------------	-----------------------------------

Definition at line 99 of file [DCM_program.c](#).

```

00100 {
00101     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00102     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00103 }

```

References [GPIOx_LOW](#), [MGPIOx_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.46.2.5 HDCM_vMotorSpeedCntrl()

```

void HDCM_vMotorSpeedCntrl (
    DCM_MotorConfiguration * pMotorConfiguration,
    u16_t A_u16SpeedValue )

```

Set a certain motor's speed.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to set its speed
in	<i>A_u16SpeedValue</i>	The speed value to set the motor to

Definition at line 108 of file [DCM_program.c](#).

```

00109 {
00110
00111     VAR(HULTSNC_ConfigType) TRIG =
00112         {
00113             .u8Port = GPIO_PORTB,
00114             .u8Pin = GPIOx_PIN8};

```

```

00115     MTIM1_vGeneratePWM(pMotorConfiguration->u8SpeedPin, PWM1, CENTER1,
00116                           PSC_VALUE, MAX_SPEED, CR_VALUE);
00117
00118     HULTSNC_vInit(&TRIG);
00119
00120     switch (pMotorConfiguration->u8SpeedPin)
00121     {
00122
00123         case TIM1_CH1:
00124             MTIM1_vSetCompareReg1Value(A_u16SpeedValue);
00125             break;
00126
00127         case TIM1_CH2:
00128             MTIM1_vSetCompareReg2Value(A_u16SpeedValue);
00129             break;
00130
00131         case TIM1_CH3:
00132             MTIM1_vSetCompareReg3Value(A_u16SpeedValue);
00133             break;
00134
00135         case TIM1_CH4:
00136             MTIM1_vSetCompareReg4Value(A_u16SpeedValue);
00137             break;
00138
00139     default:
00140         break;
00141     }
00142 }
00143 }
```

References [GPIO_PORTB](#), [GPIOx_PIN8](#), [HULTSNC_vInit\(\)](#), [MAX_SPEED](#), [DCM_MotorConfiguration::u8SpeedPin](#), and [VAR](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.46.2.6 HDCM_vGetSpeedValue()

```

u32_t HDCM_vGetSpeedValue (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Get a certain motor's speed.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to get its speed
----	----------------------------	--

Returns

The speed value of the motor

Definition at line 148 of file [DCM_program.c](#).

```

00149 {
00150
00151     u32_t L_u32SpeedValue = INITIAL_ZERO;
00152
00153     switch (pMotorConfiguration->u8SpeedPin)
00154     {
00155
00156         case TIM1_CH1:
00157             L_u32SpeedValue = MTIM1_u16GetCaptureReg1Value();
00158             break;
00159
00160         case TIM1_CH2:
00161             L_u32SpeedValue = MTIM1_u16GetCaptureReg2Value();
00162             break;
00163
00164         case TIM1_CH3:
```

```

00165     L_u32SpeedValue = MTIM1_u16GetCaptureReg3Value();
00166     break;
00167
00168 case TIM1_CH4:
00169     L_u32SpeedValue = MTIM1_u16GetCaptureReg4Value();
00170     break;
00171
00172 default: /* Do nothing */
00173     break;
00174 }
00175
00176 return L_u32SpeedValue;
00177 }
```

References [INITIAL_ZERO](#), and [DCM_MotorConfiguration::u8SpeedPin](#).

7.47 DCM_program.c

[Go to the documentation of this file.](#)

```

00001
00009 /***** Include headers *****/
00010 /*
00011  *          Include headers
00012 */
00013 #include "../../LIB/LSTD_TYPES.h"
00014 #include "../../LIB/LSTD_COMPILER.h"
00015 #include "../../LIB/LSTD_VALUES.h"
00016 #include "../../LIB/LSTD_BITMATH.h"
00017
00018 #include "../../MCAL/GPIO/GPIO_interface.h"
00019 #include "../../MCAL/SysTick/SysTick_interface.h"
00020 #include "../../MCAL/TIM1/TIM1_interface.h"
00021
00022 #include "../../HAL/UltraSonic/UltraSonic_interface.h"
00023
00024 #include "DCM_interface.h"
00025 #include "DCM_private.h"
00026 #include "DCM_config.h"
00027
00028 /***** Functions implementations *****/
00029 /*          Functions implementations
00030 */
00031
00032 void HDCM_vInitMotor(DCM_MotorConfiguration *pMotorConfiguration)
00033 {
00034
00035     VAR(MGPIox_ConfigType) DCMotorPin1Init =
00036     {
00037         .Port = pMotorConfiguration->u8Port,
00038         .Pin = pMotorConfiguration->u8Pin1,
00039         .Mode = GPIOx_MODE_OUTPUT,
00040         .OutputType = GPIOx_PUSH_PULL,
00041         .OutputSpeed = GPIOx_LowSpeed,
00042         .InputType = GPIOx_NoPull};
00043
00044     VAR(MGPIox_ConfigType) DCMotorPin2Init =
00045     {
00046         .Port = pMotorConfiguration->u8Port,
00047         .Pin = pMotorConfiguration->u8Pin2,
00048         .Mode = GPIOx_MODE_OUTPUT,
00049         .OutputType = GPIOx_PUSH_PULL,
00050         .OutputSpeed = GPIOx_LowSpeed,
00051         .InputType = GPIOx_NoPull};
00052
00053     MGPIox_vInit(&DCMotorPin1Init);
00054     MGPIox_vInit(&DCMotorPin2Init);
00055
00056 // Give an initial direction to the DC Motor:
00057 switch (pMotorConfiguration->u8Direction)
00058 {
00059
00060     case FORWARD:
00061         HDCM_vMoveForward(pMotorConfiguration);
00062         break;
00063
00064     case BACKWARD:
00065         HDCM_vMoveBackward(pMotorConfiguration);
00066         break;
00067
00068     /* Default case: stop the motor */
00069 }
```

```

00069     default:
00070         HDCM_vStopMotor(pMotorConfiguration);
00071         break;
00072     }
00073
00074     // Give an initial speed to the DC Motor:
00075     HDCM_vMotorSpeedCntrl(pMotorConfiguration, pMotorConfiguration->u32SpeedRatio);
00076 }
00077
00078 /*****
00079 ****/
00080
00081 void HDCM_vMoveForward(DCM_MotorConfiguration *pMotorConfiguration)
00082 {
00083     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_HIGH);
00084     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00085 }
00086
00087 /*****
00088 ****/
00089
00090 void HDCM_vMoveBackward(DCM_MotorConfiguration *pMotorConfiguration)
00091 {
00092     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00093     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_HIGH);
00094 }
00095
00096 /*****
00097 ****/
00098
00099 void HDCM_vStopMotor(DCM_MotorConfiguration *pMotorConfiguration)
00100 {
00101     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00102     MGPIOx_vSetPinValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00103 }
00104
00105 /*****
00106 ****/
00107
00108 void HDCM_vMotorSpeedCntrl(DCM_MotorConfiguration *pMotorConfiguration, u16_t A_u16SpeedValue)
00109 {
00110
00111     VAR(HULTSNC_ConfigType) TRIG =
00112     {
00113         .u8Port = GPIO_PORTB,
00114         .u8Pin = GPIOx_PIN8};
00115
00116     MTIM1_vGeneratePWM(pMotorConfiguration->u8SpeedPin, PWM1, CENTER1,
00117                         PSC_VALUE, MAX_SPEED, CR_VALUE);
00118
00119     HULTSNC_vInit(&TRIG);
00120
00121     switch (pMotorConfiguration->u8SpeedPin)
00122     {
00123
00124     case TIM1_CH1:
00125         MTIM1_vSetCompareReg1Value(A_u16SpeedValue);
00126         break;
00127
00128     case TIM1_CH2:
00129         MTIM1_vSetCompareReg2Value(A_u16SpeedValue);
00130         break;
00131
00132     case TIM1_CH3:
00133         MTIM1_vSetCompareReg3Value(A_u16SpeedValue);
00134         break;
00135
00136     case TIM1_CH4:
00137         MTIM1_vSetCompareReg4Value(A_u16SpeedValue);
00138         break;
00139
00140     default:
00141         break;
00142     }
00143 }
00144
00145 /*****
00146 ****/

```

```

00147 /*****
00148 u32_t HDCM_vGetSpeedValue(DCM_MotorConfiguration *pMotorConfiguration)
00149 {
00150
00151     u32_t L_u32SpeedValue = INITIAL_ZERO;
00152
00153     switch (pMotorConfiguration->u8SpeedPin)
00154     {
00155
00156         case TIM1_CH1:
00157             L_u32SpeedValue = MTIM1_u16GetCaptureReg1Value();
00158             break;
00159
00160         case TIM1_CH2:
00161             L_u32SpeedValue = MTIM1_u16GetCaptureReg2Value();
00162             break;
00163
00164         case TIM1_CH3:
00165             L_u32SpeedValue = MTIM1_u16GetCaptureReg3Value();
00166             break;
00167
00168         case TIM1_CH4:
00169             L_u32SpeedValue = MTIM1_u16GetCaptureReg4Value();
00170             break;
00171
00172     default: /* Do nothing */
00173         break;
00174     }
00175
00176     return L_u32SpeedValue;
00177 }
00178
00179 /*****
00180 *****/

```

7.48 LCD_config.h

```

00001 /*
00002 * LCD_config.h
00003 *
00004 * Created on: Sep 11, 2021
00005 * Author: Ali El Bana
00006 */
00007
00008 #ifndef LCD_LCD_CONFIG_H_
00009 #define LCD_LCD_CONFIG_H_
00010
00011 #include "LCD_private.h"
00012
00013
00014
00015 #define HLCD_CTRL_PORT GPIO_PORTB
00016
00017 #define HLCD_RS_PIN      GPIOx_PIN12
00018 #define HLCD_RW_PIN      GPIOx_PIN13
00019 #define HLCD_EN_PIN      GPIOx_PIN14
00020
00021
00022 #define HLCD_DATA_PORT  GPIO_PORTA
00023
00024 #define D0_PIN GPIOx_PIN0
00025 #define D1_PIN GPIOx_PIN1
00026 #define D2_PIN GPIOx_PIN2
00027 #define D3_PIN GPIOx_PIN3
00028 #define D4_PIN GPIOx_PIN4
00029 #define D5_PIN GPIOx_PIN5
00030 #define D6_PIN GPIOx_PIN6
00031 #define D7_PIN GPIOx_PIN7
00032
00033
00034 #define HLCD_FuctionSet_Cmd          FunctionSetCmd
00035
00036 #define HLCD_DisponOffCTRL_Cmd       DispOnOffCTRLCmd
00037
00038 #define HLCD_DispoFF_Cmd            DispOffCmd
00039
00040 #define HLCD_DispcursorWithBlinking_Cmd DispCursorWithBlinkingCmd
00041
00042 #define HLCD_SetcursorBlinkingOFF_Cmd SetCursorBlinkingOFFCmd
00043

```

```

00044 #define HLCD_Dispclear_Cmd           DispClearCmd
00045
00046 #define HLCD_EntryModeSet_Cmd       EntryModeSet_Cmd
00047
00048 #define HLCD_EntryModeSet_ShiftLeftOn_Cmd EntryModeSet_ShiftLeftOn_Cmd
00049
00050
00051 #define HLCD_CharactersNums      16
00052
00053
00054 #define FIRST_ROW_IDX            0
00055
00056 #define FIRST_ROW_START          0x00
00057
00058 #define SEC_ROW_START           0x40
00059
00060 #define SET_DDRAM_AC_MASK        0x80
00061
00062 #define SET_CGRAM_AC_MASK         0x40
00063
00064 #define NumOf_CGRAM_Patterns     8
00065
00066 #define FirstByteInCGRAM_Pattern 0
00067
00068 #define LastByteInCGRAM_Pattern 8
00069
00070
00071
00072
00073 #endif /* LCD_LCD_CONFIG_H */

```

7.49 LCD_interface.h

```

00001 /*
00002 * LCD_interface.h
00003 *
00004 * Created on: Sep 11, 2021
00005 * Author: Ali El Bana
00006 */
00007
00008 #ifndef LCD_LCD_INTERFACE_H_
00009 #define LCD_LCD_INTERFACE_H_
00010
00011
00012 /***** Functions prototypes *****/
00013 /* Functions prototypes */
00014 /***** */
00015
00016 void HLCD_vInit           ( void ) ; ;
00017
00018 void HLCD_vSendCommand   ( u8_t A_u8Cmd ) ; ;
00019
00020 void HLCD_vSendData      ( u8_t A_u8Data ) ; ;
00021
00022 void HLCD_vClear         ( void ) ; ;
00023
00024 void HLCD_vDispString   ( c8_t * A_c8Char ) ; ;
00025
00026 void HLCD_vDispNumber   ( VAR(s32_t) A_s32Num ) ; ;
00027
00028 void HLCD_vSaveCustomChar ( u8_t A_u8Address, u8_t * A_u8CustomChar ) ; ;
00029
00030 void HLCD_vSaveDispChar ( u8_t A_u8Address, u8_t * A_u8CustomChar ) ; ;
00031
00032 void HLCD_vGoTo         ( u8_t A_u8Row, u8_t A_u8Col ) ; ;
00033
00034 void HLCD_vSetShiftLeftOn (void) ; ;
00035
00036 void HLCD_vSetDispOFF   (void) ; ;
00037
00038 void HLCD_vDispCursorWithBlinking (void) ; ;
00039
00040 void HLCD_vSetCursorBlinkingOFF (void) ; ;
00041
00042 void HLCD_vDispShiftLeftString ( c8_t * A_c8Char ) ; ;
00043
00044 void HLCD_vClearChar    ( u8_t A_u8Row, u8_t A_u8Col ) ; ;
00045
00046 /***** Interfacing macros *****/
00047 /* Interfacing macros */
00048 /***** */
00049
00050 #define MIN_IDX_OF_ROWS 0

```

```
00051 #define MAX_IDX_OF_ROWS 1
00053
00054 #define MIN_IDX_OF_COL 0
00055
00056 #define MAX_IDX_OF_COL 15
00057
00058
00059 #define HLCD_LINE1 0
00060
00061 #define HLCD_LINE2 1
00062
00063
00064 #define HLCD_Square1 0
00065
00066 #define HLCD_Square2 1
00067
00068 #define HLCD_Square3 2
00069
00070 #define HLCD_Square4 3
00071
00072 #define HLCD_Square5 4
00073
00074 #define HLCD_Square6 5
00075
00076 #define HLCD_Square7 6
00077
00078 #define HLCD_Square8 7
00079
00080 #define HLCD_Square9 8
00081
00082 #define HLCD_Square10 9
00083
00084 #define HLCD_Square11 10
00085
00086 #define HLCD_Square12 11
00087
00088 #define HLCD_Square13 12
00089
00090 #define HLCD_Square14 13
00091
00092 #define HLCD_Square15 14
00093
00094 #define HLCD_Square16 15
00095
00096
00097 #define CGRAM_AddressOfPattern0 0
00098
00099 #define CGRAM_AddressOfPattern1 1
00100
00101 #define CGRAM_AddressOfPattern2 2
00102
00103 #define CGRAM_AddressOfPattern3 3
00104
00105 #define CGRAM_AddressOfPattern4 4
00106
00107 #define CGRAM_AddressOfPattern5 5
00108
00109 #define CGRAM_AddressOfPattern7 6
00110
00111 #define CGRAM_AddressOfPattern8 7
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130 #endif /* LCD_LCD_INTERFACE_H_ */
```

7.50 LCD_private.h

```
00001 /*
```

```

00002 * LCD_private.h
00003 *
00004 * Created on: Sep 11, 2021
00005 * Author: Ali El Bana
00006 */
00007
00008 #ifndef LCD_LCD_PRIVATE_H_
00009 #define LCD_LCD_PRIVATE_H_
00010
00011
00012
00013
00014 #define FuctionSetCmd          0b00111000
00015
00016 #define DispOnOffCTRLCmd       0b00001100
00017
00018 #define DispCursorWithBlinkingCmd 0b00001111
00019
00020 #define SetCursorBlinkingOFFCmd 0b00001110
00021
00022 #define DispOffCmd            0b00001000
00023
00024 #define DispClearCmd          0b00000001
00025
00026 #define EntryModeSet_Cmd      0b00000110
00027
00028 #define EntryModeSet_ShiftLeftOn_Cmd 0b0000011000
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055 #endif /* LCD_LCD_PRIVATE_H_ */

```

7.51 LCD_program.c

```

00001 /*
00002 * LCD_programme.c
00003 *
00004 * Created on: Sep 11, 2021
00005 * Author: Ali El Bana
00006 */
00007
00008 /***** Include headers *****/
00009 /*           Include headers           */
00010 /***** Include headers *****/
00011
00012 #include "../../LIB/LSTD_TYPES.h"
00013 #include "../../LIB/LSTD_COMPILER.h"
00014 #include "../../LIB/LSTD_VALUES.h"
00015 #include "../../LIB/LSTD_BITMATH.h"
00016
00017 #include "../../MCAL/RCC/MRCC_interface.h"
00018 #include "../../MCAL/GPIO/GPIO_interface.h"
00019 #include "../../MCAL/SysTick/SysTick_interface.h"
00020
00021 #include "LCD_config.h"
00022 #include "LCD_interface.h"
00023 #include "LCD_private.h"
00024
00025
00026 /*****

```

```

00027 /* Functions implementations */
00028 /***** */
00029
00030 void HLCD_vInit(void)
00031 {
00032
00033     MSysTick_vInit( );
00034
00035     MSysTick_vDelayMilliSec( 1000 ) ;
00036
00037     // Set your pin directions:
00038     MGPIOx_ConfigType RS =
00039     {
00040
00041         .Port      = HLCD_CTRL_PORT, .Pin = HLCD_RS_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00042             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00043             .InputType  = GPIOx_NoPull
00044     } ;
00045
00046     MGPIOx_ConfigType RW =
00047     {
00048
00049         .Port      = HLCD_CTRL_PORT, .Pin = HLCD_RW_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00050             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00051             .InputType  = GPIOx_NoPull
00052     } ;
00053
00054     MGPIOx_ConfigType EN =
00055     {
00056
00057         .Port      = HLCD_CTRL_PORT, .Pin = HLCD_EN_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00058             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00059             .InputType  = GPIOx_NoPull
00060     } ;
00061
00062     MGPIOx_ConfigType D0 =
00063     {
00064
00065         .Port      = HLCD_DATA_PORT, .Pin = D0_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00066             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00067             .InputType  = GPIOx_NoPull
00068     } ;
00069
00070     MGPIOx_ConfigType D1 =
00071     {
00072
00073         .Port      = HLCD_DATA_PORT, .Pin = D1_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00074             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00075             .InputType  = GPIOx_NoPull
00076     } ;
00077
00078     MGPIOx_ConfigType D2 =
00079     {
00080
00081         .Port      = HLCD_DATA_PORT, .Pin = D2_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00082             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00083             .InputType  = GPIOx_NoPull
00084     } ;
00085
00086     MGPIOx_ConfigType D3 =
00087     {
00088
00089         .Port      = HLCD_DATA_PORT, .Pin = D3_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00090             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00091             .InputType  = GPIOx_NoPull
00092     } ;
00093
00094     MGPIOx_ConfigType D4 =
00095     {
00096
00097         .Port      = HLCD_DATA_PORT, .Pin = D4_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00098             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00099             .InputType  = GPIOx_NoPull
00100     } ;
00101
00102     MGPIOx_ConfigType D5 =
00103     {
00104
00105         .Port      = HLCD_DATA_PORT, .Pin = D5_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =

```

```

00106     .OutputSpeed = GPIOx_LowSpeed,
00107     .InputType   = GPIOx_NoPull
00108 } ;
00109
00110     MGPIOx_ConfigType D6 =
00111 {
00112
00113     .Port      = LCD_DATA_PORT, .Pin = D6_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00114     GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00115     .InputType   = GPIOx_NoPull
00116 } ;
00117
00118     MGPIOx_ConfigType D7 =
00119 {
00120
00121     .Port      = LCD_DATA_PORT, .Pin = D7_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00122     GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00123     .InputType   = GPIOx_NoPull
00124 } ;
00125
00126     MGPIOx_vInit( &RS ) ;
00127     MGPIOx_vInit( &RW ) ;
00128     MGPIOx_vInit( &EN ) ;
00129     MGPIOx_vInit( &D0 ) ;
00130     MGPIOx_vInit( &D1 ) ;
00131     MGPIOx_vInit( &D2 ) ;
00132     MGPIOx_vInit( &D3 ) ;
00133     MGPIOx_vInit( &D4 ) ;
00134     MGPIOx_vInit( &D5 ) ;
00135     MGPIOx_vInit( &D6 ) ;
00136     MGPIOx_vInit( &D7 ) ;
00137
00138 // Start initialization sequence.
00139 MSysTick_vDelayMilliSec( 50 ) ;
00140
00141 LCD_vSendCommand( LCD_FuctionSet_Cmd ) ;
00142
00143 MSysTick_vDelayMilliSec( 1 ) ;
00144
00145 LCD_vSendCommand( LCD_DisponOffCTRL_Cmd ) ;
00146
00147 MSysTick_vDelayMilliSec( 1 ) ;
00148
00149 LCD_vSendCommand( LCD_Dispclear_Cmd ) ;
00150
00151 MSysTick_vDelayMilliSec( 3 ) ;
00152
00153 LCD_vSendCommand( LCD_EntryModeSet_Cmd ) ;
00154
00155 MSysTick_vDelayMilliSec( 1 ) ;
00156
00157 }
00158
00159
00160 //*****
00161 //*****
00162 void LCD_vSendCommand( u8_t A_u8Cmd )
00163 {
00164
00165 // Set RS pin low to send a command.
00166 MGPIOx_vSetValue( LCD_CTRL_PORT, LCD_RS_PIN, GPIOx_LOW ) ;
00167
00168 // Set RW pin low to write the command.
00169 MGPIOx_vSetValue( LCD_CTRL_PORT, LCD_RW_PIN, GPIOx_LOW ) ;
00170
00171 MSysTick_vDelayMilliSec( 1 ) ;
00172
00173 // Put the command on the data bus.
00174 GPIO_vSetNibbleLowValue( LCD_DATA_PORT, A_u8Cmd ) ;
00175
00176 MSysTick_vDelayMilliSec( 1 ) ;
00177
00178 // Set the enable pin high.
00179 MGPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_HIGH ) ;
00180
00181 MSysTick_vDelayMilliSec( 1 ) ;
00182
00183 // Set the enable pin low.
00184 MGPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_LOW ) ;
00185
00186 MSysTick_vDelayMilliSec( 1 ) ;
00187

```

```

00188 }
00189
00190
00191 //*****
00192 //*****
00193 void HLCD_vSendData(u8_t A_u8Data)
00194 {
00195
00196     // Set RS pin high to send data.
00197     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_RS_PIN, GPIOx_HIGH ) ;
00198
00199     // Set RW pin low to write the data.
00200     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_RW_PIN, GPIOx_LOW ) ;
00201
00202     MSysTick_vDelayMilliSec( 1 ) ;
00203
00204     // Put the data on the data bus.
00205     GPIO_vSetNibbleLowValue( LCD_DATA_PORT, A_u8Data ) ;
00206
00207     MSysTick_vDelayMilliSec( 1 ) ;
00208
00209     // Set the enable pin high.
00210     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_HIGH ) ;
00211
00212     MSysTick_vDelayMilliSec( 1 ) ;
00213
00214     // Set the enable pin low.
00215     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_LOW ) ;
00216
00217     MSysTick_vDelayMilliSec( 1 ) ;
00218
00219 }
00220
00221
00222 //*****
00223
00224 void HLCD_vClear(void)
00225 {
00226
00227     MSysTick_vDelayMilliSec( 1 ) ;
00228
00229     LCD_vSendCommand( DispClear_Cmd ) ;
00230
00231     MSysTick_vDelayMilliSec( 1 ) ;
00232
00233 }
00234
00235
00236
00237
00238 void HLCD_vDispString( c8_t * A_c8Char )
00239 {
00240
00241     for( VAR(u8_t) L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00242     {
00243
00244         LCD_vSendData( A_c8Char[L_u8I] ) ;
00245
00246     }
00247
00248 }
00249
00250
00251
00252
00253 void HLCD_vDispNumber( VAR(s32_t) A_s32Num )
00254 {
00255
00256     VAR(u8_t) L_u8Counter      = 0 ;
00257
00258     VAR(u8_t) LR_u8Digits[10]   = {0} ;
00259
00260     // In case the number is signed.
00261     if( A_s32Num < 0 )
00262     {
00263
00264         LCD_vSendData( '-' ) ;
00265
00266         A_s32Num = A_s32Num * ( -1 ) ;

```

```

00267
00268 }
00269
00270 // In case the number the number contains zeros.
00271 if( A_s32Num == 0 )
00272 {
00273     LCD_vSendData( '0' );
00274
00275 }
00276
00277 // Save reversed digits in the array.
00278 for( L_u8Counter = 0 ; A_s32Num != 0 ; L_u8Counter++ )
00279 {
00280     LR_u8Digits[L_u8Counter] = A_s32Num % 10 ;
00281
00282     A_s32Num /= 10 ;
00283
00284 }
00285
00286 // For arrangement the digits.
00287 for (VAR(s8_t) L_s8ArrangedNums = (L_u8Counter - 1); L_s8ArrangedNums >= 0; L_s8ArrangedNums--)
00288 {
00289
00290     LCD_vSendData( '0' + LR_u8Digits[ L_s8ArrangedNums ] ) ;
00291
00292 }
00293
00294 }
00295
00296
00297
00298 /*****
00299 ****/
00300 // void LCD_vDispNum(s32 A_s32Num)
00301 //{
00302 //
00303 //
00304 //    VAR(u32_t) L_u32Rev = 0 ;
00305 //
00306 //    while( A_s32Num >0 )
00307 //
00308 //
00309 //        L_u32Rev = L_u32Rev*10 + (A_s32Num % 10) ;
00310 //
00311 //        A_s32Num /= 10 ;
00312 //
00313 //
00314 //
00315 //
00316 //    while( L_u32Rev >0 )
00317 //
00318 //
00319 //        LCD_vSendData( (L_u32Rev % 10) + '0' ) ; // Convert from integer to char.
00320 //
00321 //        L_u32Rev /= 10 ;
00322 //
00323 //
00324 //
00325 //
00326 //
00327 //
00328 //
00329
00330
00331 /*****
00332 ****/
00333 void LCD_vSaveCustomChar( u8_t A_u8Address, u8_t * A_u8CustomChar )
00334 {
00335
00336 // 1-Set CGRAM Address.
00337 LCD_vSendCommand( SET_CGRAM_AC_MASK + (NumOf_CGRAM_Patterns * A_u8Address) ) ;
00338
00339 // 2- Send custom char data.
00340 for( u8_t L_u8I = FirstByteInCGRAM_Pattern ; L_u8I < LastByteInCGRAM_Pattern ; L_u8I++ )
00341 {
00342
00343     LCD_vSendData( A_u8CustomChar[L_u8I] ) ;
00344
00345 }
00346
00347 // 3-Set DDRAM address
00348 LCD_vSendCommand( SET_DDRAM_AC_MASK ) ;
00349

```

```
00350 }
00351
00352 /******
00353 *****/
00354
00355 void HLCD_vGoTo( u8_t A_u8Row, u8_t A_u8Col )
00356 {
00357
00358     // Valid Range.
00359     if ( (A_u8Row <= MAX_IDX_OF_ROWS) && (A_u8Col <= MAX_IDX_OF_COL) )
00360     {
00361         switch( A_u8Row )
00362         {
00363             case 0: HLCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + FIRST_ROW_START ) ; break;
00364             case 1: HLCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + SEC_ROW_START ) ; break;
00365         }
00366     }
00367
00368     else
00369     {
00370         // Do Nothing
00371     }
00372
00373
00374
00375
00376
00377
00378
00379 }
00380
00381
00382 /******
00383 *****/
00384
00385 void HLCD_vSetShiftLeftOn(void)
00386 {
00387
00388     MSysTick_vDelayMilliSec( 500 ) ;
00389
00390     HLCD_vSendCommand( HLCD_EntryModeSet_ShiftLeftOn_Cmd ) ;
00391
00392     MSysTick_vDelayMilliSec( 500 ) ;
00393
00394 }
00395
00396
00397
00398 /******
00399 *****/
00400 void HLCD_vSetDispOFF(void)
00401 {
00402
00403     MSysTick_vDelayMilliSec( 500 ) ;
00404
00405     HLCD_vSendCommand( HLCD_DispOff_Cmd ) ;
00406
00407     MSysTick_vDelayMilliSec( 500 ) ;
00408
00409 }
00410
00411
00412 /******
00413 *****/
00414
00415 void HLCD_vDispCursorWithBlinking(void)
00416 {
00417
00418     MSysTick_vDelayMilliSec( 500 ) ;
00419
00420     HLCD_vSendCommand( HLCD_DispCursorWithBlinking_Cmd ) ;
00421
00422     MSysTick_vDelayMilliSec( 500 ) ;
00423
00424 }
00425
00426
00427 /******
```

```

00428 //*****
00429 void LCD_vSetCursorBlinkingOFF(void)
00430 {
00431     MSysTick_vDelayMilliSec( 500 ) ;
00432     LCD_vSendCommand( LCD_SetCursorBlinkingOFF_Cmd ) ;
00433     MSysTick_vDelayMilliSec( 500 ) ;
00434 }
00435
00436 //*****
00437
00438 }
00439
00440 //*****
00441 //*****
00442
00443 void LCD_vDispShiftLeftString( c8_t * A_c8Char )
00444 {
00445
00446     VAR(u8_t) L_u8I = 0 ;
00447
00448     for( L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00449     {
00450         LCD_vSendData( A_c8Char[L_u8I] ) ;
00451
00452
00453     }
00454
00455     MSysTick_vDelayMilliSec( 500 ) ;
00456
00457     if( L_u8I > LCD_CharactersNums )
00458     {
00459
00460         for( u8_t i = 0 ; i < L_u8I - LCD_CharactersNums ; i++ )
00461         {
00462             LCD_vSetShiftLeftOn() ;
00463
00464         }
00465
00466     }
00467 }
00468
00469 }
00470
00471 //*****
00472 //*****
00473
00474 void LCD_vClearChar( u8_t A_u8Row, u8_t A_u8Col )
00475 {
00476
00477     LCD_vGoTo( A_u8Row, A_u8Col ) ;
00478
00479     LCD_vSendData( ' ' ) ;
00480
00481 }
00482
00483
00484 //*****
00485 //*****
00486
00487
00488
00489
00490
00491
00492
00493

```

7.52 LDR_config.c

```

00001 /*
00002 * LDR_config.c
00003 *
00004 * Created on: Dec 22, 2022
00005 * Author: Ali El Bana
00006 */
00007

```

```

00008 /*****  
00009 /*           Include headers           */  
00010 *****/  
00011  
00012 #include "../../LIB/LSTD_TYPES.h"  
00013 #include "../../LIB/LSTD_COMPILER.h"  
00014 #include "../../LIB/LSTD_VALUES.h"  
00015 #include "../../LIB/LSTD_BITMATH.h"  
00016  
00017 #include "../../MCAL/GPIO/GPIO_interface.h"  
00018 #include "../../MCAL/ADC/ADC_interface.h"  
00019  
00020 #include "LDR_config.h"  
00021  
00022 /*****  
00023 /*           Configuration parameters       */  
00024 *****/  
00025  
00026 VAR(MGPIox_ConfigType) LDR_Vo =  
00027 {  
00028     .Port      = LDR_Vo_PORT,  
00029     .Pin       = LDR_Vo_PIN,  
00030     .Mode      = GPIOx_MODE_ANALOG,  
00031  
00032 } ;  
00033  
00034  
00035  
00036  
00037  
00038  
00039 /*****  
00040  
00041 VAR(u8_t) LDR_CHANNEL = CHANNEL8 ;  
00042  
00043  
00044 /*****  
00045  
00046  
00047  
00048  
00049  
00050  
00051  
00052

```

7.53 LDR_config.h

```

00001 /* FILENAME: LDR_config  
00002 * Author:   Ali El Bana  
00003 * Version:  V1.0  
00004 * DATE:    Thu 12/22/2022  
00005 */  
00006 #ifndef _LDR_config_H  
00007 #define _LDR_config_H  
00008  
00009 /*****  
00010 /*           Configuration macros           */  
00011 *****/  
00012  
00013 #define LDR_Vo_PORT GPIO_PORTB  
00014  
00015 #define LDR_Vo_PIN GPIOx_PIN0  
00016  
00017  
00018  
00019 #endif // _LDR_config_H

```

7.54 LDR_interface.h

```

00001 /* FILENAME: LDR_interface  
00002 * Author:   Ali El Bana  
00003 * Version:  V1.0  
00004 * DATE:    Thu 12/22/2022  
00005 */

```

```

00006 #ifndef _LDR_interface_H
00007 #define _LDR_interface_H
00008
00009
00010 /***** Interfacing macros *****/
00011 /*          */
00012 /***** */
00013
00014 #define DARK      3500
00015
00016 #define VERY_DIM   3000
00017
00018 #define MODERATE  2000
00019
00020 #define OVERCAST   1500
00021
00022 #define VERY_BRIGHT 1000
00023
00024 /***** Functions prototypes *****/
00025 /*          */
00026 /***** */
00027
00028 void HLDR_vInit( void ) ;
00029
00030 u16_t HLDR_u16DigitalOutputValue( void ) ;
00031
00032
00033 #endif // _LDR_interface_H

```

7.55 LDR_private.h

```

00001 /* FILENAME: LDR_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 12/22/2022
00005 */
00006 #ifndef _LDR_private_H
00007 #define _LDR_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _LDR_private_H

```

7.56 LDR_program.c

```

00001 /* FILENAME: LDR_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 12/22/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*          */
00009 /***** */
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013 #include "../../LIB/LSTD_VALUES.h"
00014 #include "../../LIB/LSTD_BITMATH.h"
00015
00016 #include "../../MCAL/RCC/MRCC_interface.h"
00017 #include "../../MCAL/GPIO/GPIO_interface.h"
00018 #include "../../MCAL/ADC/ADC_interface.h"
00019
00020 #include "LDR_interface.h"
00021 #include "LDR_private.h"
00022 #include "LDR_config.h"
00023
00024 /***** Externed variables *****/
00025 /*          */
00026 /***** */
00027
00028 extern VAR(MGPIox_ConfigType) LDR_Vo;
00029
00030 extern VAR(u8_t) LDR_CHANNEL;
00031

```

```
00032 /*****  
00033 /* Functions' implementations */  
00034 *****/  
00035  
00036 void HLDR_vInit( void )  
00037 {  
00038     MGPIox_vInit( &LDR_Vo ) ;  
00039     MADC_vInit( ) ;  
00040  
00041 }  
00042  
00043 }  
00044  
00045  
00046 /*****  
00047  
00048 u16_t HLDR_u16DigitalOutputValue( void )  
00049 {  
00050     VAR(u16_t) L_u16DigitalResult = INITIAL_ZERO ;  
00051  
00052     L_u16DigitalResult = MADC_u16ConvertToDigital( LDR_CHANNEL ) ;  
00053  
00054     return L_u16DigitalResult ;  
00055  
00056 }  
00057 }  
00058  
00059  
00060 /*****  
00061  
00062  
00063  
00064  
00065  
00066  
00067  
00068
```

7.57 LED_config.h

```
00001 /* FILENAME: LED_config.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00006 #ifndef _LED_CONFIG_H
00007 #define _LED_CONFIG_H
00008
00009 #include "../../MCAL/GPIO/GPIO_interface.h"
00010
00011 /*****
00012 /* Configuration macros
00013 *****/
00014
00015 #define LED_PORT GPIO_PORTA
00016
00017 #define LED_PIN GPIOx_PIN0
00018
00019 #endif // _LED_CONFIG_H
```

7.58 LED_interface.h

```
00001 /* FILENAME: LED_interface.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00006 #ifndef _LED_INTERFACE_H
00007 #define _LED_INTERFACE_H
00008
00009 /*****
00010 *****/
00011
00017 typedef struct
00018 {
```

```

00022     u8_t u8Port;
00026     u8_t u8Pin;
00027
00028 } LED_LEDConfiguration;
00029
00030 /***** Functions prototypes ****/
00031 /*
00032 /*****
00033
00038 void HLED_vInit(LED_LEDConfiguration * pLED_Cfg);
00039
00044 void HLED_vTurnLightOn(LED_LEDConfiguration * pLED_Cfg);
00045
00050 void HLED_vTurnLightOff(LED_LEDConfiguration * pLED_Cfg);
00051
00056 void HLED_vToggleLight(LED_LEDConfiguration * pLED_Cfg);
00057
00063 void HLED_vBlinkLED(LED_LEDConfiguration * pLED_Cfg, u32_t A_u32MilliSecs);
00064
00065 #endif // _LED_INTERFACE_H

```

7.59 LED_private.h

```
00001 /* FILENAME: LED_private.h
00002 * Author: Mohamed Alaa
00003 * Version: V1.0
00004 * DATE: 1/27/2022
00005 */
00006 #ifndef _LED_PRIVATE_H
00007 #define _LED_PRIVATE_H
00008
00009 #endif // _LED_PRIVATE_H
```

7.60 LED_program.c

```
00001 /* FILENAME: LDR_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 12/22/2022
00005 */
00006
00007 /*****
00008 /*           Include headers
00009 /*****
00010
00011 #include "../../LIB/LSTD_TYPES.h"
00012 #include "../../LIB/LSTD_COMPILER.h"
00013
00014 #include "../../MCAL/GPIO/GPIO_interface.h"
00015 #include "../../MCAL/SysTick/SysTick_interface.h"
00016
00017 #include "LED_private.h"
00018 #include "LED_interface.h"
00019 #include "LED_config.h"
00020
00021 /*****
00022 /*           Functions' implementations
00023 /*****
00024
00025 void HLED_vInit(LED_LEDConfiguration * pLED_Cfg)
00026 {
00027     VAR(MGPIOx_ConfigType)
00028     LED_Cfg =
00029     {
00030         .Port = pLED_Cfg->u8Port,
00031         .Pin = pLED_Cfg->u8Pin,
00032         .Mode = GPIOx_MODE_OUTPUT,
00033         .OutputType = GPIOx_PUSH_PULL,
00034         .OutputSpeed = GPIOx_LowSpeed,
00035         .InputType = GPIOx_NoPull};
00036
00037     MGPIOx_vInit(&LED_Cfg);
00038 }
00039
00040
00041 /*****
00042 /*****
```

```

00043 void HLED_vTurnLightOn(LED_LEDConfiguration * pLED_Cfg)
00044 {
00045     MGPIox_vSetValue(pLED_Cfg->u8Port, pLED_Cfg->u8Pin, GPIOx_HIGH);
00046 }
00047
00048 /*****
00049 ****
00050
00051 void HLED_vTurnLightOff(LED_LEDConfiguration * pLED_Cfg)
00052 {
00053     MGPIox_vSetValue(pLED_Cfg->u8Port, pLED_Cfg->u8Pin, GPIOx_LOW);
00054 }
00055
00056 /*****
00057 ****
00058
00059 void HLED_vToggleLight(LED_LEDConfiguration * pLED_Cfg)
00060 {
00061     MGPIox_vTogglePinValue(pLED_Cfg->u8Port, pLED_Cfg->u8Pin);
00062 }
00063
00064 /*****
00065 ****
00066
00067 void HLED_vBlinkLED(LED_LEDConfiguration * pLED_Cfg, u32_t A_u32MilliSecs)
00068 {
00069
00070     MGPIox_vSetValue(pLED_Cfg->u8Port, pLED_Cfg->u8Pin, GPIOx_HIGH);
00071
00072     MSysTick_vDelayMilliSec(A_u32MilliSecs);
00073
00074     MGPIox_vSetValue(pLED_Cfg->u8Port, pLED_Cfg->u8Pin, GPIOx_LOW);
00075
00076     MSysTick_vDelayMilliSec(A_u32MilliSecs);
00077 }
00078
00079 /*****
00080 ****

```

7.61 TFT_Config.c

```

00001 /*
00002 * TFT_Config.c
00003 *
00004 * Created on: Oct 13, 2022
00005 * Author: Ali El Bana
00006 */
00007
00008 #include "../../LIB/LSTD_TYPES.h"
00009
00010 #include "../../MCAL/GPIO/GPIO_interface.h"
00011
00012 #include "TFT_config.h"
00013
00014 MGPIox_ConfigType TFT_A0 =
00015 {
00016     .Port      = A0_PORT      , .Pin      = A0_PIN
00017     .Mode     = GPIOx_MODE_OUTPUT , .OutputType = GPIOx_PUSH_PULL ,
00018     .OutputSpeed = GPIOx_LowSpeed , .InputType = GPIOx_NoPull
00019 } ;
00020
00021
00022 MGPIox_ConfigType TFT_RST =
00023 {
00024     .Port      = RST_PORT      , .Pin      = RST_PIN
00025     .Mode     = GPIOx_MODE_OUTPUT , .OutputType = GPIOx_PUSH_PULL ,
00026     .OutputSpeed = GPIOx_LowSpeed , .InputType = GPIOx_NoPull
00027 } ;
00028
00029
00030
00031
00032
00033
00034

```

```
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
```

7.62 TFT_config.h

```
00001 /* FILENAME: TFT_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 10/13/2022
00005 */
00006 #ifndef _TFT_config_H
00007 #define _TFT_config_H
00008
00009
00010 #define A0_PORT          GPIO_PORTA
00011 #define RST_PORT          GPIO_PORTA
00012
00013 #define A0_PIN             GPIOx_PIN1
00014 #define RST_PIN            GPIOx_PIN0
00015
00016 #define SPI_SRC           SPI1
00017 #define SPI_RELATION       SPIx_MSTR
00018 #define SPI_DATA_DIRECTION SPIx_FULL_DUPLEX
00019
00020 #define SLPOUT_CMD         0x11
00021 #define DISPON_CMD         0x29
00022 #define XPOSITION_CMD      0x2A
00023 #define YPOSITION_CMD      0x2B
00024 #define COLOR_MODE_CMD     0x3A
00025 #define MEMORY_WRITE_CMD   0x2C
00026 #define RGB565_CMD         0x05
00027 #define RGB444              0x03
00028 #define RGB666              0x06
00029
00030
00031 #endif // _TFT_config_H
```

7.63 TFT_interface.h

```
00001 /* FILENAME: TFT_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 10/13/2022
00005 */
00006 #ifndef _TFT_interface_H
00007 #define _TFT_interface_H
00008
00009
00010 void HTFT_vInit( void ) ;
00011
00012 void HTFT_vShowImage( u16_t const *A_u16Image, u32_t A_u32ImageSize ) ;
00013
00014 void HTFT_vSetXPosition( u16_t A_u16XStart, u16_t A_u16XEnd ) ;
00015
00016 void HTFT_vSetYPosition( u16_t A_u16YStart, u16_t A_u16YEnd ) ;
00017
00018 void HTFT_vFillRectangle( u16_t A_u16Color ) ;
00019
00020 void HTFT_vFillBackground( u16_t A_u16Color ) ;
00021
00022 void HTFT_vDrawLine( u16_t A_u16X1, u16_t A_u16X2, u16_t A_u16Y1, u16_t A_u16Y2 ) ;
00023
00024
00025
00026
00027
00028
00029
00030
```

```

00031
00032
00033
00034
00035
00036
00037
00038
00039
00040 #endif // _TFT_interface_H

```

7.64 TFT_private.h

```

00001 /* FILENAME: TFT_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 10/13/2022
00005 */
00006 #ifndef _TFT_private_H
00007 #define _TFT_private_H
00008
00009
00010
00011
00012
00013
00014 #endif // _TFT_private_H

```

7.65 TFT_program.c

```

00001 /* FILENAME: TFT_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 10/13/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*           */
00009 /***** */
00010 #include "../../LIB/LSTD_TYPES.h"
00011 #include "../../LIB/LSTD_COMPILER.h"
00012 #include "../../LIB/LSTD_VALUES.h"
00013 #include "../../LIB/LSTD_BITMATH.h"
00014
00015 #include "../../../../MCAL/RCC/MRCC_interface.h"
00016 #include "../../../../MCAL/GPIO/GPIO_interface.h"
00017 #include "../../../../MCAL/Systick/SysTick_interface.h"
00018 #include "../../../../MCAL/SPI/SPI_interface.h"
00019
00020 #include "TFT_interface.h"
00021 #include "TFT_private.h"
00022 #include "TFT_config.h"
00023
00024 extern MGPIOx_ConfigType TFT_A0 ;
00025 extern MGPIOx_ConfigType TFT_RST ;
00026
00027 static ul6_t CurrentXStart      = INITIAL_ZERO ;
00028 static ul6_t CurrentXEnd       = INITIAL_ZERO ;
00029 static ul6_t CurrentYStart      = INITIAL_ZERO ;
00030 static ul6_t CurrentYEnd       = INITIAL_ZERO ;
00031
00032 /*****
00033 /*****
00034
00035 void HTFT_vWriteCMD( ul6_t A_u16CMD )
00036 {
00037
00038     MGPIOx_vSetValue( TFT_A0.Port, TFT_A0.Pin, GPIOx_LOW ) ;
00039
00040     (void)MSPI_u16Transceive( SPI_SRC, A_u16CMD ) ;
00041
00042 }
00043
00044 /*****
00045 /*****

```

```

00046
00047 void HTFT_vWriteData( u16_t A_u16Data )
00048 {
00049
00050     MGPIOx_vSetPinValue( TFT_A0.Port, TFT_A0.Pin, GPIOx_HIGH ) ;
00051
00052     (void)MSPI_u16Transcieve( SPI_SRC, A_u16Data ) ;
00053
00054 }
00055
00056
00057 /*****
00058 *****/
00059 void HTFT_vResetSequence( void )
00060 {
00061
00062     MGPIOx_vSetPinValue( TFT_RST.Port, TFT_RST.Pin, GPIOx_HIGH ) ;
00063
00064     MSysTick_vDelayMicroSec( 150 ) ; //100us
00065
00066     MGPIOx_vSetPinValue( TFT_RST.Port, TFT_RST.Pin, GPIOx_LOW ) ;
00067
00068     MSysTick_vDelayMicroSec( 5 ) ; //1us
00069
00070     MGPIOx_vSetPinValue( TFT_RST.Port, TFT_RST.Pin, GPIOx_HIGH ) ;
00071
00072     MSysTick_vDelayMicroSec( 150 ) ; //100us
00073
00074     MGPIOx_vSetPinValue( TFT_RST.Port, TFT_RST.Pin, GPIOx_LOW ) ;
00075
00076     MSysTick_vDelayMicroSec( 5 ) ; //1us
00077
00078     MGPIOx_vSetPinValue( TFT_RST.Port, TFT_RST.Pin, GPIOx_HIGH ) ;
00079
00080     MSysTick_vDelayMilliSec( 150 ) ; //120ms
00081
00082 }
00083
00084
00085 /*****
00086 *****/
00087 void HTFT_vInit( void )
00088 {
00089
00090     // Set pin directions:
00091     MGPIOx_vInit( &TFT_A0 ) ;
00092     MGPIOx_vInit( &TFT_RST ) ;
00093
00094     MSPI_vInit( SPI_SRC, SPI_RELATION, SPI_DATA_DIRECTION ) ;
00095
00096     MSysTick_vInit() ;
00097
00098     // Rest sequence:
00099     HTFT_vResetSequence( ) ;
00100
00101     // Send SLPOUT CMD (0x11):
00102     HTFT_vWriteCMD( SLPOUT_CMD ) ;
00103
00104     // Wait 15ms:
00105     MSysTick_vDelayMilliSec( 15 ) ;
00106
00107     // Send color mode CMD -> RGB565:
00108     HTFT_vWriteCMD( COLOR_MODE_CMD ) ;
00109     HTFT_vWriteData( RGB565_CMD ) ;
00110
00111     // DISPON CMD (0X29):
00112     HTFT_vWriteCMD( DISPON_CMD ) ;
00113
00114 }
00115
00116
00117 /*****
00118 *****/
00119 void HTFT_vShowImage( u16_t const *A_u16Image, u32_t A_u32ImageSize )
00120 {
00121
00122     // Set X position:
00123     HTFT_vWriteCMD( XPOSITION_CMD ) ;
00124
00125     // Xstart: 0, Xend: 127
00126     HTFT_vWriteData( 0 ) ;

```

```

00127     HTFT_vWriteData( 0 )      ;
00128     HTFT_vWriteData( 0 )      ;
00129     HTFT_vWriteData( 127 )    ;
00130
00131     // Set Y position:
00132     HTFT_vWriteCMD( YPOSITION_CMD ) ;
00133
00134     // Ystart: 0, Yend: 159
00135     HTFT_vWriteData( 0 )      ;
00136     HTFT_vWriteData( 0 )      ;
00137     HTFT_vWriteData( 0 )      ;
00138     HTFT_vWriteData( 159 )    ;
00139
00140     // Send image data:
00141     HTFT_vWriteCMD( MEMORY_WRITE_CMD ) ;
00142
00143     for( u16_t PX = INITIAL_ZERO; PX < A_u32ImageSize; PX++ )
00144     {
00145
00146         HTFT_vWriteData( A_u16Image[PX] >> 8      ) ; // MSB
00147         HTFT_vWriteData( A_u16Image[PX] & 0x00FF ) ; // LSB
00148
00149     }
00150
00151 }
00152
00153
00154 /*****
00155 ****
00156 void HTFT_vSetXPosition( u16_t A_u16XStart, u16_t A_u16XEnd )
00157 {
00158
00159     CurrentXStart    = A_u16XStart      ;
00160     CurrentXEnd     = A_u16XEnd       ;
00161
00162     HTFT_vWriteCMD( XPOSITION_CMD ) ;
00163
00164     // Xstart: 0, Xend: 127
00165     HTFT_vWriteData( 0 )      ;
00166     HTFT_vWriteData( A_u16XStart ) ;
00167     HTFT_vWriteData( 0 )      ;
00168     HTFT_vWriteData( A_u16XEnd ) ;
00169
00170 }
00171
00172
00173 /*****
00174 ****
00175 void HTFT_vSetYPosition( u16_t A_u16YStart, u16_t A_u16YEnd )
00176 {
00177
00178     CurrentYStart    = A_u16YStart      ;
00179     CurrentYEnd     = A_u16YEnd       ;
00180
00181     HTFT_vWriteCMD( YPOSITION_CMD ) ;
00182
00183     // Ystart: 0, Yend: 127
00184     HTFT_vWriteData( 0 )      ;
00185     HTFT_vWriteData( A_u16YStart ) ;
00186     HTFT_vWriteData( 0 )      ;
00187     HTFT_vWriteData( A_u16YEnd ) ;
00188
00189 }
00190
00191
00192 /*****
00193 ****
00194 void HTFT_vFillRectangle( u16_t A_u16Color )
00195 {
00196
00197     u32_t PXLS_Num = (CurrentXEnd-CurrentXStart) * (CurrentYEnd-CurrentYStart) ;
00198
00199     HTFT_vWriteCMD( MEMORY_WRITE_CMD ) ;
00200
00201     for( u8_t PX = 0; PX < PXLS_Num ; PX++ )
00202     {
00203
00204         HTFT_vWriteData( A_u16Color >> 8      ) ; // MSB
00205         HTFT_vWriteData( A_u16Color & 0x00FF ) ; // LSB
00206
00207     }

```

```

00208
00209 }
00210
00211
00212 /******
00213 *****/
00214 void HTFT_vFillBackground( u16_t A_u16Color )
00215 {
00216
00217     // Set X position:
00218     HTFT_vWriteCMD( XPOSITION_CMD ) ;
00219
00220     // Xstart: 0, Xend: 127
00221     HTFT_vWriteData( 0 ) ;
00222     HTFT_vWriteData( 0 ) ;
00223     HTFT_vWriteData( 0 ) ;
00224     HTFT_vWriteData( 127 ) ;
00225
00226     // Set Y position:
00227     HTFT_vWriteCMD( YPOSITION_CMD ) ;
00228
00229     // Ystart: 0, Yend: 127
00230     HTFT_vWriteData( 0 ) ;
00231     HTFT_vWriteData( 0 ) ;
00232     HTFT_vWriteData( 0 ) ;
00233     HTFT_vWriteData( 159 ) ;
00234
00235     HTFT_vFillRectangle( A_u16Color ) ;
00236
00237 }
00238
00239
00240 /******
00241 *****/
00242
00243
00244
00245
00246
00247
00248

```

7.66 UltraSonic_config.h

```

00001 /* FILENAME: UltraSonic_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 02/24/2023
00005 */
00006 #ifndef _UltraSonic_config_H
00007 #define _UltraSonic_config_H
00008
00009
00010
00011
00012
00013
00014
00015
00016
00017
00018
00019
00020
00021
00022
00023 #endif // _UltraSonic_config_H

```

7.67 COTS/HAL/UltraSonic/UltraSonic_interface.h File Reference

This file is a header file that contains the Ultrasonic sensor functions prototypes.

Data Structures

- struct [HULTSNC_ConfigType](#)

Ultrasonic Configurations.

Functions

- void [HULTSNC_vInit \(HULTSNC_ConfigType *TRIG_psCfg\)](#)

This function is responsible for initializing the ultrasonic sensor.

- void [HULTSNC_vTrigger \(HULTSNC_ConfigType *TRIG_psCfg\)](#)

This function is responsible for triggering the ultrasonic sensor.

- [f32_t HULTSNC_f32GetDistance \(void\)](#)

This function is responsible for getting the distance measured by the ultrasonic sensor.

7.67.1 Detailed Description

This file is a header file that contains the Ultrasonic sensor functions prototypes.

Version

1.0

Date

02/24/2023

Author

Ali El Bana

Definition in file [UltraSonic_interface.h](#).

7.67.2 Function Documentation

7.67.2.1 HULTSNC_vInit()

```
void HULTSNC_vInit (
    HULTSNC_ConfigType * TRIG_psCfg )
```

This function is responsible for initializing the ultrasonic sensor.

Parameters

<code>TRIG_psCfg</code>	pointer to a struct that holds the trigger pin configurations
-------------------------	---

Definition at line 28 of file [UltraSonic_program.c](#).

```

00029 {
00030
00031     VAR(MGPIox_ConfigType) TRIG_Cfg =
00032     {
00033         .Port      = TRIG_psCfg->u8Port, .Pin      = TRIG_psCfg->u8Pin      ,
00034         .Mode     = GPIOx_MODE_OUTPUT , .OutputType = GPIOx_PUSHPULL ,
00035         .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull
00036     } ;
00037
00038     MGPIox_vInit( &TRIG_Cfg ) ;
00039
00040     MTIM1_vSetPrescalerValue( 7 ) ;
00041
00042     MTIM1_vSetAutoReloadValue( 10000 ) ;
00043
00044     MTIM1_vReadPWM( ) ;
00045
00046 }
```

References [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSHPULL](#), [HULTSNC_vInit\(\)](#), [MGPIox_vInit\(\)](#), [HULTSNC_ConfigType::u8Pin](#), [HULTSNC_ConfigType::u8Port](#), and [VAR](#).

Referenced by [HDCM_vMotorSpeedCntrl\(\)](#), and [HULTSNC_vInit\(\)](#).

7.67.2.2 HULTSNC_vTrigger()

```
void HULTSNC_vTrigger (
    HULTSNC_ConfigType * TRIG_psCfg )
```

This function is responsible for triggering the ultrasonic sensor.

Parameters

<i>TRIG_psCfg</i>	pointer to a struct that holds the trigger pin configurations
-------------------	---

Definition at line 51 of file [UltraSonic_program.c](#).

```

00052 {
00053
00054     MGPIox_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_LOW ) ;
00055
00056     MSysTick_vDelayMicroSec( 2 ) ;
00057
00058     MGPIox_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_HIGH ) ;
00059
00060     MSysTick_vDelayMicroSec( 10 ) ;
00061
00062     MGPIox_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_LOW ) ;
00063
00064 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [HULTSNC_vTrigger\(\)](#), [MGPIox_vSetResetAtomic\(\)](#), [MSysTick_vDelayMicroSec\(\)](#), [HULTSNC_ConfigType::u8Pin](#), and [HULTSNC_ConfigType::u8Port](#).

Referenced by [HULTSNC_vTrigger\(\)](#).

7.67.2.3 HULTSNC_f32GetDistance()

```
f32_t HULTSNC_f32GetDistance (
    void )
```

This function is responsible for getting the distance measured by the ultrasonic sensor.

Returns

distance measured by the ultrasonic sensor

Definition at line 69 of file [UltraSonic_program.c](#).

```
00070 {
00071
00072     f32_t L_f32PulseTime    = MTIM1_u16GetCaptureReg2Value( )    ;
00073
00074     f32_t L_f32Period       = MTIM1_u16GetCaptureReg1Value( )    ;
00075
00076     f32_t L_f32Distance      = INITIAL_ZERO                      ;
00077
00078     L_f32Distance = (float)( L_f32PulseTime * 0.0174 ) ;
00079
00080     return L_f32Distance ;
00081
00082 }
```

References [HULTSNC_f32GetDistance\(\)](#), and [INITIAL_ZERO](#).

Referenced by [HULTSNC_f32GetDistance\(\)](#).

7.68 UltraSonic_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 #ifndef _UltraSonic_interface_H
0010 #define _UltraSonic_interface_H
0011
0012 /***** Interfacing macros ****/
0013 /*          Functions prototypes          */
0014 /*****
0015
0020 typedef struct
0021 {
0025     u8_t u8Port;
0029     u8_t u8Pin;
0030 } HULTSNC_ConfigType;
0031
0032 /***** Functions prototypes ****/
0033 /*          Functions prototypes          */
0034 /*****
0035
0040 void HULTSNC_vInit(HULTSNC_ConfigType * TRIG_psCfg);
0041
0046 void HULTSNC_vTrigger(HULTSNC_ConfigType * TRIG_psCfg);
0047
0052 f32_t HULTSNC_f32GetDistance(void);
0053
0054 #endif // _UltraSonic_interface_H
```

7.69 UltraSonic_private.h

```
00001 /* FILENAME: UltraSonic_private
00002 * Author:   Ali El Bana
00003 * Version:  V1.0
00004 * DATE:    Fri 02/24/2023
00005 */
00006 #ifndef _UltraSonic_private_H
00007 #define _UltraSonic_private_H
00008
00009
00010
00011
00012
00013
0014 #endif // _UltraSonic_private_H
```

7.70 UltraSonic_program.c

```

00001 /* FILENAME: UltraSonic_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 02/24/2023
00005 */
00006 /***** Include headers *****/
00007 /*           */
00008 /***** */
00009
00010 #include "../../LIB/LSTD_TYPES.h"
00011 #include "../../LIB/LSTD_COMPILER.h"
00012 #include "../../LIB/LSTD_VALUES.h"
00013 #include "../../LIB/LSTD_BITMATH.h"
00014
00015 #include "../../MCAL/RCC/MRCC_interface.h"
00016 #include "../../MCAL/GPIO(GPIO_interface.h"
00017 #include "../../MCAL/SysTick/SysTick_interface.h"
00018 #include "../../MCAL/TIM1/TIM1_interface.h"
00019
00020 #include "UltraSonic_interface.h"
00021 #include "UltraSonic_private.h"
00022 #include "UltraSonic_config.h"
00023
00024 /***** Functions implementations *****/
00025 /*           */
00026 /***** */
00027
00028 void HULTSNC_vInit( HULTSNC_ConfigType * TRIG_psCfg )
00029 {
00030
00031     VAR(MGPIOx_ConfigType) TRIG_Cfg =
00032     {
00033         .Port      = TRIG_psCfg->u8Port, .Pin      = TRIG_psCfg->u8Pin ,
00034         .Mode      = GPIOx_MODE_OUTPUT , .OutputType = GPIOx_PUSH_PULL ,
00035         .OutputSpeed = GPIOx_LowSpeed , .InputType = GPIOx_NoPull
00036     } ;
00037
00038     MGPIOx_vInit( &TRIG_Cfg ) ;
00039
00040     MTIM1_vSetPrescalerValue( 7 ) ;
00041
00042     MTIM1_vSetAutoReloadValue( 10000 ) ;
00043
00044     MTIM1_vReadPWM( ) ;
00045
00046 }
00047
00048
00049 /***** */
00050 /***** */
00051 void HULTSNC_vTrigger( HULTSNC_ConfigType * TRIG_psCfg )
00052 {
00053
00054     MGPIOx_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_LOW ) ;
00055
00056     MSysTick_vDelayMicroSec( 2 ) ;
00057
00058     MGPIOx_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_HIGH ) ;
00059
00060     MSysTick_vDelayMicroSec( 10 ) ;
00061
00062     MGPIOx_vSetResetAtomic( TRIG_psCfg->u8Port, TRIG_psCfg->u8Pin, GPIOx_LOW ) ;
00063
00064 }
00065
00066
00067 /***** */
00068 /***** */
00069 f32_t HULTSNC_f32GetDistance( void )
00070 {
00071
00072     f32_t L_f32PulseTime = MTIM1_u16GetCaptureReg2Value( ) ;
00073
00074     f32_t L_f32Period = MTIM1_u16GetCaptureReg1Value( ) ;
00075
00076     f32_t L_f32Distance = INITIAL_ZERO ;
00077
00078     L_f32Distance = (float)( L_f32PulseTime * 0.0174 ) ;
00079
00080     return L_f32Distance ;
00081

```

```
00082 }
00083
00084 /******
00085 *****/
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
```

7.71 COTS/LIB/LSTD_BITMATH.h File Reference

This file contains the bit math manipulation macro-functions.

Macros

- #define [SET_BIT](#)(Reg, bitnum) (Reg) |= (1 << (bitnum))
- #define [CLR_BIT](#)(Reg, bitnum) (Reg) &= ~(1 << (bitnum))
- #define [TOGGLE_BIT](#)(Reg, bitnum) (Reg) ^= (1 << (bitnum))
- #define [GET_BIT](#)(Reg, bitnum) (((Reg)>>(bitnum)) & 1)
- #define [SET_BITs](#)(Reg, bits, bitnum, factor) (Reg) |= ((bits) << (bitnum * factor))
- #define [CLR_BITs](#)(Reg, bits, bitnum, factor) (Reg) &= ~((bits) << (bitnum * factor))
- #define [TOGGLE_BITs](#)(Reg, bits, bitnum, factor) (Reg) ^= ((bits) << (bitnum * factor))
- #define [GET_BITs](#)(Reg, bits, bitnum, factor) (((Reg) >> (bitnum * factor)) & (bits))

7.71.1 Detailed Description

This file contains the bit math manipulation macro-functions.

Author

Ali El Bana

Version

1.0

Date

10/29/2022

Definition in file [LSTD_BITMATH.h](#).

7.72 LSTD_BITMATH.h

[Go to the documentation of this file.](#)

```

00001
00009 #ifndef COTS_LIB_LSTD_BITMATH_H_
00010 #define COTS_LIB_LSTD_BITMATH_H_
00011
00023 #define SET_BIT(Reg, bitnum) (Reg) |= (1 << (bitnum))
00024
00030 #define CLR_BIT(Reg, bitnum) (Reg) &= ~(1 << (bitnum))
00031
00037 #define TOGGLE_BIT(Reg, bitnum) (Reg) ^= (1 << (bitnum))
00038
00044 #define GET_BIT(Reg, bitnum) (((Reg)>>(bitnum)) & 1)
00045
00059 #define SET_BITS(Reg, bits, bitnum, factor) (Reg) |= ((bits) << (bitnum * factor))
00060
00066 #define CLR_BITS(Reg, bits, bitnum, factor) (Reg) &= ~((bits) << (bitnum * factor))
00067
00073 #define TOGGLE_BITS(Reg, bits, bitnum, factor) (Reg) ^= ((bits) << (bitnum * factor))
00074
00080 #define GET_BITS(Reg, bits, bitnum, factor) (((Reg) >> (bitnum * factor)) & (bits))
00081
00084 #endif /* COTS_LIB_LSTD_BITMATH_H_ */

```

7.73 COTS/LIB/LSTD_COMPILER.h File Reference

This file contains the compiler standard macros.

Macros

- #define **VAR**(vartype) vartype
- #define **FUNC**(rettype) rettype
- #define **P2VAR**(ptrtype) ptrtype *
- #define **P2CONST**(ptrtype) const ptrtype *
- #define **CONSTP2VAR**(ptrtype) ptrtype * const
- #define **CONSTP2CONST**(ptrtype) const ptrtype * const
- #define **P2FUNC**(rettype, fctname) rettype (*fctname)
- #define **CONST**(consttype) const consttype
- #define **STATIC** static

7.73.1 Detailed Description

This file contains the compiler standard macros.

Author

Mohamed Alaa

Version

1.0

Date

11/04/2022

Definition in file [LSTD_COMPILER.h](#).

7.74 LSTD_COMPILER.h

[Go to the documentation of this file.](#)

```

00001
00009 #ifndef COTS_LIB_LSTD_COMPILER_H_
00010 #define COTS_LIB_LSTD_COMPILER_H_
00011
00023 #define VAR(vartype) vartype
00024
00030 #define FUNC(rettype) rettype
00031
00037 #define P2VAR(ptrtype) ptrtype *
00038
00044 #define P2CONST(ptrtype) const ptrtype *
00045
00051 #define CONSTP2VAR(ptrtype) ptrtype * const
00052
00058 #define CONSTP2CONST(ptrtype) const ptrtype * const
00059
00065 #define P2FUNC(rettype, fctname) rettype (*fctname)
00066
00072 #ifndef CONST
00073 #define CONST(consttype) const consttype
00074 #endif
00075
00081 #ifndef STATIC
00082 #define STATIC static
00083 #endif
00084
00087 #endif /* COTS_LIB_LSTD_COMPILER_H_ */

```

7.75 COTS/LIB/LSTD_MCU_UTILITIES.h File Reference

This file contains the MCU utility macro-functions.

Macros

- `#define MY_MS_DELAY(T) do{ u32 Timer = (T * 500); while (Timer--) { asm ("nop"); } } while(0);`

7.75.1 Detailed Description

This file contains the MCU utility macro-functions.

Author

Ali El Bana

Version

1.0

Date

10/29/2022

Definition in file [LSTD_MCU_UTILITIES.h](#).

7.76 LSTD MCU UTILITIES.h

[Go to the documentation of this file.](#)

```
00001  
00009 #ifndef COTS_LIB_LSTD MCU UTILITIES_H_  
00010 #define COTS_LIB_LSTD MCU UTILITIES_H_  
00023 #define MY_MS_DELAY(T) do{ u32 Timer = (T * 500); while (Timer--) { asm ("nop"); } } while(0);  
00024  
00027 #endif /* COTS_LIB_LSTD MCU UTILITIES_H */
```

7.77 COTS/LIB/LSTD TYPES.h File Reference

This file contains the standard types.

Typedefs

- `typedef unsigned char bool_t`
- `typedef unsigned char u8_t`
- `typedef char c8_t`
- `typedef unsigned short int u16_t`
- `typedef unsigned int u32_t`
- `typedef unsigned long int u64_t`
- `typedef signed long int s64_t`
- `typedef signed char s8_t`
- `typedef signed short int s16_t`
- `typedef signed int s32_t`
- `typedef float f32_t`
- `typedef double f64_t`

7.77.1 Detailed Description

This file contains the standard types.

Author

Ali El Bana

Version

1.0

Date

10/29/2022

Definition in file [LSTD TYPES.h](#).

7.78 LSTD_TYPES.h

[Go to the documentation of this file.](#)

```
00001  
00009 #ifndef COTS_LIB_LSTD_TYPES_H_  
00010 #define COTS_LIB_LSTD_TYPES_H_  
00011  
00023 typedef unsigned char bool_t;  
00024  
00030 typedef unsigned char u8_t;  
00031  
00037 typedef char c8_t;  
00038  
00044 typedef unsigned short int u16_t;  
00045  
00051 typedef unsigned int u32_t;  
00052  
00058 typedef unsigned long int u64_t ;  
00059  
00065 typedef signed long int s64_t ;  
00066  
00072 typedef signed char s8_t;  
00073  
00079 typedef signed short int s16_t;  
00080  
00086 typedef signed int s32_t;  
00087  
00093 typedef float f32_t;  
00094  
00100 typedef double f64_t;  
00101  
00104 #endif /* COTS_LIB_LSTD_TYPES_H_ */
```

7.79 COTS/LIB/LSTD_VALUES.h File Reference

This file contains the standard values.

Macros

- #define TRUE (1)
- #define FALSE (0)
- #define NULL ((void*)0)
- #define INITIAL_ZERO (0)
- #define FLAG_SET (1)
- #define FLAG_CLEARED (0)
- #define RUN (1)
- #define STOP (0)
- #define PRESSED (1)
- #define RELEASED (0)
- #define SAME_STRING (0)
- #define DIFFERENT_STRING (1)

7.79.1 Detailed Description

This file contains the standard values.

Author

Ali El Bana

Version

1.0

Date

10/29/2022

Definition in file [LSTD_VALUES.h](#).

7.80 LSTD_VALUES.h

[Go to the documentation of this file.](#)

```
00001
00009 #ifndef COTS_LIB_LSTD_VALUES_H_
00010 #define COTS_LIB_LSTD_VALUES_H_
00011
00018 #ifndef TRUE
00024 #define TRUE (1)
00025 #endif /* ifndef TRUE */
00026
00027 #ifndef FALSE
00033 #define FALSE (0)
00034 #endif /* ifndef FALSE */
00035
00036 #ifndef NULL
00042 #define NULL ( (void*)0 )
00043 #endif /* ifndef NULL */
00044
00045 #ifndef NULL
00046 #define NULL ((P2VAR(void))0)
00047 #endif /* ifndef NULL */
00048
00049
00050 #ifndef INITIAL_ZERO
00056 #define INITIAL_ZERO (0)
00057 #endif /* ifndef INITIAL_ZERO */
00058
00059 #ifndef FLAG_SET
00065 #define FLAG_SET (1)
00066 #endif /* ifndef FLAG_SET */
00067
00068 #ifndef FLAG_CLEARED
00074 #define FLAG_CLEARED (0)
00075 #endif /* ifndef FLAG_CLEARED */
00076
00082 #ifndef RUN
00083 #define RUN (1)
00084 #endif /* ifndef RUN */
00085
00086
00092 #ifndef STOP
00093 #define STOP (0)
00094 #endif /* ifndef STOP */
00095
00101 #ifndef PRESSED
00102 #define PRESSED (1)
00103 #endif /* ifndef PRESSED */
00104
00110 #ifndef RELEASED
00111 #define RELEASED (0)
00112 #endif /* ifndef RELEASED */
00113
00119 #ifndef SAME_STRING
00120 #define SAME_STRING (0)
00121 #endif /* ifndef SAME_STRING */
00122
00128 #ifndef DIFFERENT_STRING
00129 #define DIFFERENT_STRING (1)
00130 #endif /* ifndef DIFFERENT_STRING */
00131
00132
00133
00136 #endif /* COTS_LIB_LSTD_VALUES_H_ */
```

7.81 ADC_config.h

```

00001 /* FILENAME: ADC_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 12/14/2022
00005 */
00006 #ifndef _ADC_config_H
00007 #define _ADC_config_H
00008
00009
00010 /***** ADC_CR1 configurations *****/
00011 /* ADC_CR1 configurations */
00012 /***** ADC_CR1 configurations *****/
00013
00014 /*options:
00015 *ENABLE
00016 *DISABLE
00017 */
00018 #define OVERRUN_INT_DISABLE
00019
00020 /***** */
00021
00022 /*options:
00023 *_12_BITS
00024 *_10_BITS
00025 *_8_BITS
00026 *_6_BITS
00027 */
00028 #define RESOLUTION _12_BITS
00029
00030 /***** */
00031
00032 /*options:
00033 *ENABLE
00034 *DISABLE
00035 */
00036 #define ANALOG_WDT_DISABLE
00037
00038 /***** */
00039
00040 /*options:
00041 *ENABLE
00042 *DISABLE
00043 */
00044 #define INJ_ANALOG_WDT NOTBYBASED
00045
00046 /***** */
00047
00048 /*options:
00049 *_1_CHANNEL
00050 *_2_CHANNELS
00051 *_3_CHANNELS
00052 *_4_CHANNELS
00053 *_5_CHANNELS
00054 *_6_CHANNELS
00055 *_7_CHANNELS
00056 *_8_CHANNELS
00057 */
00058 #define DISC_CHANNEL_NUM _1_CHANNEL
00059
00060 /***** */
00061
00062 /*options:
00063 *ENABLE
00064 *DISABLE
00065 */
00066 #define INJ_DISC_MODE DISABLE
00067
00068 /***** */
00069
00070 /*options:
00071 *ENABLE
00072 *DISABLE
00073 */
00074 #define DISC_MODE DISABLE
00075
00076 /***** */
00077
00078 /*options:
00079 *ENABLE
00080 *DISABLE
00081 */
00082 #define INJ_AUTO_MODE DISABLE
00083
00084 /***** */
00085

```

```
00086 /*options:  
00087 *ENABLE  
00088 *DISABLE  
00089 */  
00090 #define AWDSGL_MODE DISABLE  
00091  
00092 /*****  
00093  
00094 /*options:  
00095 *ENABLE  
00096 *DISABLE  
00097 */  
00098 #define SCAN_MODE DISABLE  
00099  
00100 /*****  
00101  
00102 /*options:  
00103 *ENABLE  
00104 *DISABLE  
00105 */  
00106 #define JEOCIE_MODE DISABLE  
00107  
00108 /*****  
00109  
00110 /*options:  
00111 *ENABLE  
00112 *DISABLE  
00113 */  
00114 #define AWDIE_MODE DISABLE  
00115  
00116 /*****  
00117  
00118 /*options:  
00119 *ENABLE  
00120 *DISABLE  
00121 */  
00122 #define EOCIE_MODE DISABLE  
00123  
00124 /*****  
00125  
00126 /*options:  
00127 *ZERO  
00128 *ONE  
00129 *THREE  
00130 *FOUR  
00131 *FIVE  
00132 *SIX  
00133 *SEVEN  
00134 *EIGHT  
00135 *NINE  
00136 *TEN  
00137 *ELEVEN  
00138 *TWELVE  
00139 *THERTEEN  
00140 *FOURTEEN  
00141 *FIFTEEN  
00142 *SIXTEEN  
00143 *SEVENTEEN  
00144 *EIGHTEEN  
00145 */  
00146 #define AWD_CHANNEL ZERO  
00147  
00148 /*****  
00149 /* ADC_CR2 configurations */  
00150 /*****  
00151  
00152 /*options:  
00153 *ENABLE  
00154 *DISABLE  
00155 */  
00156 #define REG_START_CONV ENABLE  
00157  
00158 /*****  
00159  
00160 /*options:  
00161 *DISABLE  
00162 *RISING  
00163 *FALLING  
00164 *ON_CHANGE  
00165 */  
00166 #define REG_EXTERNAL_TRIGGER DISABLE  
00167  
00168 /*****  
00169  
00170 /*options:  
00171 *T1_CC1  
00172 *T1_CC2
```

```
00173 *T1_CC3
00174 *T2_CC2
00175 *T2_CC3
00176 *T2_CC4
00177 *REG_T2_TRGO
00178 *T3_CC1
00179 *T3_TRGO
00180 *T4_CC4
00181 *T5_CC1
00182 *T5_CC2
00183 *T5_CC3
00184 *EXTI_LINE11
00185 */
00186 #define REG_EXT_EVENT_SRC EXTI_LINE11
00187
00188 /*****
00189
00190 /*options:
00191 *ENABLE
00192 *DISABLE
00193 */
00194 #define INJ_START_CONV DISABLE
00195
00196 /*****
00197
00198 /*options:
00199 *DISABLE
00200 *RISING
00201 *FALLING
00202 *ON_CHANGE
00203 */
00204 #define INJ_EXTERNAL_TRIGGER DISABLE
00205
00206 /*****
00207
00208 /*options:
00209 *T1_CC4
00210 *T1_TRGO
00211 *T2_CC1
00212 *INJ_T2_TRGO
00213 *T3_CC2
00214 *T3_CC4
00215 *T4_CC1
00216 *T4_CC2
00217 *T4_CC3
00218 *T4_TRGO
00219 *T5_CC4
00220 *T5_TRGO
00221 *EXTI_LINE15
00222 */
00223 #define INJ_EXT_EVENT_SRC EXTI_LINE15
00224
00225 /*****
00226
00227 /*options:
00228 *RIGHT
00229 *LEFT
00230 */
00231 #define DATA_ALIGN RIGHT
00232
00233 /*****
00234
00235 /*options:
00236 *SEQUENCE
00237 *REG_CONV
00238 */
00239 #define EOC_SELECTION REG_CONV
00240
00241 /*****
00242
00243 /*options:
00244 *ENABLE
00245 *DISABLE
00246 */
00247 #define DMA_MODE DISABLE
00248
00249 /*****
00250
00251 /*options:
00252 *ENABLE
00253 *DISABLE
00254 */
00255 #define DMA_DIS_SELECTION DISABLE
00256
00257 /*****
00258
00259 /*options:
```

```
00260 /*ENABLE
00261 *DISABLE
00262 */
00263 #define CONT_CONV ENABLE
00264
00265 /***** */
00266
00267 /*options:
00268 *ENABLE
00269 *DISABLE
00270 */
00271 #define ADC_ON ENABLE
00272
00273 /***** */
00274 /*           ADC_SMPR1 configurations      */
00275 /***** */
00276
00277 /*options:
00278 *_3_CYCLES
00279 *_15_CYCLES
00280 *_28_CYCLES
00281 *_56_CYCLES
00282 *_84_CYCLES
00283 *_112_CYCLES
00284 *_144_CYCLES
00285 *_480_CYCLES
00286 */
00287 #define SMP10 _3_CYCLES
00288
00289 /***** */
00290
00291 /*options:
00292 *_3_CYCLES
00293 *_15_CYCLES
00294 *_28_CYCLES
00295 *_56_CYCLES
00296 *_84_CYCLES
00297 *_112_CYCLES
00298 *_144_CYCLES
00299 *_480_CYCLES
00300 */
00301 #define SMP11 _3_CYCLES
00302
00303 /***** */
00304
00305 /*options:
00306 *_3_CYCLES
00307 *_15_CYCLES
00308 *_28_CYCLES
00309 *_56_CYCLES
00310 *_84_CYCLES
00311 *_112_CYCLES
00312 *_144_CYCLES
00313 *_480_CYCLES
00314 */
00315 #define SMP12 _3_CYCLES
00316
00317 /***** */
00318
00319 /*options:
00320 *_3_CYCLES
00321 *_15_CYCLES
00322 *_28_CYCLES
00323 *_56_CYCLES
00324 *_84_CYCLES
00325 *_112_CYCLES
00326 *_144_CYCLES
00327 *_480_CYCLES
00328 */
00329 #define SMP13 _3_CYCLES
00330
00331 /***** */
00332
00333 /*options:
00334 *_3_CYCLES
00335 *_15_CYCLES
00336 *_28_CYCLES
00337 *_56_CYCLES
00338 *_84_CYCLES
00339 *_112_CYCLES
00340 *_144_CYCLES
00341 *_480_CYCLES
00342 */
00343 #define SMP14 _3_CYCLES
00344
00345 /***** */
00346
```

```
00347 /*options:  
00348 *_3_CYCLES  
00349 *_15_CYCLES  
00350 *_28_CYCLES  
00351 *_56_CYCLES  
00352 *_84_CYCLES  
00353 *_112_CYCLES  
00354 *_144_CYCLES  
00355 *_480_CYCLES  
00356 */  
00357 #define SMP15 _3_CYCLES  
00358  
00359 /*****  
00360  
00361 /*options:  
00362 *_3_CYCLES  
00363 *_15_CYCLES  
00364 *_28_CYCLES  
00365 *_56_CYCLES  
00366 *_84_CYCLES  
00367 *_112_CYCLES  
00368 *_144_CYCLES  
00369 *_480_CYCLES  
00370 */  
00371 #define SMP16 _3_CYCLES  
00372  
00373 /*****  
00374  
00375 /*options:  
00376 *_3_CYCLES  
00377 *_15_CYCLES  
00378 *_28_CYCLES  
00379 *_56_CYCLES  
00380 *_84_CYCLES  
00381 *_112_CYCLES  
00382 *_144_CYCLES  
00383 *_480_CYCLES  
00384 */  
00385 #define SMP17 _3_CYCLES  
00386  
00387 /*****  
00388  
00389 /*options:  
00390 *_3_CYCLES  
00391 *_15_CYCLES  
00392 *_28_CYCLES  
00393 *_56_CYCLES  
00394 *_84_CYCLES  
00395 *_112_CYCLES  
00396 *_144_CYCLES  
00397 *_480_CYCLES  
00398 */  
00399 #define SMP18 _3_CYCLES  
00400  
00401 /*****  
00402 /* ADC_SMPR2 configurations */  
00403 /*****  
00404  
00405 /*options:  
00406 *_3_CYCLES  
00407 *_15_CYCLES  
00408 *_28_CYCLES  
00409 *_56_CYCLES  
00410 *_84_CYCLES  
00411 *_112_CYCLES  
00412 *_144_CYCLES  
00413 *_480_CYCLES  
00414 */  
00415 #define SMP0 _3_CYCLES  
00416  
00417 /*****  
00418  
00419 /*options:  
00420 *_3_CYCLES  
00421 *_15_CYCLES  
00422 *_28_CYCLES  
00423 *_56_CYCLES  
00424 *_84_CYCLES  
00425 *_112_CYCLES  
00426 *_144_CYCLES  
00427 *_480_CYCLES  
00428 */  
00429 #define SMP1 _3_CYCLES  
00430  
00431 /*****  
00432  
00433 /*options:
```

```
00434 *_3_CYCLEs
00435 *_15_CYCLEs
00436 *_28_CYCLEs
00437 *_56_CYCLEs
00438 *_84_CYCLEs
00439 *_112_CYCLEs
00440 *_144_CYCLEs
00441 *_480_CYCLEs
00442 */
00443 #define SMP2 _3_CYCLEs
00444
00445 /***** 
00446
00447 /*options:
00448 *_3_CYCLEs
00449 *_15_CYCLEs
00450 *_28_CYCLEs
00451 *_56_CYCLEs
00452 *_84_CYCLEs
00453 *_112_CYCLEs
00454 *_144_CYCLEs
00455 *_480_CYCLEs
00456 */
00457 #define SMP3 _3_CYCLEs
00458
00459 /***** 
00460
00461 /*options:
00462 *_3_CYCLEs
00463 *_15_CYCLEs
00464 *_28_CYCLEs
00465 *_56_CYCLEs
00466 *_84_CYCLEs
00467 *_112_CYCLEs
00468 *_144_CYCLEs
00469 *_480_CYCLEs
00470 */
00471 #define SMP4 _3_CYCLEs
00472
00473 /***** 
00474
00475 /*options:
00476 *_3_CYCLEs
00477 *_15_CYCLEs
00478 *_28_CYCLEs
00479 *_56_CYCLEs
00480 *_84_CYCLEs
00481 *_112_CYCLEs
00482 *_144_CYCLEs
00483 *_480_CYCLEs
00484 */
00485 #define SMP5 _3_CYCLEs
00486
00487 /***** 
00488
00489 /*options:
00490 *_3_CYCLEs
00491 *_15_CYCLEs
00492 *_28_CYCLEs
00493 *_56_CYCLEs
00494 *_84_CYCLEs
00495 *_112_CYCLEs
00496 *_144_CYCLEs
00497 *_480_CYCLEs
00498 */
00499 #define SMP6 _3_CYCLEs
00500
00501 /***** 
00502
00503 /*options:
00504 *_3_CYCLEs
00505 *_15_CYCLEs
00506 *_28_CYCLEs
00507 *_56_CYCLEs
00508 *_84_CYCLEs
00509 *_112_CYCLEs
00510 *_144_CYCLEs
00511 *_480_CYCLEs
00512 */
00513 #define SMP7 _3_CYCLEs
00514
00515 /***** 
00516
00517 /*options:
00518 *_3_CYCLEs
00519 *_15_CYCLEs
00520 *_28_CYCLEs
```

```
00521 *_56_CYCLES
00522 *_84_CYCLES
00523 *_112_CYCLES
00524 *_144_CYCLES
00525 *_480_CYCLES
00526 */
00527 #define SMP8 _3_CYCLES
00528
00529 /*****options: *****/
00530
00531 /*_3_CYCLES
00532 *_15_CYCLES
00533 *_28_CYCLES
00534 *_56_CYCLES
00535 *_84_CYCLES
00536 *_112_CYCLES
00537 *_144_CYCLES
00538 *_480_CYCLES
00539 */
00540 */
00541 #define SMP9 _3_CYCLES
00542
00543 /*****ADC_SQRL configurations *****/
00544 /*
00545 ADC_SQRL configurations */
00546
00547 /*options:
00548 *_1_CONV
00549 *_2_CONV
00550 *_3_CONV
00551 *_4_CONV
00552 *_5_CONV
00553 *_6_CONV
00554 *_7_CONV
00555 *_8_CONV
00556 *_9_CONV
00557 *_10_CONV
00558 *_11_CONV
00559 *_15_CONV
00560 *_16_CONV
00561 */
00562 #define REG_SO_LENGTH _1_CONV
00563
00564 /*****ADC_CCR configurations *****/
00565 /*
00566 ADC_CCR configurations */
00567
00568 /*options:
00569 *ENABLE
00570 *DISABLE
00571 */
00572 #define TEMP_SENSOR_AND_VREF DISABLE
00573
00574 /*****V_BATTERY DISABLE *****/
00575
00576 /*options:
00577 *ENABLE
00578 *DISABLE
00579 */
00580 #define V_BATTERY DISABLE
00581
00582 /*****Bit Manipulation Macros *****/
00583
00584 /*options:
00585 *DIV_BY_2
00586 *DIV_BY_4
00587 *DIV_BY_6
00588 *DIV_BY_8
00589 */
00590 #define ADC_PRESCALER DIV_BY_2
00591
00592 /*****Bit Manipulation Macros *****/
00593 /*
00594 Bit Manipulation Macros */
00595
00596 #define SQ1_BIT_MANIPULATION 0xFFFFFE0
00597
00598
00599
00600
00601
00602
00603 #endif //ADC_config_H
```

7.82 COTS/MCAL/ADC/ADC_interface.h File Reference

This file contains the interfacing information of ADC Module in STM32F103C8 ARM Microcontroller.

Macros

- #define CHANNEL0 0
Channel 0.
- #define CHANNEL1 1
Channel 1.
- #define CHANNEL2 2
Channel 2.
- #define CHANNEL3 3
Channel 3.
- #define CHANNEL4 4
Channel 4.
- #define CHANNEL5 5
Channel 5.
- #define CHANNEL6 6
Channel 6.
- #define CHANNEL7 7
Channel 7.
- #define CHANNEL8 8
Channel 8.
- #define CHANNEL9 9
Channel 9.
- #define CHANNEL10 10
Channel 10.
- #define CHANNEL11 11
Channel 11.
- #define CHANNEL12 12
Channel 12.
- #define CHANNEL13 13
Channel 13.
- #define CHANNEL14 14
Channel 14.
- #define CHANNEL15 15
Channel 15.
- #define CHANNEL16 16
Channel 16.
- #define CHANNEL17 17
Channel 17.
- #define CHANNEL18 18
Channel 18.

Functions

- void [MADC_vInit](#) (void)
This function is responsible for initializing the ADC module.
- u16_t [MADC_u16ConvertToDigital](#) (u8_t A_u8ChannelNum)
This function is responsible for converting the analog value to digital value.
- void [MADC_vDisable](#) (void)
This function is responsible for disabling the ADC module.
- void [MADC_vEnable](#) (void)
This function is responsible for enabling the ADC module.
- void [MADC_vRegINT_Disable](#) (void)
This function is responsible for disabling the ADC module interrupt.
- void [MADC_vRegINTEnable](#) (void)
This function is responsible for enabling the ADC module interrupt.
- void [MADC_vSelectChannel](#) (u8_t A_u8ChannelNum)
This function is responsible for selecting the channel of the ADC module.
- void [MADC_vSetCallBack](#) (void(*MEXTI_vpPointerTo_ISR_function)(void))
This function is responsible for setting the callback function of the ADC module.

7.82.1 Detailed Description

This file contains the interfacing information of ADC Module in STM32F103C8 ARM Microcontroller.

This file contains the prototyping of the APIs of ADC Module in STM32F103C8 ARM Microcontroller

Author

Ali El Bana

Version

1.0

Date

12/14/2022

Definition in file [ADC_interface.h](#).

7.82.2 Function Documentation

7.82.2.1 MADC_vInit()

```
void MADC_vInit (
    void )
```

This function is responsible for initializing the ADC module.

Definition at line 30 of file [ADC_program.c](#).

```
00031 {
00032
00033     // EN CLK on APB2 bus to be able to operate ADC peripheral:
00034     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00035
00036     // EN/DIS power for ADC to be turned on/off:
00037 #if ADC_ON == ENABLE
00038
00039     SET_BIT( ADC->CR2, ADON ) ;
00040
00041 #elif ADC_ON == DISABLE
00042
00043     CLR_BIT( ADC->CR2, ADON ) ;
00044
00045 #endif
00046
00047     // Select ADC prescalers(2/4/6/8):
00048 #if ADC_PRESCALER == DIV_BY_2
00049
00050     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00051     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00052
00053 #elif ADC_PRESCALER == DIV_BY_4
00054
00055     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00056     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00057
00058 #elif ADC_PRESCALER == DIV_BY_6
00059
00060     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00061     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00062
00063 #elif ADC_PRESCALER == DIV_BY_8
00064
00065     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00066     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00067
00068 #endif
00069
00070
00071     // Select ADC Resolution(12/10/8/6):
00072 #if RESOLUTION == _12_BITS
00073
00074     CLR_BIT( ADC->CR1, RES0 ) ;
00075     CLR_BIT( ADC->CR1, RES1 ) ;
00076
00077 #elif RESOLUTION == _10_BITS
00078
00079     SET_BIT( ADC->CR1, RES0 ) ;
00080     CLR_BIT( ADC->CR1, RES1 ) ;
00081
00082 #elif RESOLUTION == _8_BITS
00083
00084     CLR_BIT( ADC->CR1, RES0 ) ;
00085     SET_BIT( ADC->CR1, RES1 ) ;
00086
00087 #elif RESOLUTION == _6_BITS
00088
00089     SET_BIT( ADC->CR1, RES0 ) ;
00090     SET_BIT( ADC->CR1, RES1 ) ;
00091
00092 #endif
00093
00094
00095     // ADC Data Alignment(Right/Left):
00096 #if DATA_ALIGN == RIGHT
00097
00098     CLR_BIT( ADC->CR2, ALIGN ) ;
00099
00100 #elif DATA_ALIGN == LEFT
00101
00102     SET_BIT( ADC->CR2, ALIGN ) ;
00103
00104 #endif
00105
```

```
00106     // EN/DIS Injected INT:  
00107 #if JEOCIE_MODE == ENABLE  
00108     SET_BIT( ADC->CR1, JEOCIE ) ;  
00109  
00110 #elif JEOCIE_MODE == DISABLE  
00111     CLR_BIT( ADC->CR1, JEOCIE ) ;  
00112  
00113 #endif  
00114  
00115     // EN/DIS Regular INT:  
00116 #if EOCIE_MODE == ENABLE  
00117     SET_BIT( ADC->CR1, EOCIE ) ;  
00118  
00119 #elif EOCIE_MODE == DISABLE  
00120     CLR_BIT( ADC->CR1, EOCIE ) ;  
00121  
00122 #endif  
00123  
00124     // EN/DIS Overrun interrupt:  
00125 #if OVERRUN_INT == ENABLE  
00126     SET_BIT( ADC->CR1, OVRIE ) ;  
00127  
00128 #elif OVERRUN_INT == DISABLE  
00129     CLR_BIT( ADC->CR1, OVRIE ) ;  
00130  
00131 #endif  
00132  
00133     // Select Regular channel sequence length(1/2/3/.../16):  
00134 #if REG_SQ_LENGTH == _1_CONV  
00135     CLR_BIT( ADC->SQR1, L0 ) ;  
00136     CLR_BIT( ADC->SQR1, L1 ) ;  
00137     CLR_BIT( ADC->SQR1, L2 ) ;  
00138     CLR_BIT( ADC->SQR1, L3 ) ;  
00139  
00140 #elif REG_SQ_LENGTH == _2_CONV  
00141     SET_BIT( ADC->SQR1, L0 ) ;  
00142     CLR_BIT( ADC->SQR1, L1 ) ;  
00143     CLR_BIT( ADC->SQR1, L2 ) ;  
00144     CLR_BIT( ADC->SQR1, L3 ) ;  
00145  
00146 #elif REG_SQ_LENGTH == _3_CONV  
00147     CLR_BIT( ADC->SQR1, L0 ) ;  
00148     SET_BIT( ADC->SQR1, L1 ) ;  
00149     CLR_BIT( ADC->SQR1, L2 ) ;  
00150     CLR_BIT( ADC->SQR1, L3 ) ;  
00151  
00152 #elif REG_SQ_LENGTH == _4_CONV  
00153     SET_BIT( ADC->SQR1, L0 ) ;  
00154     SET_BIT( ADC->SQR1, L1 ) ;  
00155     CLR_BIT( ADC->SQR1, L2 ) ;  
00156     CLR_BIT( ADC->SQR1, L3 ) ;  
00157  
00158 #elif REG_SQ_LENGTH == _5_CONV  
00159     CLR_BIT( ADC->SQR1, L0 ) ;  
00160     CLR_BIT( ADC->SQR1, L1 ) ;  
00161     SET_BIT( ADC->SQR1, L2 ) ;  
00162     CLR_BIT( ADC->SQR1, L3 ) ;  
00163  
00164 #elif REG_SQ_LENGTH == _6_CONV  
00165     SET_BIT( ADC->SQR1, L0 ) ;  
00166     CLR_BIT( ADC->SQR1, L1 ) ;  
00167     SET_BIT( ADC->SQR1, L2 ) ;  
00168     CLR_BIT( ADC->SQR1, L3 ) ;  
00169  
00170 #elif REG_SQ_LENGTH == _7_CONV  
00171     CLR_BIT( ADC->SQR1, L0 ) ;  
00172     CLR_BIT( ADC->SQR1, L1 ) ;  
00173     SET_BIT( ADC->SQR1, L2 ) ;  
00174     CLR_BIT( ADC->SQR1, L3 ) ;  
00175  
00176 #elif REG_SQ_LENGTH == _8_CONV  
00177     SET_BIT( ADC->SQR1, L0 ) ;  
00178     CLR_BIT( ADC->SQR1, L1 ) ;  
00179     SET_BIT( ADC->SQR1, L2 ) ;  
00180     CLR_BIT( ADC->SQR1, L3 ) ;  
00181  
00182 #elif REG_SQ_LENGTH == _9_CONV  
00183     CLR_BIT( ADC->SQR1, L0 ) ;  
00184     SET_BIT( ADC->SQR1, L1 ) ;  
00185     CLR_BIT( ADC->SQR1, L2 ) ;  
00186     SET_BIT( ADC->SQR1, L3 ) ;  
00187  
00188 #endif  
00189  
00190 #endif  
00191     SET_BIT( ADC->SQR1, L0 ) ;
```

```

00193     SET_BIT( ADC->SQR1, L1 ) ;
00194     SET_BIT( ADC->SQR1, L2 ) ;
00195     CLR_BIT( ADC->SQR1, L3 ) ;
00196
00197 #elif REG_SO_LENGTH == _9_CONV
00198
00199     CLR_BIT( ADC->SQR1, L0 ) ;
00200     CLR_BIT( ADC->SQR1, L1 ) ;
00201     CLR_BIT( ADC->SQR1, L2 ) ;
00202     SET_BIT( ADC->SQR1, L3 ) ;
00203
00204 #elif REG_SO_LENGTH == _10_CONV
00205
00206     SET_BIT( ADC->SQR1, L0 ) ;
00207     CLR_BIT( ADC->SQR1, L1 ) ;
00208     CLR_BIT( ADC->SQR1, L2 ) ;
00209     SET_BIT( ADC->SQR1, L3 ) ;
00210
00211 #elif REG_SO_LENGTH == _11_CONV
00212
00213     CLR_BIT( ADC->SQR1, L0 ) ;
00214     SET_BIT( ADC->SQR1, L1 ) ;
00215     CLR_BIT( ADC->SQR1, L2 ) ;
00216     SET_BIT( ADC->SQR1, L3 ) ;
00217
00218 #elif REG_SO_LENGTH == _12_CONV
00219
00220     SET_BIT( ADC->SQR1, L0 ) ;
00221     SET_BIT( ADC->SQR1, L1 ) ;
00222     CLR_BIT( ADC->SQR1, L2 ) ;
00223     SET_BIT( ADC->SQR1, L3 ) ;
00224
00225 #elif REG_SO_LENGTH == _13_CONV
00226
00227     CLR_BIT( ADC->SQR1, L0 ) ;
00228     CLR_BIT( ADC->SQR1, L1 ) ;
00229     SET_BIT( ADC->SQR1, L2 ) ;
00230     SET_BIT( ADC->SQR1, L3 ) ;
00231
00232 #elif REG_SO_LENGTH == _14_CONV
00233
00234     SET_BIT( ADC->SQR1, L0 ) ;
00235     CLR_BIT( ADC->SQR1, L1 ) ;
00236     SET_BIT( ADC->SQR1, L2 ) ;
00237     SET_BIT( ADC->SQR1, L3 ) ;
00238
00239 #elif REG_SO_LENGTH == _15_CONV
00240
00241     CLR_BIT( ADC->SQR1, L0 ) ;
00242     SET_BIT( ADC->SQR1, L1 ) ;
00243     SET_BIT( ADC->SQR1, L2 ) ;
00244     SET_BIT( ADC->SQR1, L3 ) ;
00245
00246 #elif REG_SO_LENGTH == _16_CONV
00247
00248     SET_BIT( ADC->SQR1, L0 ) ;
00249     SET_BIT( ADC->SQR1, L1 ) ;
00250     SET_BIT( ADC->SQR1, L2 ) ;
00251     SET_BIT( ADC->SQR1, L3 ) ;
00252
00253 #endif
00254
00255 // EN/DIS SCAN MODE:
00256 #if SCAN_MODE == ENABLE
00257
00258     SET_BIT( ADC->CR1, SCAN ) ;
00259
00260 #elif SCAN_MODE == DISABLE
00261
00262     CLR_BIT( ADC->CR1, SCAN ) ;
00263
00264 #endif
00265
00266 // EN/DIS Discontinuous mode on Regular:
00267 #if DISC_MODE == ENABLE
00268
00269     SET_BIT( ADC->CR1, DISCEN ) ;
00270
00271 #elif DISC_MODE == DISABLE
00272
00273     CLR_BIT( ADC->CR1, DISCEN ) ;
00274
00275 #endif
00276
00277 // Discontinuous mode channel count(1/2/3/4/.../8):
00278 #if DISC_CHANNEL_NUM == _1_CHANNEL
00279

```

```

00280     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00281     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00282     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00283
00284 #elif DISC_CHANNEL_NUM == _2_CHANNELS
00285
00286     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00287     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00288     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00289
00290 #elif DISC_CHANNEL_NUM == _3_CHANNELS
00291
00292     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00293     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00294     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00295
00296 #elif DISC_CHANNEL_NUM == _4_CHANNELS
00297
00298     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00299     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00300     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00301
00302 #elif DISC_CHANNEL_NUM == _5_CHANNELS
00303
00304     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00305     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00306     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00307
00308 #elif DISC_CHANNEL_NUM == _6_CHANNELS
00309
00310     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00311     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00312     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00313
00314 #elif DISC_CHANNEL_NUM == _7_CHANNELS
00315
00316     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00317     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00318     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00319
00320 #elif DISC_CHANNEL_NUM == _8_CHANNELS
00321
00322     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00323     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00324     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00325
00326
00327 #endif
00328
00329 // EN/DIS Continuous mode:
00330 #if CONT_CONV == ENABLE
00331
00332     SET_BIT( ADC->CR2, CONT ) ;
00333
00334 #elif CONT_CONV == DISABLE
00335
00336     CLR_BIT( ADC->CR2, CONT ) ;
00337
00338 #endif
00339
00340 }

```

References [APB2ENR_ADC1EN](#), [CLR_BIT](#), [MADC_vInit\(\)](#), [MRCC_vEnablePeriphralCLK\(\)](#), [RCC_APB2](#), and [SET_BIT](#).

Referenced by [MADC_vInit\(\)](#).

7.82.2.2 MADC_u16ConvertToDigital()

```

u16_t MADC_u16ConvertToDigital (
    u8_t A_u8ChannelNum )

```

This function is responsible for converting the analog value to digital value.

Returns

`u16_t` The digital value of the analog value

Definition at line 356 of file [ADC_program.c](#).

```
00357 {
00358
00359     // Select Regular channel(1/2/3/.../16)
00360     ADC->SQR3 = (ADC->SQR3 & SQ1_BIT_MANIPULATION) | ( A_u8ChannelNum ) ;
00361
00362     // Start conversion of regular channels:
00363     SET_BIT( ADC->CR2, SWSTART ) ;
00364
00365     // Poll on the end of conversion flag until it's raised:
00366     while( GET_BIT(ADC->SR, EOC) != FLAG_SET ) ;
00367
00368     // Clear the flag:
00369     CLR_BIT( ADC->SR, EOC ) ;
00370
00371     // Return ADC Data:
00372     return ADC->DR ;
00373
00374 }
```

References [CLR_BIT](#), [FLAG_SET](#), [GET_BIT](#), [MADC_u16ConvertToDigital\(\)](#), and [SET_BIT](#).

Referenced by [MADC_u16ConvertToDigital\(\)](#).

7.82.2.3 MADC_vDisable()

```
void MADC_vDisable (
    void )
```

This function is responsible for disabling the ADC module.

Definition at line 379 of file [ADC_program.c](#).

```
00380 {
00381
00382     // DIS CLK on APB2 bus to be able to operate ADC peripheral:
00383     MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00384
00385     // Power off ADC:
00386     CLR_BIT( ADC->CR2, ADON ) ;
00387
00388 }
```

References [APB2ENR_ADC1EN](#), [CLR_BIT](#), [MADC_vDisable\(\)](#), [MRCC_vDisablePeriphralCLK\(\)](#), and [RCC_APB2](#).

Referenced by [MADC_vDisable\(\)](#).

7.82.2.4 MADC_vEnable()

```
void MADC_vEnable (
    void )
```

This function is responsible for enabling the ADC module.

Definition at line 393 of file [ADC_program.c](#).

```
00394 {
00395
00396     // EN CLK on APB2 bus to be able to operate ADC peripheral:
00397     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00398
00399     // Power on ADC:
00400     SET_BIT( ADC->CR2, ADON ) ;
00401
00402 }
```

References [APB2ENR_ADC1EN](#), [MADC_vEnable\(\)](#), [MRCC_vEnablePeriphralCLK\(\)](#), [RCC_APB2](#), and [SET_BIT](#).

Referenced by [MADC_vEnable\(\)](#).

7.82.2.5 MADC_vRegINT_Disable()

```
void MADC_vRegINT_Disable (
    void )
```

This function is responsible for disabling the ADC module interrupt.

Definition at line 407 of file [ADC_program.c](#).

```
00408 {
00409     CLR_BIT( ADC->CR1, EOCIE ) ;
00410 }
```

References [CLR_BIT](#), and [MADC_vRegINT_Disable\(\)](#).

Referenced by [MADC_vRegINT_Disable\(\)](#).

7.82.2.6 MADC_vRegINTEnable()

```
void MADC_vRegINTEnable (
    void )
```

This function is responsible for enabling the ADC module interrupt.

Definition at line 415 of file [ADC_program.c](#).

```
00416 {
00417     SET_BIT( ADC->CR1, EOCIE ) ;
00418 }
```

References [MADC_vRegINTEnable\(\)](#), and [SET_BIT](#).

Referenced by [MADC_vRegINTEnable\(\)](#).

7.82.2.7 MADC_vSelectChannel()

```
void MADC_vSelectChannel (
    u8_t A_u8ChannelNum )
```

This function is responsible for selecting the channel of the ADC module.

Definition at line 345 of file [ADC_program.c](#).

```
00346 {
00347
00348     // Select Regular channel(1/2/3/.../16)
00349     ADC->SQR3 = (ADC->SQR3 & SQ1_BIT_MANIPULATION) | ( A_u8ChannelNum ) ;
00350
00351 }
```

References [MADC_vSelectChannel\(\)](#).

Referenced by [MADC_vSelectChannel\(\)](#).

7.82.2.8 MADC_vSetCallBack()

```
void MADC_vSetCallBack (
    void(*)(void) MEXTI_vpPointerTo_ISR_function )
```

This function is responsible for setting the callback function of the ADC module.

Parameters

in	<i>MEXTI_vpPointerTo_ISR_function</i>	A pointer to the callback function
----	---------------------------------------	------------------------------------

Definition at line 423 of file [ADC_program.c](#).

```
00424 {
00425
00426     if( MEXTI_vpPointerTo_ISR_function != NULL )
00427     {
00428         MADC_vpPointerToFunction = MEXTI_vpPointerTo_ISR_function ;
00429     }
00430 }
00431 }
00432 }
```

References [MADC_vSetCallBack\(\)](#), and [NULL](#).

Referenced by [MADC_vSetCallBack\(\)](#).

7.83 ADC_interface.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef _ADC_interface_H
00012 #define _ADC_interface_H
00013
00014 /***** Functions prototypes ****/
00015 /*          Functions prototypes          */
00016 /***** ****/
00017
00022 void MADC_vInit(void);
00023
00030 u16_t MADC_u16ConvertToDigital(u8_t A_u8ChannelNum);
00031
00036 void MADC_vDisable(void);
00037
00042 void MADC_vEnable(void);
00043
00048 void MADC_vRegINT_Disable(void);
00049
00054 void MADC_vRegINTEnable(void);
00055
00060 void MADC_vSelectChannel(u8_t A_u8ChannelNum);
00061
00067 void MADC_vSetCallBack(void (*MEXTI_vpPointerTo_ISR_function)(void));
00068
00069 /***** Interfacing macros ****/
00070 /*          Interfacing macros          */
00071 /***** ****/
00072
00083 #define CHANNEL0 0
00084
00090 #define CHANNEL1 1
00091
00097 #define CHANNEL2 2
00098
00104 #define CHANNEL3 3
00105
00111 #define CHANNEL4 4
00112
00118 #define CHANNEL5 5
00119
00125 #define CHANNEL6 6
00126
00132 #define CHANNEL7 7
00133
00139 #define CHANNEL8 8
00140
00146 #define CHANNEL9 9
00147
00153 #define CHANNEL10 10
00154
00160 #define CHANNEL11 11
00161
```

```
00167 #define CHANNEL12 12
00168
00174 #define CHANNEL13 13
00175
00181 #define CHANNEL14 14
00182
00188 #define CHANNEL15 15
00189
00195 #define CHANNEL16 16
00196
00202 #define CHANNEL17 17
00203
00209 #define CHANNEL18 18
00210
00213 #endif // _ADC_interface_H
```

7.84 ADC_private.h

```
00001 /* FILENAME: ADC_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 12/14/2022
00005 */
00006 #ifndef _ADC_private_H
00007 #define _ADC_private_H
00008
00009 // TODO: Review the comments and modify them.
00010
00011 /***** Peripherals declaration ****/
00012 /*
00013 *****/
00014
00020 typedef struct
00021 {
00022
00026     u32_t SR;
00027
00031     u32_t CR1;
00032
00036     u32_t CR2;
00037
00041     u32_t SMPR1;
00042
00046     u32_t SMPR2;
00047
00051     u32_t JOFR1;
00052
00056     u32_t JOFR2;
00057
00061     u32_t JOFR3;
00062
00066     u32_t JOFR4;
00067
00071     u32_t HTR;
00072
00076     u32_t LTR;
00077
00081     u32_t SQR1;
00082
00086     u32_t SQR2;
00087
00091     u32_t SQR3;
00092
00096     u32_t JSQR;
00097
00101     u32_t JDR1;
00102
00106     u32_t JDR2;
00107
00111     u32_t JDR3;
00112
00116     u32_t JDR4;
00117
00121     u32_t DR;
00122
00126     u32_t CCR;
00127
00128
00129 } ADC_MemoryMapType;
00130
00131
00132 #define ADC_BASE_ADDRESS 0x40012000
00133
00134
00135
00136
00137
00138
```

```

00144 #define ADC ( (volatile P2VAR(ADC_MemoryMapType) ) (ADC_BASE_ADDRESS) ) // Is a pointer to the struct.
00145
00146
00147 /****** ADC_SR register bits *****/
00148 /* ADC_SR register bits */
00149 /****** ADC_SR register bits *****/
00150
00151 #define OVR      5
00152 #define STRT     4
00153 #define JSTRT    3
00154 #define JEOC      2
00155 #define EOC       1
00156 #define AWD       0
00157
00158 /****** ADC_CRL register bits *****/
00159 /* ADC_CRL register bits */
00160 /****** ADC_CRL register bits *****/
00161
00162 #define OVRIE     26
00163 #define RES0      24
00164 #define RES1      25
00165 #define AWDEN     23
00166 #define JAWDEN    22
00167 #define DISCNUM0   13
00168 #define DISCNUM1   14
00169 #define DISCNUM2   15
00170 #define JDISCEN   13
00171 #define DISCEN    11
00172 #define JAUTO     10
00173 #define AWD GSL  9
00174 #define SCAN      8
00175 #define JEOCIE    7
00176 #define AWDIE     6
00177 #define EOCIE     5
00178 #define AWDCH0    0
00179 #define AWDCH1    1
00180 #define AWDCH2    2
00181 #define AWDCH3    3
00182 #define AWDCH4    4
00183
00184 /****** ADC_CR2 register bits *****/
00185 /* ADC_CR2 register bits */
00186 /****** ADC_CR2 register bits *****/
00187
00188 #define SWSTART   30
00189 #define EXTENO    28
00190 #define EXTN1     29
00191 #define EXTSEL0   24
00192 #define EXTSEL1   25
00193 #define EXTSEL2   26
00194 #define EXTSEL3   27
00195 #define JSWSTART  22
00196 #define JEXTENO   20
00197 #define JEXTEN1  21
00198 #define JEXTSEL0  16
00199 #define JEXTSEL1  17
00200 #define JEXTSEL2  18
00201 #define JEXTSEL3  19
00202 #define ALIGN     11
00203 #define EOCS      10
00204 #define DDS       9
00205 #define DMA       8
00206 #define CONT     1
00207 #define ADON     0
00208
00209 /****** ADC_SMPR1 register bits *****/
00210 /* ADC_SMPR1 register bits */
00211 /****** ADC_SMPR1 register bits *****/
00212
00213 #define SMP10_b0  0
00214 #define SMP10_b1  1
00215 #define SMP10_b2  2
00216
00217 #define SMP11_b0  3
00218 #define SMP11_b1  4
00219 #define SMP11_b2  5
00220
00221 #define SMP12_b0  6
00222 #define SMP12_b1  7
00223 #define SMP12_b2  8
00224
00225 #define SMP13_b0  9
00226 #define SMP13_b1 10
00227 #define SMP13_b2 11
00228
00229 #define SMP14_b0 12
00230 #define SMP14_b1 13

```

```
00231 #define SMP14_b2    14
00232
00233 #define SMP15_b0    15
00234 #define SMP15_b1    16
00235 #define SMP15_b2    17
00236
00237 #define SMP15_b0    15
00238 #define SMP15_b1    16
00239 #define SMP15_b2    17
00240
00241 #define SMP16_b0    18
00242 #define SMP16_b1    19
00243 #define SMP16_b2    20
00244
00245 #define SMP17_b0    21
00246 #define SMP17_b1    22
00247 #define SMP17_b2    23
00248
00249 #define SMP18_b0    24
00250 #define SMP18_b1    25
00251 #define SMP18_b2    26
00252
00253 /***** ADC_SMPR2 register bits *****/
00254 /*                                         */
00255 /***** ADC_SMPR2 register bits *****/
00256
00257 #define SMP0_b0 0
00258 #define SMP0_b1 1
00259 #define SMP0_b2 2
00260
00261 #define SMP1_b0 3
00262 #define SMP1_b1 4
00263 #define SMP1_b2 5
00264
00265 #define SMP2_b0 6
00266 #define SMP2_b1 7
00267 #define SMP2_b2 8
00268
00269 #define SMP3_b0 9
00270 #define SMP3_b1 10
00271 #define SMP3_b2 11
00272
00273 #define SMP4_b0 12
00274 #define SMP4_b1 13
00275 #define SMP4_b2 14
00276
00277 #define SMP5_b0 15
00278 #define SMP5_b1 16
00279 #define SMP5_b2 17
00280
00281 #define SMP5_b0 15
00282 #define SMP5_b1 16
00283 #define SMP5_b2 17
00284
00285 #define SMP6_b0 18
00286 #define SMP6_b1 19
00287 #define SMP6_b2 20
00288
00289 #define SMP7_b0 21
00290 #define SMP7_b1 22
00291 #define SMP7_b2 23
00292
00293 #define SMP8_b0 24
00294 #define SMP8_b1 25
00295 #define SMP8_b2 26
00296
00297 #define SMP9_b0 27
00298 #define SMP9_b1 28
00299 #define SMP9_b2 29
00300
00301 /***** ADC_SQR1 register bits *****/
00302 /*                                         */
00303 /***** ADC_SQR1 register bits *****/
00304
00305 #define L0      20
00306 #define L1      21
00307 #define L2      22
00308 #define L3      23
00309
00310 #define SQ16_b0 15
00311 #define SQ16_b1 16
00312 #define SQ16_b2 17
00313 #define SQ16_b3 18
00314 #define SQ16_b4 19
00315
00316 #define SQ15_b0 10
00317 #define SQ15_b1 11
```

```
00318 #define SQ15_b2 12
00319 #define SQ15_b3 13
00320 #define SQ15_b4 14
00321
00322 #define SQ14_b0 5
00323 #define SQ14_b1 6
00324 #define SQ14_b2 7
00325 #define SQ14_b3 8
00326 #define SQ14_b4 9
00327
00328 #define SQ13_b0 0
00329 #define SQ13_b1 1
00330 #define SQ13_b2 2
00331 #define SQ13_b3 3
00332 #define SQ13_b4 4
00333
00334 /***** ADC_SQR2 register bits *****/
00335 /*
00336 *****/
00337
00338 #define SQ12_b0 25
00339 #define SQ12_b1 26
00340 #define SQ12_b2 27
00341 #define SQ12_b3 28
00342 #define SQ12_b4 29
00343
00344 #define SQ11_b0 20
00345 #define SQ11_b1 21
00346 #define SQ11_b2 22
00347 #define SQ11_b3 23
00348 #define SQ11_b4 24
00349
00350 #define SQ10_b0 15
00351 #define SQ10_b1 16
00352 #define SQ10_b2 17
00353 #define SQ10_b3 18
00354 #define SQ10_b4 19
00355
00356 #define SQ9_b0 10
00357 #define SQ9_b1 11
00358 #define SQ9_b2 12
00359 #define SQ9_b3 13
00360 #define SQ9_b4 14
00361
00362 #define SQ8_b0 5
00363 #define SQ8_b1 6
00364 #define SQ8_b2 7
00365 #define SQ8_b3 8
00366 #define SQ8_b4 9
00367
00368 #define SQ7_b0 0
00369 #define SQ7_b1 1
00370 #define SQ7_b2 2
00371 #define SQ7_b3 3
00372 #define SQ7_b4 4
00373
00374 /***** ADC_SQR3 register bits *****/
00375 /*
00376 *****/
00377
00378 #define SQ6_b0 25
00379 #define SQ6_b1 26
00380 #define SQ6_b2 27
00381 #define SQ6_b3 28
00382 #define SQ6_b4 29
00383
00384 #define SQ5_b0 20
00385 #define SQ5_b1 21
00386 #define SQ5_b2 22
00387 #define SQ5_b3 23
00388 #define SQ5_b4 24
00389
00390 #define SQ4_b0 15
00391 #define SQ4_b1 16
00392 #define SQ4_b2 17
00393 #define SQ4_b3 18
00394 #define SQ4_b4 19
00395
00396 #define SQ3_b0 10
00397 #define SQ3_b1 11
00398 #define SQ3_b2 12
00399 #define SQ3_b3 13
00400 #define SQ3_b4 14
00401
00402 #define SQ2_b0 5
00403 #define SQ2_b1 6
00404 #define SQ2_b2 7
```

```
00405 #define SQ2_b3 8
00406 #define SQ2_b4 9
00407
00408 #define SQ1_b0 0
00409 #define SQ1_b1 1
00410 #define SQ1_b2 2
00411 #define SQ1_b3 3
00412 #define SQ1_b4 4
00413
00414 /***** ADC_CCR register bits *****/
00415 /*          Config.h Macros          */
00416 /***** ADC_CCR register bits *****/
00417
00418 #define TSVREFE 23
00419 #define VBATB 22
00420 #define ADCPRE0 16
00421 #define ADCPRE1 17
00422
00423 /***** ADC_CCR register bits *****/
00424 /*          Config.h Macros          */
00425 /***** ADC_CCR register bits *****/
00426
00427 #define ENABLE 1
00428 #define DISABLE 2
00429
00430 #define _12_BITS 0
00431 #define _10_BITS 1
00432 #define _8_BITS 2
00433 #define _6_BITS 3
00434
00435 #define _1_CHANNEL 0
00436 #define _2_CHANNELS 1
00437 #define _3_CHANNELS 2
00438 #define _4_CHANNELS 3
00439 #define _5_CHANNELS 4
00440 #define _6_CHANNELS 5
00441 #define _7_CHANNELS 6
00442 #define _8_CHANNELS 7
00443
00444 #define ZERO 0
00445 #define ONE 1
00446 #define THREE 2
00447 #define FOUR 3
00448 #define FIVE 4
00449 #define SIX 5
00450 #define SEVEN 6
00451 #define EIGHT 7
00452 #define NINE 8
00453 #define TEN 9
00454 #define ELEVEN 10
00455 #define TWELVE 11
00456 #define THIRTEEN 12
00457 #define FOURTEEN 13
00458 #define FIFTEEN 14
00459 #define SIXTEEN 15
00460 #define SEVENTEEN 16
00461 #define EIGHTEEN 17
00462
00463 #define T1_CC1 0
00464 #define T1_CC2 1
00465 #define T1_CC3 2
00466 #define T2_CC2 3
00467 #define T2_CC3 4
00468 #define T2_CC4 5
00469 #define REG_T2_TRGO 6
00470 #define T3_CC1 7
00471 #define T3_TRGO 8
00472 #define T4_CC4 9
00473 #define T5_CC1 10
00474 #define T5_CC2 11
00475 #define T5_CC3 12
00476 #define EXTI_LINE11 15
00477
00478 #define RISING 1
00479 #define FALLING 2
00480 #define ON_CHANGE 3
00481
00482 #define T1_CC4 0
00483 #define T1_TRGO 1
00484 #define T2_CC1 2
00485 #define INJ_T2_TRGO 3
00486 #define T3_CC2 4
00487 #define T3_CC4 5
00488 #define T4_CC1 6
00489 #define T4_CC2 7
00490 #define T4_CC3 8
00491 #define T4_TRGO 9
```

```

00492 #define T5_CC4          10
00493 #define T5_TRGO         11
00494 #define EXTI_LINE15    15
00495
00496 #define RIGHT   0
00497 #define LEFT    1
00498
00499 #define _3_CYCLES     0
00500 #define _15_CYCLES    1
00501 #define _28_CYCLES    2
00502 #define _56_CYCLES    3
00503 #define _84_CYCLES    4
00504 #define _112_CYCLES   5
00505 #define _144_CYCLES   6
00506 #define _480_CYCLES   7
00507
00508 #define _1_CONV      0
00509 #define _2_CONV      1
00510 #define _3_CONV      2
00511 #define _4_CONV      3
00512 #define _5_CONV      4
00513 #define _6_CONV      5
00514 #define _7_CONV      6
00515 #define _8_CONV      7
00516 #define _9_CONV      8
00517 #define _10_CONV     9
00518 #define _11_CONV     10
00519 #define _15_CONV     11
00520 #define _16_CONV     15
00521
00522
00523 #define DIV_BY_2      0
00524 #define DIV_BY_4      1
00525 #define DIV_BY_6      2
00526 #define DIV_BY_8      3
00527
00528
00529
00530
00531
00532
00533
00534 #endif // _ADC_private_H

```

7.85 ADC_program.c

```

00001 /* FILENAME: ADC_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 12/14/2022
00005 */
00006
00007 /*****+
00008 /*           Include headers           */
00009 /*****+
00010 #include "../../LIB/LSTD_TYPES.h"
00011 #include "../../LIB/LSTD_COMPILER.h"
00012 #include "../../LIB/LSTD_VALUES.h"
00013 #include "../../LIB/LSTD_BITMATH.h"
00014
00015 #include "../RCC/MRCC_interface.h"
00016
00017 #include "ADC_interface.h"
00018 #include "ADC_private.h"
00019 #include "ADC_config.h"
00020
00021
00022 // Global ISR pointer to function.
00023 void ( *MADC_vpPointerToFunction ) (void) = NULL ;
00024
00025
00026 /*****+
00027 /*           Functions implementations           */
00028 /*****+
00029
00030 void MADC_vInit( void )
00031 {
00032
00033     // EN CLK on APB2 bus to be able to operate ADC peripheral:
00034     MRCC_vEnablePeripheralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00035
00036     // EN/DIS power for ADC to be turned on/off:
00037 #if ADC_ON == ENABLE

```

```
00038
00039     SET_BIT( ADC->CR2, ADON ) ;
00040
00041 #elif ADC_ON == DISABLE
00042     CLR_BIT( ADC->CR2, ADON ) ;
00043
00044 #endif
00045
00046 // Select ADC prescalers(2/4/6/8):
00047 #if ADC_PRESCALER == DIV_BY_2
00048     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00049     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00050
00051 #elif ADC_PRESCALER == DIV_BY_4
00052     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00053     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00054
00055 #elif ADC_PRESCALER == DIV_BY_6
00056     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00057     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00058
00059 #elif ADC_PRESCALER == DIV_BY_8
00060     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00061     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00062
00063 #endif
00064
00065 // Select ADC Resolution(12/10/8/6):
00066 #if RESOLUTION == _12_BITS
00067     CLR_BIT( ADC->CR1, RES0 ) ;
00068     CLR_BIT( ADC->CR1, RES1 ) ;
00069
00070 #elif RESOLUTION == _10_BITS
00071     SET_BIT( ADC->CR1, RES0 ) ;
00072     CLR_BIT( ADC->CR1, RES1 ) ;
00073
00074 #elif RESOLUTION == _8_BITS
00075     CLR_BIT( ADC->CR1, RES0 ) ;
00076     SET_BIT( ADC->CR1, RES1 ) ;
00077
00078 #elif RESOLUTION == _6_BITS
00079     SET_BIT( ADC->CR1, RES0 ) ;
00080     SET_BIT( ADC->CR1, RES1 ) ;
00081
00082 #endif
00083
00084 // ADC Data Alignment(Right/Left):
00085 #if DATA_ALIGN == RIGHT
00086     CLR_BIT( ADC->CR2, ALIGN ) ;
00087
00088 #elif DATA_ALIGN == LEFT
00089     SET_BIT( ADC->CR2, ALIGN ) ;
00090
00091 #endif
00092
00093
00094 // EN/DIS Injected INT:
00095 #if JEOCIE_MODE == ENABLE
00096     SET_BIT( ADC->CR1, JEOCIE ) ;
00097
00098 #elif JEOCIE_MODE == DISABLE
00099     CLR_BIT( ADC->CR1, JEOCIE ) ;
00100
00101 #endif
00102
00103
00104 // EN/DIS Regular INT:
00105 #if EOCIE_MODE == ENABLE
00106     SET_BIT( ADC->CR1, EOCIE ) ;
00107
00108 #elif EOCIE_MODE == DISABLE
00109     CLR_BIT( ADC->CR1, EOCIE ) ;
00110
00111 #endif
00112
00113
00114 // EN/DIS Software INT:
00115 #if SWOCIE_MODE == ENABLE
00116     SET_BIT( ADC->CR1, SWOCIE ) ;
00117
00118 #elif SWOCIE_MODE == DISABLE
00119     CLR_BIT( ADC->CR1, SWOCIE ) ;
00120
00121 #endif
00122
00123 // EN/DIS Software INT:
00124 #if SWOCIE_MODE == ENABLE
00125     SET_BIT( ADC->CR1, SWOCIE ) ;
00126
00127 #elif SWOCIE_MODE == DISABLE
00128     CLR_BIT( ADC->CR1, SWOCIE ) ;
00129
00130 #endif
```

```

00125     CLR_BIT( ADC->CR1, EOCIE ) ;
00126
00127 #endif
00128
00129 // EN/DIS Overrun interrupt:
00130 #if OVERRUN_INT == ENABLE
00131
00132     SET_BIT( ADC->CR1, OVRIE ) ;
00133
00134 #elif OVERRUN_INT == DISABLE
00135
00136     CLR_BIT( ADC->CR1, OVRIE ) ;
00137
00138 #endif
00139
00140 // Select Regular channel sequence length(1/2/3/.../16):
00141 #if REG_SO_LENGTH == _1_CONV
00142
00143     CLR_BIT( ADC->SQR1, L0 ) ;
00144     CLR_BIT( ADC->SQR1, L1 ) ;
00145     CLR_BIT( ADC->SQR1, L2 ) ;
00146     CLR_BIT( ADC->SQR1, L3 ) ;
00147
00148 #elif REG_SO_LENGTH == _2_CONV
00149
00150     SET_BIT( ADC->SQR1, L0 ) ;
00151     CLR_BIT( ADC->SQR1, L1 ) ;
00152     CLR_BIT( ADC->SQR1, L2 ) ;
00153     CLR_BIT( ADC->SQR1, L3 ) ;
00154
00155 #elif REG_SO_LENGTH == _3_CONV
00156
00157     CLR_BIT( ADC->SQR1, L0 ) ;
00158     SET_BIT( ADC->SQR1, L1 ) ;
00159     CLR_BIT( ADC->SQR1, L2 ) ;
00160     CLR_BIT( ADC->SQR1, L3 ) ;
00161
00162 #elif REG_SO_LENGTH == _4_CONV
00163
00164     SET_BIT( ADC->SQR1, L0 ) ;
00165     SET_BIT( ADC->SQR1, L1 ) ;
00166     CLR_BIT( ADC->SQR1, L2 ) ;
00167     CLR_BIT( ADC->SQR1, L3 ) ;
00168
00169 #elif REG_SO_LENGTH == _5_CONV
00170
00171     CLR_BIT( ADC->SQR1, L0 ) ;
00172     CLR_BIT( ADC->SQR1, L1 ) ;
00173     SET_BIT( ADC->SQR1, L2 ) ;
00174     CLR_BIT( ADC->SQR1, L3 ) ;
00175
00176 #elif REG_SO_LENGTH == _6_CONV
00177
00178     SET_BIT( ADC->SQR1, L0 ) ;
00179     CLR_BIT( ADC->SQR1, L1 ) ;
00180     SET_BIT( ADC->SQR1, L2 ) ;
00181     CLR_BIT( ADC->SQR1, L3 ) ;
00182
00183 #elif REG_SO_LENGTH == _7_CONV
00184
00185     CLR_BIT( ADC->SQR1, L0 ) ;
00186     SET_BIT( ADC->SQR1, L1 ) ;
00187     SET_BIT( ADC->SQR1, L2 ) ;
00188     CLR_BIT( ADC->SQR1, L3 ) ;
00189
00190 #elif REG_SO_LENGTH == _8_CONV
00191
00192     SET_BIT( ADC->SQR1, L0 ) ;
00193     SET_BIT( ADC->SQR1, L1 ) ;
00194     SET_BIT( ADC->SQR1, L2 ) ;
00195     CLR_BIT( ADC->SQR1, L3 ) ;
00196
00197 #elif REG_SO_LENGTH == _9_CONV
00198
00199     CLR_BIT( ADC->SQR1, L0 ) ;
00200     CLR_BIT( ADC->SQR1, L1 ) ;
00201     CLR_BIT( ADC->SQR1, L2 ) ;
00202     SET_BIT( ADC->SQR1, L3 ) ;
00203
00204 #elif REG_SO_LENGTH == _10_CONV
00205
00206     SET_BIT( ADC->SQR1, L0 ) ;
00207     CLR_BIT( ADC->SQR1, L1 ) ;
00208     CLR_BIT( ADC->SQR1, L2 ) ;
00209     SET_BIT( ADC->SQR1, L3 ) ;
00210
00211 #elif REG_SO_LENGTH == _11_CONV

```

```

00212
00213     CLR_BIT( ADC->SQR1, L0 ) ;
00214     SET_BIT( ADC->SQR1, L1 ) ;
00215     CLR_BIT( ADC->SQR1, L2 ) ;
00216     SET_BIT( ADC->SQR1, L3 ) ;
00217
00218 #elif REG_SO_LENGTH == _12_CONV
00219
00220     SET_BIT( ADC->SQR1, L0 ) ;
00221     SET_BIT( ADC->SQR1, L1 ) ;
00222     CLR_BIT( ADC->SQR1, L2 ) ;
00223     SET_BIT( ADC->SQR1, L3 ) ;
00224
00225 #elif REG_SO_LENGTH == _13_CONV
00226
00227     CLR_BIT( ADC->SQR1, L0 ) ;
00228     CLR_BIT( ADC->SQR1, L1 ) ;
00229     SET_BIT( ADC->SQR1, L2 ) ;
00230     SET_BIT( ADC->SQR1, L3 ) ;
00231
00232 #elif REG_SO_LENGTH == _14_CONV
00233
00234     SET_BIT( ADC->SQR1, L0 ) ;
00235     CLR_BIT( ADC->SQR1, L1 ) ;
00236     SET_BIT( ADC->SQR1, L2 ) ;
00237     SET_BIT( ADC->SQR1, L3 ) ;
00238
00239 #elif REG_SO_LENGTH == _15_CONV
00240
00241     CLR_BIT( ADC->SQR1, L0 ) ;
00242     SET_BIT( ADC->SQR1, L1 ) ;
00243     SET_BIT( ADC->SQR1, L2 ) ;
00244     SET_BIT( ADC->SQR1, L3 ) ;
00245
00246 #elif REG_SO_LENGTH == _16_CONV
00247
00248     SET_BIT( ADC->SQR1, L0 ) ;
00249     SET_BIT( ADC->SQR1, L1 ) ;
00250     SET_BIT( ADC->SQR1, L2 ) ;
00251     SET_BIT( ADC->SQR1, L3 ) ;
00252
00253 #endif
00254
00255 // EN/DIS SCAN MODE:
00256 #if SCAN_MODE == ENABLE
00257
00258     SET_BIT( ADC->CR1, SCAN ) ;
00259
00260 #elif SCAN_MODE == DISABLE
00261
00262     CLR_BIT( ADC->CR1, SCAN ) ;
00263
00264 #endif
00265
00266 // EN/DIS Discontinuous mode on Regular:
00267 #if DISC_MODE == ENABLE
00268
00269     SET_BIT( ADC->CR1, DISCEN ) ;
00270
00271 #elif DISC_MODE == DISABLE
00272
00273     CLR_BIT( ADC->CR1, DISCEN ) ;
00274
00275 #endif
00276
00277 // Discontinuous mode channel count (1/2/3/4/.../8):
00278 #if DISC_CHANNEL_NUM == _1_CHANNEL
00279
00280     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00281     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00282     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00283
00284 #elif DISC_CHANNEL_NUM == _2_CHANNELS
00285
00286     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00287     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00288     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00289
00290 #elif DISC_CHANNEL_NUM == _3_CHANNELS
00291
00292     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00293     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00294     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00295
00296 #elif DISC_CHANNEL_NUM == _4_CHANNELS
00297
00298     SET_BIT( ADC->CR1, DISCNUM0 ) ;

```

```

00299     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00300     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00301
00302 #elif DISC_CHANNEL_NUM == _5_CHANNELS
00303
00304     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00305     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00306     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00307
00308 #elif DISC_CHANNEL_NUM == _6_CHANNELS
00309
00310     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00311     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00312     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00313
00314 #elif DISC_CHANNEL_NUM == _7_CHANNELS
00315
00316     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00317     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00318     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00319
00320 #elif DISC_CHANNEL_NUM == _8_CHANNELS
00321
00322     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00323     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00324     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00325
00326
00327 #endif
00328
00329 // EN/DIS Continuous mode:
00330 #if CONT_CONV == ENABLE
00331
00332     SET_BIT( ADC->CR2, CONT ) ;
00333
00334 #elif CONT_CONV == DISABLE
00335
00336     CLR_BIT( ADC->CR2, CONT ) ;
00337
00338 #endif
00339
00340 }
00341
00342
00343 //*****
00344 //*****
00345 void MADC_vSelectChannel( u8_t A_u8ChannelNum )
00346 {
00347
00348 // Select Regular channel(1/2/3.../16)
00349 ADC->SQR3 = (ADC->SQR3 & SQ1_BIT_MANIPULATION) | ( A_u8ChannelNum ) ;
00350
00351 }
00352
00353
00354 //*****
00355 //*****
00356 u16_t MADC_u16ConvertToDigital( u8_t A_u8ChannelNum )
00357 {
00358
00359 // Select Regular channel(1/2/3.../16)
00360 ADC->SQR3 = (ADC->SQR3 & SQ1_BIT_MANIPULATION) | ( A_u8ChannelNum ) ;
00361
00362 // Start conversion of regular channels:
00363     SET_BIT( ADC->CR2, SWSTART ) ;
00364
00365 // Poll on the end of conversion flag until it's raised:
00366 while( GET_BIT(ADC->SR, EOC) != FLAG_SET ) ;
00367
00368 // Clear the flag:
00369     CLR_BIT( ADC->SR, EOC ) ;
00370
00371 // Return ADC Data:
00372     return ADC->DR ;
00373
00374 }
00375
00376
00377 //*****
00378 //*****
00379 void MADC_vDisable( void )

```

```
00380 {
00381     // DIS CLK on APB2 bus to be able to operate ADC peripheral:
00382     MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00384
00385     // Power off ADC:
00386     CLR_BIT( ADC->CR2, ADON ) ;
00387
00388 }
00389
00390
00391 /*****
00392 ****/
00393 void MADC_vEnable( void )
00394 {
00395
00396     // EN CLK on APB2 bus to be able to operate ADC peripheral:
00397     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00398
00399     // Power on ADC:
00400     SET_BIT( ADC->CR2, ADON ) ;
00401
00402 }
00403
00404
00405
00406
00407 void MADC_vRegINT_Disable( void )
00408 {
00409     CLR_BIT( ADC->CR1, EOCIE ) ;
00410 }
00411
00412
00413
00414
00415 void MADC_vRegINTEnable( void )
00416 {
00417     SET_BIT( ADC->CR1, EOCIE ) ;
00418 }
00419
00420
00421
00422
00423 void MADC_vSetCallBack( void (*MEXTI_vpPointerTo_ISR_function) (void) )
00424 {
00425
00426     if( MEXTI_vpPointerTo_ISR_function != NULL )
00427     {
00428
00429         MADC_vpPointerToFunction = MEXTI_vpPointerTo_ISR_function ;
00430
00431     }
00432
00433 }
00434
00435
00436
00437
00438 void ADC_IRQHandler(void)
00439 {
00440
00441     if( MADC_vpPointerToFunction != NULL )
00442     {
00443         MADC_vpPointerToFunction() ;
00444     }
00445
00446     // Clear the flags.
00447     CLR_BIT( ADC->SR, EOC ) ;
00448     CLR_BIT( ADC->SR, JEOC ) ;
00449 }
```

```
00457
00458
00459
00460
00461
```

7.86 EXTI_config.h

```
00001 /* FILENAME: EXTI_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 09/02/2022
00005 */
00006 #ifndef _EXTI_config_H
00007 #define _EXTI_config_H
00008
00009
00010 /***** // EXTI Configurations *****/
00011 // EXTI Configurations //
00012 /***** *****/
00013
00014 /*options:
00015 *ENABLE
00016 *DISABLE
00017 */
00018 #define EXTI_LINE0_EN ENABLE
00019
00020 /***** *****/
00021
00022 #define EXTI_LINE0_TRIGGER EXTI_FallingEdge
00023
00024 /***** *****/
00025
00026 /*options:
00027 *ENABLE
00028 *DISABLE
00029 */
00030 #define EXTI_LINE1_EN ENABLE
00031
00032 /***** *****/
00033
00034 /*ENABLE
00035 *DISABLE
00036 */
00037 #define EXTI_LINE1_TRIGGER EXTI_RisingEdge
00038
00039 /***** *****/
00040
00041 #define EXTI_LINE2_EN ENABLE
00042
00043 /***** *****/
00044
00045 /*options:
00046 *ENABLE
00047 *DISABLE
00048 */
00049 #define EXTI_LINE2_TRIGGER EXTI_RisingEdge
00050
00051 /***** *****/
00052
00053 /*options:
00054 *ENABLE
00055 *DISABLE
00056 */
00057 #define EXTI_LINE3_EN ENABLE
00058
00059 /***** *****/
00060
00061 #define EXTI_LINE3_TRIGGER EXTI_RisingEdge
00062
00063 /***** *****/
00064
00065 /*options:
00066 *ENABLE
00067 *DISABLE
00068 */
00069 #define EXTI_LINE4_EN ENABLE
00070
00071 /***** *****/
00072 #define EXTI_LINE4_TRIGGER EXTI_RisingEdge
00073
00074 /***** *****/
00075
00076 /*options:
00077 *ENABLE
00078 *DISABLE
00079 */
00080 #define EXTI_LINE5_EN ENABLE
00081
00082 /***** *****/
00083
00084 /*options:
00085 *ENABLE
00086 *DISABLE
00087 */
00088 #define EXTI_LINE6_EN ENABLE
00089
00090 /***** *****/
00091
00092 /*options:
00093 *ENABLE
00094 *DISABLE
00095 */
00096 #define EXTI_LINE7_EN ENABLE
00097
00098 /***** *****/
00099
00100 #define EXTI_LINE8_EN ENABLE
00101
00102 /***** *****/
00103
00104 /*options:
00105 *ENABLE
```

```
00106 /*DISABLE
00107 */
00108 #define EXTI_LINE5_EN ENABLE
00109
00110 /*****
00111
00118 #define EXTI_LINE5_TRIGGER EXTI_RisingEdge
00119
00120 *****/
00121
00122 /*options:
00123 *ENABLE
00124 *DISABLE
00125 */
00126 #define EXTI_LINE6_EN ENABLE
00127
00128 *****/
00129
00136 #define EXTI_LINE6_TRIGGER EXTI_RisingEdge
00137
00138 *****/
00139
00140 /*options:
00141 *ENABLE
00142 *DISABLE
00143 */
00144 #define EXTI_LINE7_EN ENABLE
00145
00146 *****/
00147
00154 #define EXTI_LINE7_TRIGGER EXTI_RisingEdge
00155
00156 *****/
00157
00158 /*options:
00159 *ENABLE
00160 *DISABLE
00161 */
00162 #define EXTI_LINE8_EN ENABLE
00163
00164 *****/
00165
00172 #define EXTI_LINE8_TRIGGER EXTI_RisingEdge
00173
00174 *****/
00175
00176 /*options:
00177 *ENABLE
00178 *DISABLE
00179 */
00180 #define EXTI_LINE9_EN ENABLE
00181
00182 *****/
00183
00190 #define EXTI_LINE9_TRIGGER EXTI_RisingEdge
00191
00192 *****/
00193
00194 /*options:
00195 *ENABLE
00196 *DISABLE
00197 */
00198 #define EXTI_LINE10_EN ENABLE
00199
00200 *****/
00201
00208 #define EXTI_LINE10_TRIGGER EXTI_RisingEdge
00209
00210 *****/
00211
00212 /*options:
00213 *ENABLE
00214 *DISABLE
00215 */
00216 #define EXTI_LINE11_EN ENABLE
00217
00218 *****/
00219
00226 #define EXTI_LINE11_TRIGGER EXTI_RisingEdge
00227
00228 *****/
00229
00230 /*options:
00231 *ENABLE
00232 *DISABLE
00233 */
00234 #define EXTI_LINE12_EN ENABLE
```

```

00235
00236 /****** */
00237
00244 #define EXTI_LINE12_TRIGGER EXTI_RisingEdge
00245
00246 /****** */
00247 /*options:
00248 *ENABLE
00249 *DISABLE
00250 */
00251 #define EXTI_LINE13_EN ENABLE
00252
00253 /****** */
00254
00261 #define EXTI_LINE13_TRIGGER EXTI_RisingEdge
00262
00263 /****** */
00264
00265 /*options:
00266 *ENABLE
00267 *DISABLE
00268 */
00269 #define EXTI_LINE14_EN ENABLE
00270
00271 /****** */
00272
00279 #define EXTI_LINE14_TRIGGER EXTI_OnChange
00280
00281 /****** */
00282
00283 /*options:
00284 *ENABLE
00285 *DISABLE
00286 */
00287 #define EXTI_LINE15_EN ENABLE
00288
00289 /****** */
00290
00297 #define EXTI_LINE15_TRIGGER EXTI_RisingEdge
00298
00299 /****** */
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315 #endif // _EXTI_config_H

```

7.87 COTS/MCAL/EXTI/EXTI_interface.h File Reference

This file contains the interfacing information for the EXTI module.

Data Structures

- struct [EXTI_ConfigType](#)
Interrupt configuration structure to initialize the interrupt with.

Macros

- #define [EXTI_LINE0](#) (0)
Line 0.

- `#define EXTI_LINE1 (1)`
Line 1.
- `#define EXTI_LINE2 (2)`
Line 2.
- `#define EXTI_LINE3 (3)`
Line 3.
- `#define EXTI_LINE4 (4)`
Line 4.
- `#define EXTI_LINE5 (5)`
Line 5.
- `#define EXTI_LINE6 (6)`
Line 6.
- `#define EXTI_LINE7 (7)`
Line 7.
- `#define EXTI_LINE8 (8)`
Line 8.
- `#define EXTI_LINE9 (9)`
Line 9.
- `#define EXTI_LINE10 (10)`
Line 10.
- `#define EXTI_LINE11 (11)`
Line 11.
- `#define EXTI_LINE12 (12)`
Line 12.
- `#define EXTI_LINE13 (13)`
Line 13.
- `#define EXTI_LINE14 (14)`
Line 14.
- `#define EXTI_LINE15 (15)`
Line 15.
- `#define EXTI_FallingEdge (1)`
trigger interrupt on falling edge
- `#define EXTI_RisingEdge (2)`
trigger interrupt on rising edge
- `#define EXTI_OnChange (3)`
trigger interrupt on level change

Functions

- `void MEXTI_vInit (void)`
Initialize the EXTI module.
- `void MEXTI_vInit_WithStruct (EXTI_ConfigType *A_xINTConfig)`
Initialize the EXTI module with a certain configuration.
- `void MEXTI_vEnableLine (u8_t A_u8LineID, u8_t A_u8TriggerStatus)`
Enable the interrupt on a certain line ID with a certain trigger status.
- `void MEXTI_vDisableLine (u8_t A_u8LineID)`
Disable the interrupt on a certain line ID.
- `void MEXTI_vSWITrigger (u8_t A_u8LineID)`
perform a software event interrupt on a certain line
- `void MEXTI_vSetTrigger (u8_t A_u8LineID, u8_t A_u8TriggerStatus)`

- Set the trigger status interrupt on a certain line ID.*
- void [MEXTI_vSetCallback](#) ([u8_t](#) A_u8LineID, void(*A_vFptr)(void))
Set the callback when an interrupt occurs on a certain line ID.
 - void [MSYSCFG_vSetEXTIPort](#) ([u8_t](#) A_u8LineID, [u8_t](#) A_u8PortID)
Enable the interrupt on a certain line in a certain port.

7.87.1 Detailed Description

This file contains the interfacing information for the EXTI module.

Author

Ali El Bana

Version

1.0

Date

09/02/2022

Definition in file [EXTI_interface.h](#).

7.87.2 Function Documentation

7.87.2.1 MEXTI_vInit()

```
void MEXTI_vInit (
    void )
```

Initialize the EXTI module.

Definition at line 40 of file [EXTI_program.c](#).

```
00041 {
00042
00043     u8\_t L_u8ShiftedOffset = INITIAL_ZERO;
00044
00045 #if EXTI_LINE0_EN == ENABLE
00046     L_u8ShiftedOffset = 0;
00047 #elif EXTI_LINE1_EN == ENABLE
00048     L_u8ShiftedOffset = 1;
00049 #elif EXTI_LINE2_EN == ENABLE
00050     L_u8ShiftedOffset = 2;
00051 #elif EXTI_LINE3_EN == ENABLE
00052     L_u8ShiftedOffset = 3;
00053 #elif EXTI_LINE4_EN == ENABLE
00054     L_u8ShiftedOffset = 4;
00055 #elif EXTI_LINE5_EN == ENABLE
00056     L_u8ShiftedOffset = 5;
00057 #elif EXTI_LINE6_EN == ENABLE
00058     L_u8ShiftedOffset = 6;
00059 #elif EXTI_LINE7_EN == ENABLE
00060     L_u8ShiftedOffset = 7;
00061 #elif EXTI_LINE8_EN == ENABLE
```

```

00062     L_u8ShiftedOffset = 8;
00063 #elif EXTI_LINE9_EN == ENABLE
00064     L_u8ShiftedOffset = 9;
00065 #elif EXTI_LINE10_EN == ENABLE
00066     L_u8ShiftedOffset = 10;
00067 #elif EXTI_LINE11_EN == ENABLE
00068     L_u8ShiftedOffset = 11;
00069 #elif EXTI_LINE12_EN == ENABLE
00070     L_u8ShiftedOffset = 12;
00071 #elif EXTI_LINE13_EN == ENABLE
00072     L_u8ShiftedOffset = 13;
00073 #elif EXTI_LINE14_EN == ENABLE
00074     L_u8ShiftedOffset = 14;
00075 #elif EXTI_LINE15_EN == ENABLE
00076     L_u8ShiftedOffset = 15;
00077 #endif
00078
00079 // Get the enabled line ID.
00080 // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
00081
00082 // Clear all flags.
00083 MEXTI->PR = 0xffffffff;
00084
00085 // Enable EXTI on a line.
00086 MEXTI->IMR = 0;
00087
00088 MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00089
00090 // Set EXTI Triggered status on a line.
00091 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00092     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00093
00094 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00095     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00096
00097 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00098     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00099     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00100
00101 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00102     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00103
00104 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00105     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00106
00107 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00108     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00109
00110 #elif EXTI_LINE2_TRIGGER == EXTI_RisingEdge
00111     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00112
00113 #elif EXTI_LINE2_TRIGGER == EXTI_FallingEdge
00114     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00115
00116 #elif EXTI_LINE2_TRIGGER == EXTI_OnChange
00117     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00118
00119 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00120     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00121
00122 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00123     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00124
00125 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00126     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00127
00128 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00129     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00130
00131 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00132     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00133
00134 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00135     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00136
00137 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00138     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00139
00140 #elif EXTI_LINE5_TRIGGER == EXTI_FallingEdge
00141     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00142
00143 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00144     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00145
00146 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00147     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00148
00149 #elif EXTI_LINE5_TRIGGER == EXTI_FallingEdge
00150     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00151
00152 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00153

```

```

00154     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00155     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00156
00158 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00159     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00160
00161 #elif EXTI_LINE6_TRIGGER == EXTI_FallingEdge
00162     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00163
00164 #elif EXTI_LINE6_TRIGGER == EXTI_OnChange
00165     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00166     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00167
00169 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00170     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00171
00172 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00173     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00174
00175 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00176     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00177     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00178
00180 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge
00181     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00182
00183 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00184     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00185
00186 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00187     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00188     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00189
00191 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00192     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00193
00194 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00195     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00196
00197 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00198     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00199     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00200
00202 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00203     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00204
00205 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00206     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00207
00208 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00209     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00210     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00211
00213 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
00214     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00215
00216 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00217     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00218
00219 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00220     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00221     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00222
00224 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00225     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00226
00227 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00228     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00229
00230 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00231     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00232     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00233
00235 #elif EXTI_LINE13_TRIGGER == EXTI_RisingEdge
00236     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00237
00238 #elif EXTI_LINE13_TRIGGER == EXTI_FallingEdge
00239     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00240
00241 #elif EXTI_LINE13_TRIGGER == EXTI_OnChange
00242     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00243     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00244
00246 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00247     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00248
00249 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge

```

```

00250     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00251
00252 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00253     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00254     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00255
00256 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00257     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00258
00259 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00260     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00261
00262 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00263     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00264     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00265
00266 #endif
00267 }

```

References [INITIAL_ZERO](#), [MEXTI](#), and [SET_BIT](#).

7.87.2.2 MEXTI_vInit_WithStruct()

```

void MEXTI_vInit_WithStruct (
    EXTI_ConfigType * A_xINTConfig )

```

Initialize the EXTI module with a certain configuration.

Parameters

in	A_xINTConfig	The configuration structure to initialize the interrupt with
----	------------------------------	--

Definition at line 273 of file [EXTI_program.c](#).

```

00274 {
00275
00276     MEXTI_vEnableLine(A_xINTConfig->LineNum, A_xINTConfig->TriggerStatus);
00277
00278     MSYSCFG_vSetEXTIPort(A_xINTConfig->LineNum, A_xINTConfig->PortNum);
00279 }

```

References [EXTI_ConfigType::LineNum](#), [MEXTI_vEnableLine\(\)](#), [MSYSCFG_vSetEXTIPort\(\)](#), [EXTI_ConfigType::PortNum](#), and [EXTI_ConfigType::TriggerStatus](#).

7.87.2.3 MEXTI_vEnableLine()

```

void MEXTI_vEnableLine (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )

```

Enable the interrupt on a certain line ID with a certain trigger status.

Parameters

in	A_u8LineID	The line ID to enable the interrupt on
in	A_u8TriggerStatus	The trigger status to trigger when the interrupt occurs

Definition at line 284 of file [EXTI_program.c](#).

```
00285 {
00286
00287     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00288     {
00289         SET_BIT(MEXTI->IMR, A_u8LineID);
00290
00291         MEXTI_vSetTrigger(A_u8LineID, A_u8TriggerStatus);
00292     }
00293 }
00294 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), [MEXTI_vSetTrigger\(\)](#), and [SET_BIT](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.87.2.4 MEXTI_vDisableLine()

```
void MEXTI_vDisableLine (
    u8_t A_u8LineID )
```

Disable the interrupt on a certain line ID.

Parameters

in	A_u8LineID	Which line to disable the interrupt on
----	------------	--

Definition at line 299 of file [EXTI_program.c](#).

```
00300 {
00301
00302     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00303     {
00304         CLR_BIT(MEXTI->IMR, A_u8LineID);
00305     }
00306 }
```

References [CLR_BIT](#), [EXTI_MAX_EXTI_NUM](#), and [MEXTI](#).

7.87.2.5 MEXTI_vSWITrigger()

```
void MEXTI_vSWITrigger (
    u8_t A_u8LineID )
```

perform a software event interrupt on a certain line

Parameters

in	A_u8LineID	The line ID to perform the software event interrupt on
----	------------	--

Definition at line 311 of file [EXTI_program.c](#).

```
00312 {
00313
00314     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00315     {
```

```

00316     SET_BIT(MEXTI->SWIER, A_u8LineID);
00317 }
00318 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), and [SET_BIT](#).

7.87.2.6 MEXTI_vSetTrigger()

```

void MEXTI_vSetTrigger (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )
```

Set the trigger status interrupt on a certain line ID.

Parameters

in	<i>A_u8LineID</i>	The line ID to enable the interrupt on
in	<i>A_u8TriggerStatus</i>	The trigger status to trigger when the interrupt occurs

Definition at line 323 of file [EXTI_program.c](#).

```

00324 {
00325
00326     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00327     {
00328
00329         switch (A_u8TriggerStatus)
00330         {
00331             case EXTI_RisingEdge:
00332                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00333
00334                 CLR_BIT(MEXTI->FTSR, A_u8LineID);
00335                 break;
00336
00337             case EXTI_FallingEdge:
00338                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00339
00340                 CLR_BIT(MEXTI->RTSR, A_u8LineID);
00341                 break;
00342
00343             case EXTI_OnChange:
00344                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00345
00346                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00347                 break;
00348         }
00349     }
00350 }
```

References [CLR_BIT](#), [EXTI_FallingEdge](#), [EXTI_MAX_EXTI_NUM](#), [EXTI_OnChange](#), [EXTI_RisingEdge](#), [MEXTI](#), and [SET_BIT](#).

Referenced by [MEXTI_vEnableLine\(\)](#).

7.87.2.7 MEXTI_vSetCallback()

```

void MEXTI_vSetCallback (
    u8_t A_u8LineID,
    void(*)(void) A_vFptr )
```

Set the callback when an interrupt occurs on a certain line ID.

Parameters

in	<i>A_u8LineID</i>	The line ID to set the callback on
in	<i>A_vFptr</i>	The callback to call when the interrupt occurs on a certain line

Definition at line 355 of file [EXTI_program.c](#).

```
00356 {
00357     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00358     {
00359         GS_vEXTI_Callback[A_u8LineID] = A_vFptr;
00360     }
00361 }
```

References [EXTI_MAX_EXTI_NUM](#).

7.87.2.8 MSYSCFG_vSetEXTIPort()

```
void MSYSCFG_vSetEXTIPort (
    u8_t A_u8LineID,
    u8_t A_u8PortID )
```

Enable the interrupt on a certain line in a certain port.

Parameters

in	<i>A_u8LineID</i>	The line ID to enable the interrupt onto
in	<i>A_u8PortID</i>	Port that the line belongs to

Definition at line 28 of file [EXTI_program.c](#).

```
00029 {
00030     u8_t L_u8Index = A_u8LineID / 4;
00031     u8_t L_u8ShiftAmount = A_u8LineID % 4;
00032
00033     CLR_BITs(MSYSCFG->EXTICR[L_u8Index], 0b1111, L_u8ShiftAmount, 4);
00034     SET_BITs(MSYSCFG->EXTICR[L_u8Index], A_u8PortID, L_u8ShiftAmount, 4);
00035 }
```

References [CLR_BITs](#), [MSYSCFG](#), and [SET_BITs](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.88 EXTI_interface.h

[Go to the documentation of this file.](#)

```
00001 /* Header file guard */
00010 #ifndef _EXTI_interface_H
00011 #define _EXTI_interface_H
00012
00018 typedef struct
00019 {
00023     u8_t LineNum;
00024
00028     u8_t PortNum ;
00029
00033     u8_t TriggerStatus;
```

```

00034
00035
00036
00037 } EXTI_ConfigType;
00038
00039 /***** Functions prototypes *****/
00040 /*          */
00041 /***** */
00042
00043 void MEXTI_vInit(void);
00044 void MEXTI_vInit_WithStruct(EXTI_ConfigType * A_xINTConfig);
00045
00046 void MEXTI_vEnableLine(u8_t A_u8LineID, u8_t A_u8TriggerStatus);
00047
00048 void MEXTI_vDisableLine(u8_t A_u8LineID);
00049
00050 void MEXTI_vSWITrigger(u8_t A_u8LineID);
00051
00052 void MEXTI_vSetTrigger(u8_t A_u8LineID, u8_t A_u8TriggerStatus);
00053
00054 void MEXTI_vSetCallback(u8_t A_u8LineID, void (*A_vFptr)(void));
00055
00056 void MSYSCFG_vSetEXTIPort(u8_t A_u8LineID, u8_t A_u8PortID);
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104 #define EXTI_LINE0 (0)
00105 #define EXTI_LINE1 (1)
00106 #define EXTI_LINE2 (2)
00107 #define EXTI_LINE3 (3)
00108 #define EXTI_LINE4 (4)
00109 #define EXTI_LINE5 (5)
00110 #define EXTI_LINE6 (6)
00111 #define EXTI_LINE7 (7)
00112 #define EXTI_LINE8 (8)
00113 #define EXTI_LINE9 (9)
00114 #define EXTI_LINE10 (10)
00115 #define EXTI_LINE11 (11)
00116 #define EXTI_LINE12 (12)
00117 #define EXTI_LINE13 (13)
00118 #define EXTI_LINE14 (14)
00119 #define EXTI_LINE15 (15)
00120
00121 #define EXTI_FallingEdge (1)
00122 #define EXTI_RisingEdge (2)
00123 #define EXTI_OnChange (3)
00124
00125 #endif ///_EXTI_interface_H

```

7.89 COTS/MCAL/EXTI/EXTI_private.h File Reference

This file contains the private information related to the EXTI module.

Data Structures

- struct [EXTI_Type](#)
EXTI Configuration structure for interrupt initialization process.

Macros

- #define [EXTI_BASE_ADDRESS](#) (0x40013C00)
- #define [MEXTI](#) ((volatile P2VAR(EXTI_Type))(EXTI_BASE_ADDRESS))
- #define [ENABLE](#) (1)
- #define [DISABLE](#) (2)
- #define [EXTI_MAX_EXTI_NUM](#) (22)
- #define [EXTI IRQs](#) 23

7.89.1 Detailed Description

This file contains the private information related to the EXTI module.

Author

Ali El Bana

Version

1.0

Date

09/02/2022

Definition in file [EXTI_private.h](#).

7.89.2 Macro Definition Documentation

7.89.2.1 EXTI IRQs

```
#define EXTI_IRQs 23
```

Number of available interrupt IRQs

Definition at line 112 of file [EXTI_private.h](#).

7.90 EXTI_private.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _EXTI_private_H
00011 #define _EXTI_private_H
00012
00018 typedef struct
00019 {
00023     u32_t IMR;
00027     u32_t EMR;
00031     u32_t RTSR;
00035     u32_t FTSR;
00039     u32_t SWIER;
00043     u32_t PR;
00044 } EXTI_Type;
00045
00057 #define EXTI_BASE_ADDRESS (0x40013C00)
00071 #define MEXTI ((volatile P2VAR(EXTI_Type))(EXTI_BASE_ADDRESS))
00085 #define ENABLE (1)
00091 #define DISABLE (2)
00105 #define EXTI_MAX_EXTI_NUM (22)
00112 #define EXTI_IRQs 23
00113
00114
00115 #endif // _EXTI_private_H
```

7.91 COTS/MCAL/EXTI/EXTI_program.c File Reference

This file contains the source code of the interfacing for the EXTI module.

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "EXTI_interface.h"
#include "EXTI_private.h"
#include "EXTI_config.h"
#include "SYSCFG_private.h"
```

Functions

- void **MSYSCFG_vSetEXTIPort** (**u8_t** A_u8LineID, **u8_t** A_u8PortID)

Enable the interrupt on a certain line in a certain port.
- void **MEXTI_vInit** (**void**)

Initialize the EXTI module.
- void **MEXTI_vInit_WithStruct** (**EXTI_ConfigType** *A_xINTConfig)

Initialize the EXTI module with a certain configuration.
- void **MEXTI_vEnableLine** (**u8_t** A_u8LineID, **u8_t** A_u8TriggerStatus)

Enable the interrupt on a certain line ID with a certain trigger status.
- void **MEXTI_vDisableLine** (**u8_t** A_u8LineID)

Disable the interrupt on a certain line ID.
- void **MEXTI_vSWITrigger** (**u8_t** A_u8LineID)

perform a software event interrupt on a certain line
- void **MEXTI_vSetTrigger** (**u8_t** A_u8LineID, **u8_t** A_u8TriggerStatus)

Set the trigger status interrupt on a certain line ID.
- void **MEXTI_vSetCallback** (**u8_t** A_u8LineID, **void**(***A_vFptr**)(**void**))

Set the callback when an interrupt occurs on a certain line ID.
- void **EXTI0_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 0 module.
- void **EXTI1_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 1 module.
- void **EXTI2_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 2 module.
- void **EXTI3_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 3 module.
- void **EXTI4_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 4 module.
- void **EXTI9_5_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 5-9 module.
- void **EXTI15_10_IRQHandler** (**void**)

This function is responsible for clearing the pending flag of the External interrupt 10-15 module.

7.91.1 Detailed Description

This file contains the source code of the interfacing for the EXTI module.

Author

Ali El Bana

Version

1.0

Date

09/02/2022

Definition in file [EXTI_program.c](#).

7.91.2 Function Documentation

7.91.2.1 MSYSCFG_vSetEXTIPort()

```
void MSYSCFG_vSetEXTIPort (
    u8_t A_u8LineID,
    u8_t A_u8PortID )
```

Enable the interrupt on a certain line in a certain port.

Parameters

in	<i>A_u8LineID</i>	The line ID to enable the interrupt onto
in	<i>A_u8PortID</i>	Port that the line belongs to

Definition at line 28 of file [EXTI_program.c](#).

```
00029 {
00030     u8_t L_u8Index = A_u8LineID / 4;
00031     u8_t L_u8ShiftAmount = A_u8LineID % 4;
00032
00033     CLR_BITS(MSYSCFG->EXTICR[L_u8Index], 0b1111, L_u8ShiftAmount, 4);
00034     SET_BITS(MSYSCFG->EXTICR[L_u8Index], A_u8PortID, L_u8ShiftAmount, 4);
00035 }
```

References [CLR_BITs](#), [MSYSCFG](#), and [SET_BITs](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.91.2.2 MEXTI_vInit()

```
void MEXTI_vInit (
    void )
```

Initialize the EXTI module.

Definition at line 40 of file [EXTI_program.c](#).

```
00041 {
00042     u8_t L_u8ShiftedOffset = INITIAL_ZERO;
00044
00045 #if EXTI_LINE0_EN == ENABLE
00046     L_u8ShiftedOffset = 0;
00047 #elif EXTI_LINE1_EN == ENABLE
00048     L_u8ShiftedOffset = 1;
00049 #elif EXTI_LINE2_EN == ENABLE
00050     L_u8ShiftedOffset = 2;
00051 #elif EXTI_LINE3_EN == ENABLE
00052     L_u8ShiftedOffset = 3;
00053 #elif EXTI_LINE4_EN == ENABLE
00054     L_u8ShiftedOffset = 4;
00055 #elif EXTI_LINE5_EN == ENABLE
00056     L_u8ShiftedOffset = 5;
00057 #elif EXTI_LINE6_EN == ENABLE
00058     L_u8ShiftedOffset = 6;
00059 #elif EXTI_LINE7_EN == ENABLE
00060     L_u8ShiftedOffset = 7;
00061 #elif EXTI_LINE8_EN == ENABLE
00062     L_u8ShiftedOffset = 8;
00063 #elif EXTI_LINE9_EN == ENABLE
00064     L_u8ShiftedOffset = 9;
00065 #elif EXTI_LINE10_EN == ENABLE
00066     L_u8ShiftedOffset = 10;
00067 #elif EXTI_LINE11_EN == ENABLE
00068     L_u8ShiftedOffset = 11;
00069 #elif EXTI_LINE12_EN == ENABLE
00070     L_u8ShiftedOffset = 12;
00071 #elif EXTI_LINE13_EN == ENABLE
00072     L_u8ShiftedOffset = 13;
00073 #elif EXTI_LINE14_EN == ENABLE
00074     L_u8ShiftedOffset = 14;
00075 #elif EXTI_LINE15_EN == ENABLE
00076     L_u8ShiftedOffset = 15;
00077 #endif
00078
00079     // Get the enabled line ID.
00080     // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
00081
00082     // Clear all flags.
00083     MEXTI->PR = 0xffffffff;
00084
00085     // Enable EXTI on a line.
00086     MEXTI->IMR = 0;
00087
00088     MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00089
00090 // Set EXTI Triggered status on a line.
00091 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00092
00093     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00094
00095 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00096     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00097
00098 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00099     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00100     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00101
00103 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00104     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00105
00106 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00107     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00108
00109 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00110     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00111     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00112
00114 #elif EXTI_LINE2_TRIGGER == EXTI_RisingEdge
00115     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00116
00117 #elif EXTI_LINE2_TRIGGER == EXTI_FallingEdge
```

```

00118     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00119
00120 #elif EXTI_LINE2_TRIGGER == EXTI_OnChange
00121     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00122     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00123
00125 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00126     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00127
00128 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00129     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00130
00131 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00132     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00133     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00134
00136 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00137     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00138
00139 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00140     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00141
00142 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00143     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00144     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00145
00147 #elif EXTI_LINES_TRIGGER == EXTI_RisingEdge
00148     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00149
00150 #elif EXTI_LINES_TRIGGER == EXTI_FallingEdge
00151     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00152
00153 #elif EXTI_LINES_TRIGGER == EXTI_OnChange
00154     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00155     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00156
00158 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00159     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00160
00161 #elif EXTI_LINE6_TRIGGER == EXTI_FallingEdge
00162     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00163
00164 #elif EXTI_LINE6_TRIGGER == EXTI_OnChange
00165     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00166     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00167
00169 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00170     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00171
00172 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00173     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00174
00175 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00176     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00177     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00178
00180 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge
00181     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00182
00183 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00184     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00185
00186 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00187     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00188     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00189
00191 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00192     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00193
00194 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00195     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00196
00197 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00198     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00199     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00200
00202 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00203     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00204
00205 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00206     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00207
00208 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00209     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00210     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00211
00213 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge

```

```

00214     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00215
00216 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00217     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00218
00219 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00220     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00221     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00222
00224 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00225     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00226
00227 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00228     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00229
00230 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00231     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00232     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00233
00235 #elif EXTI_LINE13_TRIGGER == EXTI_RisingEdge
00236     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00237
00238 #elif EXTI_LINE13_TRIGGER == EXTI_FallingEdge
00239     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00240
00241 #elif EXTI_LINE13_TRIGGER == EXTI_OnChange
00242     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00243     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00244
00246 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00247     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00248
00249 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge
00250     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00251
00252 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00253     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00254     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00255
00257 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00258     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00259
00260 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00261     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00262
00263 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00264     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00265     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00266
00267 #endif
00268 }
```

References [INITIAL_ZERO](#), [MEXTI](#), and [SET_BIT](#).

7.91.2.3 MEXTI_vInit_WithStruct()

```
void MEXTI_vInit_WithStruct (
    EXTI_ConfigType * A_xINTConfig )
```

Initialize the EXTI module with a certain configuration.

Parameters

in	<code>A_xINTConfig</code>	The configuration structure to initialize the interrupt with
----	---------------------------	--

Definition at line 273 of file [EXTI_program.c](#).

```

00274 {
00275
00276     MEXTI_vEnableLine(A_xINTConfig->LineNum, A_xINTConfig->TriggerStatus);
00277
00278     MSYSCFG_vSetEXTIPort(A_xINTConfig->LineNum, A_xINTConfig->PortNum);
00279 }
```

References [EXTI_ConfigType::LineNum](#), [MEXTI_vEnableLine\(\)](#), [MSYSCFG_vSetEXTIPort\(\)](#), [EXTI_ConfigType::PortNum](#), and [EXTI_ConfigType::TriggerStatus](#).

7.91.2.4 MEXTI_vEnableLine()

```
void MEXTI_vEnableLine (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )
```

Enable the interrupt on a certain line ID with a certain trigger status.

Parameters

in	A_u8LineID	The line ID to enable the interrupt on
in	A_u8TriggerStatus	The trigger status to trigger when the interrupt occurs

Definition at line 284 of file [EXTI_program.c](#).

```
00285 {
00286
00287     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00288     {
00289         SET_BIT (MEXTI->IMR, A_u8LineID);
00290         MEXTI_vSetTrigger (A_u8LineID, A_u8TriggerStatus);
00291     }
00292 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), [MEXTI_vSetTrigger\(\)](#), and [SET_BIT](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.91.2.5 MEXTI_vDisableLine()

```
void MEXTI_vDisableLine (
    u8_t A_u8LineID )
```

Disable the interrupt on a certain line ID.

Parameters

in	A_u8LineID	Which line to disable the interrupt on
----	------------	--

Definition at line 299 of file [EXTI_program.c](#).

```
00300 {
00301
00302     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00303     {
00304         CLR_BIT (MEXTI->IMR, A_u8LineID);
00305     }
00306 }
```

References [CLR_BIT](#), [EXTI_MAX_EXTI_NUM](#), and [MEXTI](#).

7.91.2.6 MEXTI_vSWITrigger()

```
void MEXTI_vSWITrigger (
    u8_t A_u8LineID )
```

perform a software event interrupt on a certain line

Parameters

in	A_u8LineID	The line ID to perform the software event interrupt on
----	------------	--

Definition at line 311 of file [EXTI_program.c](#).

```
00312 {
00313
00314     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00315     {
00316         SET_BIT(MEXTI->SWIER, A_u8LineID);
00317     }
00318 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), and [SET_BIT](#).

7.91.2.7 MEXTI_vSetTrigger()

```
void MEXTI_vSetTrigger (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )
```

Set the trigger status interrupt on a certain line ID.

Parameters

in	A_u8LineID	The line ID to enable the interrupt on
in	A_u8TriggerStatus	The trigger status to trigger when the interrupt occurs

Definition at line 323 of file [EXTI_program.c](#).

```
00324 {
00325
00326     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00327     {
00328
00329         switch (A_u8TriggerStatus)
00330         {
00331             case EXTI_RisingEdge:
00332                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00333
00334                 CLR_BIT(MEXTI->FTSR, A_u8LineID);
00335                 break;
00336
00337             case EXTI_FallingEdge:
00338                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00339
00340                 CLR_BIT(MEXTI->RTSR, A_u8LineID);
00341                 break;
00342
00343             case EXTI_OnChange:
00344                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00345
00346                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00347                 break;
00348         }
00349 }
```

```
00349      }
00350 }
```

References [CLR_BIT](#), [EXTI_FallingEdge](#), [EXTI_MAX_EXTI_NUM](#), [EXTI_OnChange](#), [EXTI_RisingEdge](#), [MEXTI](#), and [SET_BIT](#).

Referenced by [MEXTI_vEnableLine\(\)](#).

7.91.2.8 MEXTI_vSetCallback()

```
void MEXTI_vSetCallback (
    u8_t A_u8LineID,
    void(*)(void) A_vFptr )
```

Set the callback when an interrupt occurs on a certain line ID.

Parameters

in	<i>A_u8LineID</i>	The line ID to set the callback on
in	<i>A_vFptr</i>	The callback to call when the interrupt occurs on a certain line

Definition at line 355 of file [EXTI_program.c](#).

```
00356 {
00357
00358     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00359     {
00360         GS_vEXTI_Callback[A_u8LineID] = A_vFptr;
00361     }
00362 }
```

References [EXTI_MAX_EXTI_NUM](#).

7.91.2.9 EXTI0_IRQHandler()

```
void EXTI0_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 0 module.

Definition at line 371 of file [EXTI_program.c](#).

```
00372 {
00373
00374     if (GS_vEXTI_Callback[EXTI_LINE0] != NULL)
00375     {
00376         GS_vEXTI_Callback[EXTI_LINE0]();
00377     }
00378
00379     // Clear the flag.
00380     SET_BIT(MEXTI->PR, EXTI_LINE0);
00381 }
```

References [EXTI_LINE0](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.91.2.10 EXTI1_IRQHandler()

```
void EXTI1_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 1 module.

Definition at line 390 of file [EXTI_program.c](#).

```
00391 {
00392
00393     if (GS_vEXTI_Callback[EXTI_LINE1] != NULL)
00394     {
00395         GS_vEXTI_Callback[EXTI_LINE1]();
00396     }
00397
00398     // Clear the flag.
00399     SET_BIT(MEXTI->PR, EXTI_LINE1);
00400 }
```

References [EXTI_LINE1](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.91.2.11 EXTI2_IRQHandler()

```
void EXTI2_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 2 module.

Definition at line 409 of file [EXTI_program.c](#).

```
00410 {
00411
00412     if (GS_vEXTI_Callback[EXTI_LINE2] != NULL)
00413     {
00414         GS_vEXTI_Callback[EXTI_LINE2]();
00415     }
00416
00417     // Clear the flag.
00418     SET_BIT(MEXTI->PR, EXTI_LINE2);
00419 }
```

References [EXTI_LINE2](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.91.2.12 EXTI3_IRQHandler()

```
void EXTI3_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 3 module.

Definition at line 428 of file [EXTI_program.c](#).

```
00429 {
00430
00431     if (GS_vEXTI_Callback[EXTI_LINE3] != NULL)
00432     {
00433         GS_vEXTI_Callback[EXTI_LINE3]();
00434     }
00435
00436     // Clear the flag.
00437     SET_BIT(MEXTI->PR, EXTI_LINE3);
00438 }
```

References [EXTI_LINE3](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.91.2.13 EXTI4_IRQHandler()

```
void EXTI4_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 4 module.

Definition at line 447 of file [EXTI_program.c](#).

```
00448 {
00449
00450     if (GS_vEXTI_Callback[EXTI_LINE4] != NULL)
00451     {
00452         GS_vEXTI_Callback[EXTI_LINE4] ();
00453     }
00454
00455     // Clear the flag.
00456     SET_BIT(MEXTI->PR, EXTI_LINE4);
00457 }
```

References [EXTI_LINE4](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.91.2.14 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 5-9 module.

Definition at line 466 of file [EXTI_program.c](#).

```
00467 {
00468 }
```

7.91.2.15 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
    void )
```

This function is responsible for clearing the pending flag of the External interrupt 10-15 module.

Definition at line 477 of file [EXTI_program.c](#).

```
00478 {
00479 }
```

7.92 EXTI_program.c

[Go to the documentation of this file.](#)

```

00001
00009 /***** Include headers *****/
0010 /*           Include headers           */
0011 /***** Include headers *****/
0012 #include "../../../LIB/LSTD_TYPES.h"
0013 #include "../../../LIB/LSTD_COMPILER.h"
0014 #include "../../../LIB/LSTD_VALUES.h"
0015 #include "../../../LIB/LSTD_BITMATH.h"
0016
0017 #include "EXTI_interface.h"
0018 #include "EXTI_private.h"
0019 #include "EXTI_config.h"
0020 #include "SYSCFG_private.h"
0021
0022 STATIC P2FUNC(void, GS_vEXTI_Callback[EXTI_IRQs])(void) = {NULL};
0023
0024 /***** Functions implementations *****/
0025 /*           Functions implementations           */
0026 /***** Functions implementations *****/
0027
0028 void MSYSCFG_vSetEXTIPort(u8_t A_u8LineID, u8_t A_u8PortID)
0029 {
0030     u8_t L_u8Index = A_u8LineID / 4;
0031     u8_t L_u8ShiftAmount = A_u8LineID % 4;
0032
0033     CLR_BITs(MSYSCFG->EXTICR[L_u8Index], 0b1111, L_u8ShiftAmount, 4);
0034     SET_BITs(MSYSCFG->EXTICR[L_u8Index], A_u8PortID, L_u8ShiftAmount, 4);
0035 }
0036
0037
0038 /***** *****/
0039
0040 void MEXTI_vInit(void)
0041 {
0042
0043     u8_t L_u8ShiftedOffset = INITIAL_ZERO;
0044
0045 #if EXTI_LINE0_EN == ENABLE
0046     L_u8ShiftedOffset = 0;
0047 #elif EXTI_LINE1_EN == ENABLE
0048     L_u8ShiftedOffset = 1;
0049 #elif EXTI_LINE2_EN == ENABLE
0050     L_u8ShiftedOffset = 2;
0051 #elif EXTI_LINE3_EN == ENABLE
0052     L_u8ShiftedOffset = 3;
0053 #elif EXTI_LINE4_EN == ENABLE
0054     L_u8ShiftedOffset = 4;
0055 #elif EXTI_LINE5_EN == ENABLE
0056     L_u8ShiftedOffset = 5;
0057 #elif EXTI_LINE6_EN == ENABLE
0058     L_u8ShiftedOffset = 6;
0059 #elif EXTI_LINE7_EN == ENABLE
0060     L_u8ShiftedOffset = 7;
0061 #elif EXTI_LINE8_EN == ENABLE
0062     L_u8ShiftedOffset = 8;
0063 #elif EXTI_LINE9_EN == ENABLE
0064     L_u8ShiftedOffset = 9;
0065 #elif EXTI_LINE10_EN == ENABLE
0066     L_u8ShiftedOffset = 10;
0067 #elif EXTI_LINE11_EN == ENABLE
0068     L_u8ShiftedOffset = 11;
0069 #elif EXTI_LINE12_EN == ENABLE
0070     L_u8ShiftedOffset = 12;
0071 #elif EXTI_LINE13_EN == ENABLE
0072     L_u8ShiftedOffset = 13;
0073 #elif EXTI_LINE14_EN == ENABLE
0074     L_u8ShiftedOffset = 14;
0075 #elif EXTI_LINE15_EN == ENABLE
0076     L_u8ShiftedOffset = 15;
0077 #endif
0078
0079 // Get the enabled line ID.
0080 // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
0081
0082 // Clear all flags.
0083 MEXTI->PR = 0xffffffff;
0084
0085 // Enable EXTI on a line.
0086 MEXTI->IMR = 0;
0087

```

```

00088     MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00089
00090 // Set EXTI Triggered status on a line.
00091 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00092     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00093
00094 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00095     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00096
00097 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00098     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00099     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00100
00101 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00102     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00103
00104 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00105     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00106
00107 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00108     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00109     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00110
00111 #elif EXTI_LINE2_TRIGGER == EXTI_RisingEdge
00112     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00113
00114 #elif EXTI_LINE2_TRIGGER == EXTI_FallingEdge
00115     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00116
00117 #elif EXTI_LINE2_TRIGGER == EXTI_OnChange
00118     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00119
00120 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00121     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00122     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00123
00124 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00125     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00126
00127 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00128     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00129
00130 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00131     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00132
00133 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00134     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00135
00136 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00137     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00138
00139 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00140     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00141
00142 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00143     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00144     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00145
00146 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00147     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00148
00149 #elif EXTI_LINE5_TRIGGER == EXTI_FallingEdge
00150     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00151
00152 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00153     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00154     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00155
00156 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00157     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00158
00159 #elif EXTI_LINE6_TRIGGER == EXTI_FallingEdge
00160     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00161
00162 #elif EXTI_LINE6_TRIGGER == EXTI_OnChange
00163     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00164     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00165
00166 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00167     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00168
00169 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00170     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00171
00172 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00173     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00174     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00175
00176 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge
00177     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00178
00179 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00180     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00181
00182

```

```

00183 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00184     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00185
00186 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00187     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00188     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00189
00190 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00191     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00192
00193 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00194     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00195
00196 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00197     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00198     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00199
00200 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00201     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00202
00203 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00204     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00205
00206 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00207     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00208     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00209
00210 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
00211     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00212
00213 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00214     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00215
00216 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00217     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00218
00219 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
00220     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00221     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00222
00223 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00224     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00225
00226 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00227     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00228
00229 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00230     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00231     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00232
00233 #elif EXTI_LINE13_TRIGGER == EXTI_RisingEdge
00234     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00235
00236 #elif EXTI_LINE13_TRIGGER == EXTI_FallingEdge
00237     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00238
00239 #elif EXTI_LINE13_TRIGGER == EXTI_OnChange
00240     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00241     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00242
00243 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00244     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00245
00246 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge
00247     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00248
00249 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00250     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00251
00252 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00253     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00254     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00255
00256 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00257     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00258
00259 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00260     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00261
00262 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00263     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00264     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00265
00266 #endif
00267 }
00268
00269
00270
00271 ****
00272 ****
00273 void MEXTI_vInit_WithStruct(EXTI_ConfigType *A_xINTConfig)
00274 {

```

```

00275
00276     MEXTI_vEnableLine(A_xINTConfig->LineNum, A_xINTConfig->TriggerStatus);
00277
00278     MSYSCFG_vSetEXTIPort(A_xINTConfig->LineNum, A_xINTConfig->PortNum);
00279 }
00280
00281 //*****
00282 //*****
00283
00284 void MEXTI_vEnableLine(u8_t A_u8LineID, u8_t A_u8TriggerStatus)
00285 {
00286
00287     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00288     {
00289         SET_BIT(MEXTI->IMR, A_u8LineID);
00290
00291         MEXTI_vSetTrigger(A_u8LineID, A_u8TriggerStatus);
00292     }
00293 }
00294
00295
00296 //*****
00297 //*****
00298
00299 void MEXTI_vDisableLine(u8_t A_u8LineID)
00300 {
00301
00302     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00303     {
00304         CLR_BIT(MEXTI->IMR, A_u8LineID);
00305     }
00306 }
00307
00308
00309 //*****
00310 //*****
00311 void MEXTI_vSWITrigger(u8_t A_u8LineID)
00312 {
00313
00314     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00315     {
00316         SET_BIT(MEXTI->SWIER, A_u8LineID);
00317     }
00318 }
00319
00320
00321 //*****
00322 //*****
00323 void MEXTI_vSetTrigger(u8_t A_u8LineID, u8_t A_u8TriggerStatus)
00324 {
00325
00326     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00327     {
00328
00329         switch (A_u8TriggerStatus)
00330         {
00331             case EXTI_RisingEdge:
00332                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00333
00334                 CLR_BIT(MEXTI->FTSR, A_u8LineID);
00335                 break;
00336
00337             case EXTI_FallingEdge:
00338                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00339
00340                 CLR_BIT(MEXTI->RTSR, A_u8LineID);
00341                 break;
00342
00343             case EXTI_OnChange:
00344                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00345
00346                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00347                 break;
00348         }
00349     }
00350 }
00351
00352
00353 //*****

```

```
00353 //*****
00354 void MEXTI_vSetCallback(u8_t A_u8LineID, void (*A_vFptr) (void))
00355 {
00356     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00357     {
00358         GS_vEXTI_Callback[A_u8LineID] = A_vFptr;
00359     }
00360 }
00361 }
00362 */
00363 */
00364 //*****
00365 */
00366 */
00367 void EXTI0_IRQHandler(void)
00368 {
00369     if (GS_vEXTI_Callback[EXTI_LINE0] != NULL)
00370     {
00371         GS_vEXTI_Callback[EXTI_LINE0]();
00372     }
00373
00374     // Clear the flag.
00375     SET_BIT(MEXTI->PR, EXTI_LINE0);
00376 }
00377 */
00378 */
00379 //*****
00380 */
00381 */
00382 */
00383 */
00384 //*****
00385 */
00386 */
00387 void EXTI1_IRQHandler(void)
00388 {
00389     if (GS_vEXTI_Callback[EXTI_LINE1] != NULL)
00390     {
00391         GS_vEXTI_Callback[EXTI_LINE1]();
00392     }
00393
00394     // Clear the flag.
00395     SET_BIT(MEXTI->PR, EXTI_LINE1);
00396 }
00397 */
00398 */
00399 //*****
00400 */
00401 */
00402 */
00403 */
00404 */
00405 void EXTI2_IRQHandler(void)
00406 {
00407     if (GS_vEXTI_Callback[EXTI_LINE2] != NULL)
00408     {
00409         GS_vEXTI_Callback[EXTI_LINE2]();
00410     }
00411
00412     // Clear the flag.
00413     SET_BIT(MEXTI->PR, EXTI_LINE2);
00414 }
00415 */
00416 */
00417 */
00418 */
00419 */
00420 */
00421 */
00422 */
00423 */
00424 void EXTI3_IRQHandler(void)
00425 {
00426     if (GS_vEXTI_Callback[EXTI_LINE3] != NULL)
00427     {
00428         GS_vEXTI_Callback[EXTI_LINE3]();
00429     }
00430
00431     // Clear the flag.
00432     SET_BIT(MEXTI->PR, EXTI_LINE3);
00433 }
00434 */
00435 */
00436 */
00437 */
00438 */
00439 */
00440 */
00441 */
00442 */
00443 */
00444 */
00445 */
00446 */
00447 void EXTI4_IRQHandler(void)
00448 {
```

```

00449
00450     if (GS_vEXTI_Callback[EXTI_LINE4] != NULL)
00451     {
00452         GS_vEXTI_Callback[EXTI_LINE4]();
00453     }
00454
00455     // Clear the flag.
00456     SET_BIT(MEXTI->PR, EXTI_LINE4);
00457 }
00458
00459
00460     /*****
00461     *****/
00462
00463 void EXTI9_5_IRQHandler(void)
00464 {
00465 }
00466
00467
00468
00469
00470
00471     /*****
00472     *****/
00473
00474 void EXTI15_10_IRQHandler(void)
00475 {
00476 }
00477
00478
00479
00480
00481
00482     /*****
00483     *****/

```

7.93 COTS/MCAL/EXTI/SYSCFG_private.h File Reference

This file contains the private information regarding the SYSCFG.

Data Structures

- struct [MSYSCFG_MemMap_t](#)
System configuration structure to initialize the SYSCFG module with.

Macros

- #define [SYSCFG_BASE_ADDR](#) (0x40013800)
- #define [MSYSCFG](#) ((volatile [MSYSCFG_MemMap_t](#) *)([SYSCFG_BASE_ADDR](#)))

7.93.1 Detailed Description

This file contains the private information regarding the SYSCFG.

Author

Ali El Bana

This is needed to ensure that the clocked is enabled for the EXTI module

Version

1.0

Date

09/01/2022

Definition in file [SYSCFG_private.h](#).

7.94 SYSCFG_private.h

[Go to the documentation of this file.](#)

```

00001 /* Header file guard */
00010 #ifndef MCAL_EXTI_SYSCFG_PRV_H_
00011 #define MCAL_EXTI_SYSCFG_PRV_H_
00013
00019 typedef struct
00020 {
00025     u32_t MEMRMP;
00029     u32_t PMC;
00033     u32_t EXTICR[4];
00037     u32_t CMPCR;
00038 } MSYSCFG_MemMap_t;
00039
00051 #define SYSCFG_BASE_ADDR (0x40013800)
00064 #define MSYSCFG ((volatile MSYSCFG_MemMap_t *) (SYSCFG_BASE_ADDR))
00067 #endif /* MCAL_EXTI_SYSCFG_PRV_H_ */

```

7.95 COTS/MCAL/GPIO/GPIO_config.h File Reference

This file contains the GPIO configurations.

Macros

- **#define GPIOA_PIN_POS** (0b1110000000000000)
lock PA13, PA14 and PA15
- **#define GPIOB_PIN_POS** (0b0000100000011100)
lock PB2, PB3 and PB4
- **#define LCKK_BIT_POS** (16U)
Position of LCKK bit.
- **#define PORTA_BIT_MANIPULATION** 0xE000
Port A data output manipulation mask.
- **#define PORTB_BIT_MANIPULATION** 0x001C
Port B data output manipulation mask.

7.95.1 Detailed Description

This file contains the GPIO configurations.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

08/22/2022

Definition in file [GPIO_config.h](#).

7.95.2 Macro Definition Documentation

7.95.2.1 GPIOA_PIN_POS

```
#define GPIOA_PIN_POS (0b1110000000000000)
```

lock PA13, PA14 and PA15

Definition at line [21](#) of file [GPIO_config.h](#).

7.95.2.2 GPIOB_PIN_POS

```
#define GPIOB_PIN_POS (0b0000100000011100)
```

lock PB2, PB3 and PB4

Definition at line [28](#) of file [GPIO_config.h](#).

7.95.2.3 LCKK_BIT_POS

```
#define LCKK_BIT_POS (16U)
```

Position of LCKK bit.

Definition at line [35](#) of file [GPIO_config.h](#).

7.95.2.4 PORTA_BIT_MANIPULATION

```
#define PORTA_BIT_MANIPULATION 0xE000
```

Port A data output manipulation mask.

Definition at line [41](#) of file [GPIO_config.h](#).

7.95.2.5 PORTB_BIT_MANIPULATION

```
#define PORTB_BIT_MANIPULATION 0x001C
```

Port B data output manipulation mask.

Definition at line [47](#) of file [GPIO_config.h](#).

7.96 GPIO_config.h

[Go to the documentation of this file.](#)

```

00001
00009 #ifndef _GPIO_config_H
00010 #define _GPIO_config_H
00011
00012 /***** *****/
00013 // GPIOx configurations //
00014 /***** *****/
00015
00021 #define GPIOA_PIN_POS (0b1110000000000000) // I want to lock PA13,14,15.
00022
00028 #define GPIOB_PIN_POS (0b0000100000011100) // I want to lock PB2,3,4,11.
00029
00035 #define LCKK_BIT_POS (16U) // Position of LCKK bit.
00036
00041 #define PORTA_BIT_MANIPULATION 0xE000
00042
00047 #define PORTB_BIT_MANIPULATION 0x001C
00048
00049 #endif // _GPIO_config_H

```

7.97 COTS/MCAL.GPIO/GPIO_interface.h File Reference

This file contains the interfacing information for the GPIO module.

```

#include "../LIB/LSTD_TYPES.h"
#include "../LIB/LSTD_COMPILER.h"
#include "../LIB/LSTD_VALUES.h"
#include "../LIB/LSTD_BITMATH.h"

```

Data Structures

- struct **MGPIOx_ConfigType**
MDIO Configuration structure for a specific PIN initialization.

Macros

- #define **GPIOx_MODE_INPUT** (0b00)
Control input mode.
- #define **GPIOx_MODE_OUTPUT** (0b01)
Control output mode.
- #define **GPIOx_MODE_AF** (0b10)
Control alternate function mode.
- #define **GPIOx_MODE_ANALOG** (0b11)
Control analog mode.
- #define **GPIO_PORTA** (0)
GPIO Port A.
- #define **GPIO_PORTB** (1)
GPIO Port B.
- #define **GPIO_PORTC** (2)
GPIO Port C.
- #define **GPIOx_OPENDRAIN** (1)
GPIO open-drain.

- #define **GPIOx_PUSH_PULL** (2)
GPIO push-pull.
- #define **GPIOx_LowSpeed** (0b00)
GPIO low speed.
- #define **GPIOx_MediumSpeed** (0b01)
GPIO medium speed.
- #define **GPIOx_HighSpeed** (0b10)
GPIO high speed.
- #define **GPIOx_VeryHighSpeed** (0b11)
GPIO very high speed.
- #define **GPIOx_NoPull** (0b00)
GPIO No PULL.
- #define **GPIOx_PullUp** (0b01)
GPIO Pull UP.
- #define **GPIOx_PullDown** (0b10)
GPIO Pull Down.
- #define **GPIOx_HIGH** (1)
GPIO output high.
- #define **GPIOx_LOW** (2)
GPIO output low.
- #define **GPIOx_PIN0** (0)
GPIO PIN 0.
- #define **GPIOx_PIN1** (1)
GPIO PIN 1.
- #define **GPIOx_PIN2** (2)
GPIO PIN 2.
- #define **GPIOx_PIN3** (3)
GPIO PIN 3.
- #define **GPIOx_PIN4** (4)
GPIO PIN 4.
- #define **GPIOx_PIN5** (5)
GPIO PIN 5.
- #define **GPIOx_PIN6** (6)
GPIO PIN 6.
- #define **GPIOx_PIN7** (7)
GPIO PIN 7.
- #define **GPIOx_PIN8** (8)
GPIO PIN 8.
- #define **GPIOx_PIN9** (9)
brief GPIO PIN 9
- #define **GPIOx_PIN10** (10)
GPIO PIN 10.
- #define **GPIOx_PIN11** (11)
GPIO PIN 11.
- #define **GPIOx_PIN12** (12)
GPIO PIN 12.
- #define **GPIOx_PIN13** (13)
GPIO PIN 13.
- #define **GPIOx_PIN14** (14)
GPIO PIN 14.
- #define **GPIOx_PIN15** (15)

- `#define GPIOx_AF0 (0)`
GPIO Alternate function 0.
- `#define GPIOx_AF1 (1)`
GPIO Alternate function 1.
- `#define GPIOx_AF2 (2)`
GPIO Alternate function 2.
- `#define GPIOx_AF3 (3)`
GPIO Alternate function 3.
- `#define GPIOx_AF4 (4)`
GPIO Alternate function 4.
- `#define GPIOx_AF5 (5)`
GPIO Alternate function 5.
- `#define GPIOx_AF6 (6)`
GPIO Alternate function 6.
- `#define GPIOx_AF7 (7)`
GPIO Alternate function 7.
- `#define GPIOx_AF8 (8)`
GPIO Alternate function 8.
- `#define GPIOx_AF9 (9)`
GPIO Alternate function 9.
- `#define GPIOx_AF10 (10)`
GPIO Alternate function 10.
- `#define GPIOx_AF11 (11)`
GPIO Alternate function 11.
- `#define GPIOx_AF12 (12)`
GPIO Alternate function 12.
- `#define GPIOx_AF13 (13)`
GPIO Alternate function 13.
- `#define GPIOx_AF14 (14)`
GPIO Alternate function 14.
- `#define GPIOx_AF15 (15)`
GPIO Alternate function 15.
- `#define GPIOA_LOW_NIBBLE_HIGH 0xFF`
GPIO Port A low nibble high.
- `#define GPIOA_LOW_NIBBLE_LOW 0x00`
GPIO Port A low nibble low.

Functions

- `void MGPIOp_vLockedPins (void)`
Locks the prohibited GPIO PINs.
- `void MGPIOp_vSetPinMode (8_t A_u8PortID, 8_t A_u8PinID, 8_t A_u8Mode)`
Sets a certain pin's mode on a specific port.
- `void MGPIOp_vSetPinOutputType (8_t A_u8PortID, 8_t A_u8PinID, 8_t A_u8OutputType)`
Sets a certain pin's output type on a specific port.
- `void MGPIOp_vSetPinOutputSpeed (8_t A_u8PortID, 8_t A_u8PinID, 8_t A_u8OutputSpeed)`
Sets a certain pin's output speed on a specific port.
- `void MGPIOp_vSetPinInputPullType (8_t A_u8PortID, 8_t A_u8PinID, 8_t A_u8InputPullType)`
Sets a certain pin's input pull type on a specific port.

- `u8_t MGPOx_u8GetValue (u8_t A_u8PortID, u8_t A_u8PinID)`
Gets the value currently on a certain pin.
- `void MGPOx_vSetPinValue (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8PinValue)`
Sets a certain pin's output value on a specific port.
- `void MGPOx_vSetResetAtomic (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8SetResetPinValue)`
Resets a certain pin's output value on a specific port.
- `void MGPOx_vSetAlternateFunctionON (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8AFID)`
Applies an alternative function on a certain pin.
- `void MGPOx_vSetPortConfigLock (u8_t A_u8PortID)`
Updates a port's configuration lock.
- `void MGPOx_vInit (MGPOx_ConfigType *A_xPinConfig)`
Initialize the GPIO with a certain configuration.
- `void MGPOx_vTogglePinValue (u8_t A_u8PortID, u8_t A_u8PinID)`
Toggles a certain's pin's value on a certain port.
- `void GPIO_vSetNibbleLowValue (u8_t A_u8PortID, u16_t A_u16PortValue)`
Sets a certain port's output value.

7.97.1 Detailed Description

This file contains the interfacing information for the GPIO module.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

11/9/2022

Definition in file [GPIO_interface.h](#).

7.97.2 Macro Definition Documentation

7.97.2.1 GPIOA_LOW_NIBBLE_HIGH

```
#define GPIOA_LOW_NIBBLE_HIGH 0xFF
```

GPIO Port A low nibble high.

Definition at line 577 of file [GPIO_interface.h](#).

7.97.2.2 GPIOA_LOW_NIBBLE_LOW

```
#define GPIOA_LOW_NIBBLE_LOW 0x00
```

GPIO Port A low nibble low.

Definition at line 583 of file [GPIO_interface.h](#).

7.97.3 Function Documentation

7.97.3.1 MGPIox_vLockedPins()

```
void MGPIox_vLockedPins (
    void )
```

Locks the prohibited GPIO PINs.

Definition at line 25 of file [GPIO_program.c](#).

```
00026 {
00027
00028     VAR(volatile u32_t) L_u32LockGPIOA = INITIAL_ZERO ;
00029
00030     /* Lock key write sequence */
00031
00032     /* WR LCKR[16] = '1' + LCKR[13,14,15] = *1* */
00033     L_u32LockGPIOA = ( (1UL << LCKK_BIT_POS) | (GPIOA_PIN_POS) ) ;
00034
00035     GPIOA->LCKRx = L_u32LockGPIOA ;
00036
00037     /* WR LCKR[16] = '0' + LCKR[13,14,15] should not change*/
00038     GPIOA->LCKRx = (GPIOA_PIN_POS) ;
00039
00040     /* WR LCKR[16] = '1'+ LCKR[13,14,15] should not change*/
00041     GPIOA->LCKRx = L_u32LockGPIOA ;
00042
00043     /* RD LCKR */
00044     L_u32LockGPIOA = GPIOA->LCKRx ;
00045
00046
00047     VAR(volatile u32_t) L_u32LockGPIOB = INITIAL_ZERO ;
00048
00049     /* Lock key write sequence */
00050
00051     /* WR LCKR[16] = '1' + LCKR[2,3,4,11] = *1* */
00052     L_u32LockGPIOB = ( (1UL << LCKK_BIT_POS) | (GPIOB_PIN_POS) ) ;
00053
00054     GPIOB->LCKRx = L_u32LockGPIOB ;
00055
00056     /* WR LCKR[16] = '0' + LCKR[2,3,4,11] should not change*/
00057     GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00058
00059     /* WR LCKR[16] = '1' + LCKR[2,3,4,11] should not change*/
00060     GPIOB->LCKRx = L_u32LockGPIOB ;
00061
00062     /* RD LCKR */
00063     L_u32LockGPIOB = GPIOB->LCKRx ;
00064
00065
00066 }
```

References [GPIOA](#), [GPIOA_PIN_POS](#), [GPIOB](#), [GPIOB_PIN_POS](#), [INITIAL_ZERO](#), [LCKK_BIT_POS](#), and [VAR](#).

7.97.3.2 MGPIox_vSetPinMode()

```
void MGPIox_vSetPinMode (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8Mode )
```

Sets a certain pin's mode on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8Mode</i>	The mode to apply the pin

Definition at line 71 of file [GPIO_program.c](#).

```
00072 {
00073
00074     switch (A_u8PortID)
00075     {
00076         case GPIO_PORTA:
00077             CLR_BITS(GPIOA->MODERx, 0b11, A_u8PinID, 2);
00078             SET_BITS(GPIOA->MODERx, A_u8Mode, A_u8PinID, 2);
00079             break;
00080
00081         case GPIO_PORTB:
00082             CLR_BITS(GPIOB->MODERx, 0b11, A_u8PinID, 2);
00083             SET_BITS(GPIOB->MODERx, A_u8Mode, A_u8PinID, 2);
00084             break;
00085
00086         case GPIO_PORTC:
00087             CLR_BITS(GPIOC->MODERx, 0b11, A_u8PinID, 2);
00088             SET_BITS(GPIOC->MODERx, A_u8Mode, A_u8PinID, 2);
00089             break;
00090     }
00091 }
00092 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITs](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.97.3.3 MGPIOx_vSetPinOutputType()

```
void MGPIOx_vSetPinOutputType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputType )
```

Sets a certain pin's output type on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8OutputType</i>	The output type to apply on the pin

Definition at line 97 of file [GPIO_program.c](#).

```
00098 {
00099
00100     switch (A_u8OutputType)
00101     {
00102         case GPIOx_OPENDRAIN:
00103             switch (A_u8PortID)
00104             {
00105                 case GPIO_PORTA:
00106                     SET_BIT(GPIOA->OTYPERx, A_u8PinID);
00107                     break;
00108
00109                 case GPIO_PORTB:
00110                     SET_BIT(GPIOB->OTYPERx, A_u8PinID);
```

```

00111         break;
00112
00113     case GPIO_PORTC:
00114         SET_BIT(GPIOC->OTYPERx, A_u8PinID);
00115         break;
00116     }
00117     break;
00118
00119     case GPIOx_PUSH_PULL:
00120         switch(A_u8PortID)
00121         {
00122             case GPIO_PORTA:
00123                 CLR_BIT(GPIOA->OTYPERx, A_u8PinID);
00124                 break;
00125
00126             case GPIO_PORTB:
00127                 CLR_BIT(GPIOB->OTYPERx, A_u8PinID);
00128                 break;
00129
00130             case GPIO_PORTC:
00131                 CLR_BIT(GPIOC->OTYPERx, A_u8PinID);
00132                 break;
00133         }
00134     break;
00135 }
00136 }
00137 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_OPENDRAIN](#), [GPIOx_PUSH_PULL](#), and [SET_BIT](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.97.3.4 MGPIOx_vSetPinOutputSpeed()

```

void MGPIOx_vSetPinOutputSpeed (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputSpeed )
```

Sets a certain pin's output speed on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8OutputSpeed</i>	The output speed to apply on the pin

Definition at line 142 of file [GPIO_program.c](#).

```

00143 {
00144
00145     switch(A_u8PortID)
00146     {
00147         case GPIO_PORTA:
00148             CLR_BITS(GPIOA->OSPEEDRx, 0b11, A_u8PinID, 2);
00149             SET_BITS(GPIOA->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00150             break;
00151
00152         case GPIO_PORTB:
00153             CLR_BITS(GPIOB->OSPEEDRx, 0b11, A_u8PinID, 2);
00154             SET_BITS(GPIOB->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00155             break;
00156
00157         case GPIO_PORTC:
00158             CLR_BITS(GPIOC->OSPEEDRx, 0b11, A_u8PinID, 2);
00159             SET_BITS(GPIOC->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00160             break;
00161 }
```

```
00161      }
00162
00163 }
```

References [CLR_BITS](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITS](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.97.3.5 MGPIOx_vSetPinInputPullType()

```
void MGPIOx_vSetPinInputPullType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8InputPullType )
```

Sets a certain pin's input pull type on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8InputPullType</i>	The input pull type to apply on the pin

Definition at line 168 of file [GPIO_program.c](#).

```
00169 {
00170
00171     switch (A_u8PortID)
00172     {
00173         case GPIO_PORTA:
00174             CLR_BITS(GPIOA->PUPDRx, 0b11, A_u8PinID, 2);
00175             SET_BITS(GPIOA->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00176             break;
00177
00178         case GPIO_PORTB:
00179             CLR_BITS(GPIOB->PUPDRx, 0b11, A_u8PinID, 2);
00180             SET_BITS(GPIOB->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00181             break;
00182
00183         case GPIO_PORTC:
00184             CLR_BITS(GPIOC->PUPDRx, 0b11, A_u8PinID, 2);
00185             SET_BITS(GPIOC->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00186             break;
00187     }
00188
00189 }
```

References [CLR_BITS](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITS](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.97.3.6 MGPIOx_u8GetPinValue()

```
u8_t MGPIOx_u8GetPinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Gets the value currently on a certain pin.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode

Returns

The current value on the pin

Definition at line 194 of file [GPIO_program.c](#).

```
00195 {
00196
00197     u8_t L_u8PinValue = INITIAL_ZERO;
00198
00199     switch (A_u8PortID)
00200     {
00201         case GPIO_PORTA:
00202             L_u8PinValue = GET_BIT(GPIOA->IDRx, A_u8PinID);
00203             break;
00204
00205         case GPIO_PORTB:
00206             L_u8PinValue = GET_BIT(GPIOB->IDRx, A_u8PinID);
00207             break;
00208
00209         case GPIO_PORTC:
00210             L_u8PinValue = GET_BIT(GPIOC->IDRx, A_u8PinID);
00211             break;
00212     }
00213
00214     return L_u8PinValue;
00215
00216 }
```

References [GET_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [INITIAL_ZERO](#).

7.97.3.7 MGPIox_vSetValue()

```
void MGPIox_vSetValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8PinValue )
```

Sets a certain pin's output value on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8PinValue</i>	The set value to set on the pin

Definition at line 221 of file [GPIO_program.c](#).

```
00222 {
00223
00224     switch (A_u8PinValue)
00225     {
00226         case GPIOx_HIGH:
00227             switch (A_u8PortID)
00228             {
00229                 case GPIO_PORTA:
00230                     SET_BIT(GPIOA->ODRx, A_u8PinID);
00231                     break;
```

```

00232         case GPIO_PORTB:
00233             SET_BIT(GPIOB->ODRx, A_u8PinID);
00234             break;
00235
00236         case GPIO_PORTC:
00237             SET_BIT(GPIOC->ODRx, A_u8PinID);
00238             break;
00239     }
00240     break;
00241
00242     case GPIOx_LOW:
00243         switch( A_u8PortID )
00244         {
00245             case GPIO_PORTA:
00246                 CLR_BIT(GPIOA->ODRx, A_u8PinID);
00247                 break;
00248
00249             case GPIO_PORTB:
00250                 CLR_BIT(GPIOB->ODRx, A_u8PinID);
00251                 break;
00252
00253             case GPIO_PORTC:
00254                 CLR_BIT(GPIOC->ODRx, A_u8PinID);
00255                 break;
00256         }
00257     break;
00258 }
00259 }
00260
00261 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), [GPIOx_LOW](#), and [SET_BIT](#).

Referenced by [HBUZZER_vSoundOff\(\)](#), [HBUZZER_vSoundOn\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), and [HDCM_vStopMotor\(\)](#).

7.97.3.8 MGPIOx_vSetResetAtomic()

```
void MGPIOx_vSetResetAtomic (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8SetResetPinValue )
```

Resets a certain pin's output value on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8SetResetPinValue</i>	The reset value to set on the pin

Definition at line 266 of file [GPIO_program.c](#).

```

00267 {
00268
00269     switch (A_u8PortID)
00270     {
00271
00272         case GPIO_PORTA:
00273
00274         switch (A_u8SetResetPinValue)
00275         {
00276             case GPIOx_HIGH:
00277                 GPIOA->BSRRx = (1 << A_u8PinID);
00278                 break;
00279
00280             case GPIOx_LOW:
```

```

00281         GPIOA->BSRRx = (1 << (A_u8PinID + 16));
00282         break;
00283     }
00284
00285     break;
00286
00287     case GPIO_PORTB:
00288
00289         switch (A_u8SetResetPinValue)
00290     {
00291         case GPIOx_HIGH:
00292             GPIOB->BSRRx = (1 << A_u8PinID);
00293             break;
00294
00295         case GPIOx_LOW:
00296             GPIOB->BSRRx = (1 << (A_u8PinID + 16));
00297             break;
00298         }
00299
00300         break;
00301
00302     case GPIO_PORTC:
00303
00304         switch (A_u8SetResetPinValue)
00305     {
00306         case GPIOx_HIGH:
00307             GPIOC->BSRRx = (1 << A_u8PinID);
00308             break;
00309
00310         case GPIOx_LOW:
00311             GPIOC->BSRRx = (1 << (A_u8PinID + 16));
00312             break;
00313         }
00314
00315         break;
00316     }
00317
00318 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), and [GPIOx_LOW](#).

Referenced by [HULTSNC_vTrigger\(\)](#).

7.97.3.9 MGPIOp_vSetAlternateFunctionON()

```
void MGPIOp_vSetAlternateFunctionON (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8AFID )
```

Applies an alternative function on a certain pin.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8AFID</i>	The alternative function to apply on the pin

Definition at line 323 of file [GPIO_program.c](#).

```

00324 {
00325
00326     switch (A_u8PortID)
00327     {
00328         case GPIO_PORTA:
00329             switch (A_u8PinID)
00330             {
```

```

00331     case GPIOx_PIN0:
00332     case GPIOx_PIN1:
00333     case GPIOx_PIN2:
00334     case GPIOx_PIN3:
00335     case GPIOx_PIN4:
00336     case GPIOx_PIN5:
00337     case GPIOx_PIN6:
00338     case GPIOx_PIN7:
00339         CLR_BITS(GPIOA->AFRLx, 0b1111, A_u8PinID, 4);
00340         SET_BITS(GPIOA->AFRLx, A_u8AFID, A_u8PinID, 4);
00341         break;
00342
00343     case GPIOx_PIN8:
00344     case GPIOx_PIN9:
00345     case GPIOx_PIN10:
00346     case GPIOx_PIN11:
00347     case GPIOx_PIN12:
00348     case GPIOx_PIN13:
00349     case GPIOx_PIN14:
00350     case GPIOx_PIN15:
00351         CLR_BITS(GPIOA->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00352         SET_BITS(GPIOA->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00353         break;
00354     }
00355     break;
00356
00357 case GPIO_PORTB:
00358     switch (A_u8PinID)
00359     {
00360         case GPIOx_PIN0:
00361         case GPIOx_PIN1:
00362         case GPIOx_PIN2:
00363         case GPIOx_PIN3:
00364         case GPIOx_PIN4:
00365         case GPIOx_PIN5:
00366         case GPIOx_PIN6:
00367         case GPIOx_PIN7:
00368             CLR_BITS(GPIOB->AFRLx, 0b1111, A_u8PinID, 4);
00369             SET_BITS(GPIOB->AFRLx, A_u8AFID, A_u8PinID, 4);
00370             break;
00371
00372         case GPIOx_PIN8:
00373         case GPIOx_PIN9:
00374         case GPIOx_PIN10:
00375         case GPIOx_PIN11:
00376         case GPIOx_PIN12:
00377         case GPIOx_PIN13:
00378         case GPIOx_PIN14:
00379         case GPIOx_PIN15:
00380             CLR_BITS(GPIOB->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00381             SET_BITS(GPIOB->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00382             break;
00383     }
00384     break;
00385
00386 case GPIO_PORTC:
00387     switch (A_u8PinID)
00388     {
00389         case GPIOx_PIN0:
00390         case GPIOx_PIN1:
00391         case GPIOx_PIN2:
00392         case GPIOx_PIN3:
00393         case GPIOx_PIN4:
00394         case GPIOx_PIN5:
00395         case GPIOx_PIN6:
00396         case GPIOx_PIN7:
00397             CLR_BITS(GPIOC->AFRLx, 0b1111, A_u8PinID, 4);
00398             SET_BITS(GPIOC->AFRLx, A_u8AFID, A_u8PinID, 4);
00399             break;
00400
00401         case GPIOx_PIN8:
00402         case GPIOx_PIN9:
00403         case GPIOx_PIN10:
00404         case GPIOx_PIN11:
00405         case GPIOx_PIN12:
00406         case GPIOx_PIN13:
00407         case GPIOx_PIN14:
00408         case GPIOx_PIN15:
00409             CLR_BITS(GPIOC->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00410             SET_BITS(GPIOC->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00411             break;
00412     }
00413     break;
00414 }
00415
00416 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_PIN0](#), [GPIOx_PIN1](#), [GPIOx_PIN10](#), [GPIOx_PIN11](#), [GPIOx_PIN12](#), [GPIOx_PIN13](#), [GPIOx_PIN14](#), [GPIOx_PIN15](#), [GPIOx_PIN2](#), [GPIOx_PIN3](#), [GPIOx_PIN4](#), [GPIOx_PIN5](#), [GPIOx_PIN6](#), [GPIOx_PIN7](#), [GPIOx_PIN8](#), [GPIOx_PIN9](#), and [SET_BITs](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.97.3.10 MGPIOx_vSetPortConfigLock()

```
void MGPIOx_vSetPortConfigLock (
    u8_t A_u8PortID )
```

Updates a port's configuration lock.

Parameters

in	A_u8PortID	The port to update the pin's mode
----	------------	-----------------------------------

7.97.3.11 MGPIOx_vInit()

```
void MGPIOx_vInit (
    MGPIOx_ConfigType * A_xPinConfig )
```

Initialize the GPIO with a certain configuration.

Parameters

in	A_xPinConfig	The initialization configuration for the GPIO
----	--------------	---

Definition at line 421 of file [GPIO_program.c](#).

```
00422 {
00423
00424     MGPIOx_vSetPinMode           (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->Mode
00425     );
00426     MGPIOx_vSetPinOutputType    (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputType
00427     );
00428     MGPIOx_vSetPinOutputSpeed   (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputSpeed
00429     );
00430 }
```

References [MGPIOx_ConfigType::AF_Type](#), [MGPIOx_ConfigType::InputType](#), [MGPIOx_vSetAlternateFunctionON\(\)](#), [MGPIOx_vSetPinInputPullType\(\)](#), [MGPIOx_vSetPinMode\(\)](#), [MGPIOx_vSetPinOutputSpeed\(\)](#), [MGPIOx_vSetPinOutputType\(\)](#), [MGPIOx_ConfigType::Mode](#), [MGPIOx_ConfigType::OutputSpeed](#), [MGPIOx_ConfigType::OutputType](#), [MGPIOx_ConfigType::Pin](#), and [MGPIOx_ConfigType::Port](#).

Referenced by [HBUZZER_vInit\(\)](#), [HDCM_vInitMotor\(\)](#), [HULTSNC_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.97.3.12 MGPIox_vTogglePinValue()

```
void MGPIox_vTogglePinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Toggles a certain's pin's value on a certain port.

Parameters

in	A_u8PortID	The port that the pin belongs to
in	A_u8PinID	The pin to toggle its value

Definition at line 435 of file [GPIO_program.c](#).

```
00436 {
00437
00438     switch (A_u8PortID)
00439     {
00440         case GPIO_PORTA:
00441             TOGGLE_BIT(GPIOA->ODRx, A_u8PinID);
00442             break;
00443
00444         case GPIO_PORTB:
00445             TOGGLE_BIT(GPIOB->ODRx, A_u8PinID);
00446             break;
00447
00448         case GPIO_PORTC:
00449             TOGGLE_BIT(GPIOC->ODRx, A_u8PinID);
00450             break;
00451     }
00452 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [TOGGLE_BIT](#).

Referenced by [HBUZZER_vToggleSound\(\)](#).

7.97.3.13 GPIO_vSetNibbleLowValue()

```
void GPIO_vSetNibbleLowValue (
    u8_t A_u8PortID,
    u16_t A_u16PortValue )
```

Sets a certain port's output value.

Parameters

in	A_u8PortID	The port to out the value on
in	A_u16PortValue	The value to set on the port

Definition at line 458 of file [GPIO_program.c](#).

```
00459 {
00460
00461     switch (A_u8PortID)
00462     {
00463
00464         case GPIO_PORTA :  GPIOA->ODRx = ( ((GPIOA->ODRx & PORTA_BIT_MANIPULATION) | (A_u16PortValue))
00465             << 0 ) ; break; //Start writing from PIN0 to PIN7.
00466 }
```

```

00466     case GPIO_PORTB : GPIOB->ODRx = ( ((GPIOB->ODRx & PORTB_BIT_MANIPULATION) | (A_u16PortValue))
00467     << 5 ) ; break; //Start writing from PIN5 to PIN12.
00468     case GPIO_PORTC : GPIOC->ODRx ; break; // You shouldn't write on this port.
00469     default : break;
00470 }
00471 }
00472 }
00473 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [PORTA_BIT_MANIPULATION](#), and [PORTB_BIT_MANIPULATION](#).

7.98 GPIO_interface.h

[Go to the documentation of this file.](#)

```

00001
00009 #include "../../LIB/LSTD_TYPES.h"
00010 #include "../../LIB/LSTD_COMPILER.h"
00011 #include "../../LIB/LSTD_VALUES.h"
00012 #include "../../LIB/LSTD_BITMATH.h"
00013
00014 /* Header file guard */
00015 #ifndef _GPIO_interface_H
00016 #define _GPIO_interface_H
00017
00023 typedef struct
00024 {
00028     u8_t Port;
00032     u8_t Pin;
00036     u8_t Mode;
00040     u8_t OutputType;
00044     u8_t OutputSpeed;
00048     u8_t InputType;
00052     u8_t AF_Type;
00053
00054 } MGPIox_ConfigType;
00055
00056 /***** Functions prototypes *****/
00057 /*
00058             Functions prototypes
00059
00063 void MGPIox_vLockedPins(void);
00064
00071 void MGPIox_vSetPinMode(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8Mode);
00072
00079 void MGPIox_vSetPinOutputType(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8OutputType);
00080
00087 void MGPIox_vSetPinOutputSpeed(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8OutputSpeed);
00088
00095 void MGPIox_vSetPinInputPullType(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8InputPullType);
00096
00103 u8_t MGPIox_u8GetPinValue(u8_t A_u8PortID, u8_t A_u8PinID);
00104
00111 void MGPIox_vSetPinValue(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8PinValue);
00112
00119 void MGPIox_vSetResetAtomic(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8SetResetPinValue);
00120
00127 void MGPIox_vSetAlternateFunctionON(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8AFID);
00128
00133 void MGPIox_vSetPortConfigLock(u8_t A_u8PortID);
00134
00139 void MGPIox_vInit(MGPIox_ConfigType * A_xPinConfig);
00140
00146 void MGPIox_vTogglePinValue(u8_t A_u8PortID, u8_t A_u8PinID);
00147
00153 void GPIO_vSetNibbleLowValue(u8_t A_u8PortID, u16_t A_u16PortValue);
00154
00155 /***** Interfacing macros *****/
00156 /*
00157             Interfacing macros
00158
00170 #define GPIOx_MODE_INPUT (0b00)
00171
00177 #define GPIOx_MODE_OUTPUT (0b01)
00178
00184 #define GPIOx_MODE_AF (0b10)
00185
00191 #define GPIOx_MODE_ANALOG (0b11)
00192
```

```
00206 #define GPIO_PORTA (0)
00207
00213 #define GPIO_PORTB (1)
00214
00220 #define GPIO_PORTC (2)
00221
00235 #define GPIOx_OPENDRAIN (1)
00236
00242 #define GPIOx_PUSHPULL (2)
00243
00257 #define GPIOx_LowSpeed (0b00)
00258
00264 #define GPIOx_MediumSpeed (0b01)
00265
00271 #define GPIOx_HighSpeed (0b10)
00272
00278 #define GPIOx_VeryHighSpeed (0b11)
00279
00293 #define GPIOx_NoPull (0b00)
00294
00300 #define GPIOx_PullUp (0b01)
00301
00307 #define GPIOx_PullDown (0b10)
00308
00322 #define GPIOx_HIGH (1)
00323
00329 #define GPIOx_LOW (2)
00330
00344 #define GPIOx_PIN0 (0)
00345
00351 #define GPIOx_PIN1 (1)
00352
00358 #define GPIOx_PIN2 (2)
00359
00365 #define GPIOx_PIN3 (3)
00366
00372 #define GPIOx_PIN4 (4)
00373
00379 #define GPIOx_PIN5 (5)
00380
00386 #define GPIOx_PIN6 (6)
00387
00393 #define GPIOx_PIN7 (7)
00394
00400 #define GPIOx_PIN8 (8)
00401
00407 #define GPIOx_PIN9 (9)
00408
00414 #define GPIOx_PIN10 (10)
00415
00421 #define GPIOx_PIN11 (11)
00422
00428 #define GPIOx_PIN12 (12)
00429
00435 #define GPIOx_PIN13 (13)
00436
00442 #define GPIOx_PIN14 (14)
00443
00449 #define GPIOx_PIN15 (15)
00450
00464 #define GPIOx_AF0 (0)
00465
00471 #define GPIOx_AF1 (1)
00472
00478 #define GPIOx_AF2 (2)
00479
00485 #define GPIOx_AF3 (3)
00486
00492 #define GPIOx_AF4 (4)
00493
00499 #define GPIOx_AF5 (5)
00500
00506 #define GPIOx_AF6 (6)
00507
00513 #define GPIOx_AF7 (7)
00514
00520 #define GPIOx_AF8 (8)
00521
00527 #define GPIOx_AF9 (9)
00528
00534 #define GPIOx_AF10 (10)
00535
00541 #define GPIOx_AF11 (11)
00542
00548 #define GPIOx_AF12 (12)
00549
00555 #define GPIOx_AF13 (13)
```

```
00556  
00562 #define GPIOx_AF14 (14)  
00563  
00569 #define GPIOx_AF15 (15)  
00570  
00577 #define GPIOA_LOW_NIBBLE_HIGH 0xFF  
00578  
00583 #define GPIOA_LOW_NIBBLE_LOW 0x00  
00584  
00585 #endif // _GPIO_interface_H
```

7.99 COTS/MCAL/GPIO(GPIO_private.h File Reference)

This file contains the registers information and addresses for the GPIO module.

Data Structures

- struct [GPIOx_MemoryMapType](#)
GPIO declaration structure for its registers.

Macros

- #define [GPIOA_BASE_ADDRESS](#) (0x40020000)
- #define [GPIOB_BASE_ADDRESS](#) (0x40020400)
- #define [GPIOC_BASE_ADDRESS](#) (0x40020800)
- #define [GPIOA](#) ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOA_BASE_ADDRESS))
- #define [GPIOB](#) ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOB_BASE_ADDRESS))
- #define [GPIOC](#) ((volatile P2VAR(GPIOx_MemoryMapType))(GPIOC_BASE_ADDRESS))

7.99.1 Detailed Description

This file contains the registers information and addresses for the GPIO module.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

08/22/2022

Definition in file [GPIO_private.h](#).

7.100 GPIO_private.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
0010 #ifndef _GPIO_private_H
0011 #define _GPIO_private_H
0012
0018 typedef struct
0019 {
0023     u32_t MODERx;
0024
0028     u32_t OTYPERx;
0029
0033     u32_t OSPEEDRx;
0034
0038     u32_t PUPDRx;
0039
0043     u32_t IDRx;
0044
0048     u32_t ODRx;
0049
0053     u32_t BSRRx;
0054
0058     u32_t LCKRx;
0059
0063     u32_t AFRLx;
0064
0068     u32_t AFRHx;
0069
0070 } GPIOx_MemoryMapType;
0071
0083 #define GPIOA_BASE_ADDRESS (0x40020000)
0084
0090 #define GPIOB_BASE_ADDRESS (0x40020400)
0091
0097 #define GPIOC_BASE_ADDRESS (0x40020800)
0098
0012 #define GPIOA ((volatile P2VAR(GPIOx_MemoryMapType)) (GPIOA_BASE_ADDRESS))
0013
0019 #define GPIOB ((volatile P2VAR(GPIOx_MemoryMapType)) (GPIOB_BASE_ADDRESS))
00120
00126 #define GPIOC ((volatile P2VAR(GPIOx_MemoryMapType)) (GPIOC_BASE_ADDRESS))
00127
00130 #endif // _GPIO_private_H

```

7.101 COTS/MCAL/GPIO/GPIO_program.c File Reference

This file contains the source code of the interfacing for the GPIO modules.

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "GPIO_interface.h"
#include "GPIO_private.h"
#include "GPIO_config.h"

```

Functions

- void **MGPIOx_vLockedPins** (void)
Locks the prohibited GPIO PINs.
- void **MGPIOx_vSetPinMode** (**u8_t** A_u8PortID, **u8_t** A_u8PinID, **u8_t** A_u8Mode)
Sets a certain pin's mode on a specific port.
- void **MGPIOx_vSetPinOutputType** (**u8_t** A_u8PortID, **u8_t** A_u8PinID, **u8_t** A_u8OutputType)
Sets a certain pin's output type on a specific port.

- void `MGPIOx_vSetPinOutputSpeed (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8OutputSpeed)`
Sets a certain pin's output speed on a specific port.
- void `MGPIOx_vSetPinInputPullType (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8InputPullType)`
Sets a certain pin's input pull type on a specific port.
- `u8_t MGPIOx_u8GetPinValue (u8_t A_u8PortID, u8_t A_u8PinID)`
Gets the value currently on a certain pin.
- void `MGPIOx_vSetPinValue (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8PinValue)`
Sets a certain pin's output value on a specific port.
- void `MGPIOx_vSetResetAtomic (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8SetResetPinValue)`
Resets a certain pin's output value on a specific port.
- void `MGPIOx_vSetAlternateFunctionON (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8AFID)`
Applies an alternative function on a certain pin.
- void `MGPIOx_vInit (MGPIOx_ConfigType *A_xPinConfig)`
Initialize the GPIO with a certain configuration.
- void `MGPIOx_vTogglePinValue (u8_t A_u8PortID, u8_t A_u8PinID)`
Toggles a certain's pin's value on a certain port.
- void `GPIO_vSetNibbleLowValue (u8_t A_u8PortID, u16_t A_u16PortValue)`
Sets a certain port's output value.

7.101.1 Detailed Description

This file contains the source code of the interfacing for the GPIO modules.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

08/22/2022

Definition in file [GPIO_program.c](#).

7.101.2 Function Documentation

7.101.2.1 MGPIox_vLockedPins()

```
void MGPIox_vLockedPins (
    void )
```

Locks the prohibited GPIO PINs.

Definition at line 25 of file [GPIO_program.c](#).

```
00026 {
00027
00028     VAR(volatile u32_t) L_u32LockGPIOA = INITIAL_ZERO ;
00029
00030     /* Lock key write sequence */
00031
00032     /* WR LCKR[16] = '1' + LCKR[13,14,15] = ♦1♦ */
00033     L_u32LockGPIOA = ( (1UL << LCKK_BIT_POS) | (GPIOA_PIN_POS) ) ;
00034
00035     GPIOA->LCKRx = L_u32LockGPIOA ;
00036
00037     /* WR LCKR[16] = '0' + LCKR[13,14,15] should not change*/
00038     GPIOA->LCKRx = (GPIOA_PIN_POS) ;
00039
00040     /* WR LCKR[16] = '1'+ LCKR[13,14,15] should not change*/
00041     GPIOA->LCKRx = L_u32LockGPIOA ;
00042
00043     /* RD LCKR */
00044     L_u32LockGPIOA = GPIOA->LCKRx ;
00045
00046
00047     VAR(volatile u32_t) L_u32LockGPIOB = INITIAL_ZERO ;
00048
00049     /* Lock key write sequence */
00050
00051     /* WR LCKR[16] = '1' + LCKR[2,3,4,11] = ♦1♦ */
00052     L_u32LockGPIOB = ( (1UL << LCKK_BIT_POS) | (GPIOB_PIN_POS) ) ;
00053
00054     GPIOB->LCKRx = L_u32LockGPIOB ;
00055
00056     /* WR LCKR[16] = '0' + LCKR[2,3,4,11] should not change*/
00057     GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00058
00059     /* WR LCKR[16] = '1' + LCKR[2,3,4,11] should not change*/
00060     GPIOB->LCKRx = L_u32LockGPIOB ;
00061
00062     /* RD LCKR */
00063     L_u32LockGPIOB = GPIOB->LCKRx ;
00064
00065
00066 }
```

References [GPIOA](#), [GPIOA_PIN_POS](#), [GPIOB](#), [GPIOB_PIN_POS](#), [INITIAL_ZERO](#), [LCKK_BIT_POS](#), and [VAR](#).

7.101.2.2 MGPIox_vSetPinMode()

```
void MGPIox_vSetPinMode (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8Mode )
```

Sets a certain pin's mode on a speific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8Mode</i>	The mode to apply the pin

Definition at line 71 of file [GPIO_program.c](#).

```
00072 {
00073     switch (A_u8PortID)
00074     {
00075         case GPIO_PORTA:
00076             CLR_BITS(GPIOA->MODERx, 0b11, A_u8PinID, 2);
00077             SET_BITS(GPIOA->MODERx, A_u8Mode, A_u8PinID, 2);
00078             break;
00079
00080         case GPIO_PORTB:
00081             CLR_BITS(GPIOB->MODERx, 0b11, A_u8PinID, 2);
00082             SET_BITS(GPIOB->MODERx, A_u8Mode, A_u8PinID, 2);
00083             break;
00084
00085         case GPIO_PORTC:
00086             CLR_BITS(GPIOC->MODERx, 0b11, A_u8PinID, 2);
00087             SET_BITS(GPIOC->MODERx, A_u8Mode, A_u8PinID, 2);
00088             break;
00089
00090     }
00091 }
00092 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITs](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.101.2.3 MGPIOx_vSetPinOutputType()

```
void MGPIOx_vSetPinOutputType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputType )
```

Sets a certain pin's output type on a speifc port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8OutputType</i>	The output type to apply on the pin

Definition at line 97 of file [GPIO_program.c](#).

```
00098 {
00099
00100     switch (A_u8OutputType)
00101     {
00102         case GPIOx_OPENDRAIN:
00103             switch (A_u8PortID)
00104             {
00105                 case GPIO_PORTA:
00106                     SET_BIT(GPIOA->OTYPERx, A_u8PinID);
00107                     break;
00108
00109                 case GPIO_PORTB:
00110                     SET_BIT(GPIOB->OTYPERx, A_u8PinID);
00111                     break;
00112
00113                 case GPIO_PORTC:
00114                     SET_BIT(GPIOC->OTYPERx, A_u8PinID);
00115                     break;
00116             }
00117             break;
00118
00119         case GPIOx_PUSH_PULL:
00120             switch (A_u8PortID)
00121             {
00122                 case GPIO_PORTA:
```

```

00123     CLR_BIT(GPIOA->OTYPERx, A_u8PinID);
00124     break;
00125
00126     case GPIO_PORTB:
00127         CLR_BIT(GPIOB->OTYPERx, A_u8PinID);
00128         break;
00129
00130     case GPIO_PORTC:
00131         CLR_BIT(GPIOC->OTYPERx, A_u8PinID);
00132         break;
00133     }
00134     break;
00135 }
00136
00137 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_OPENDRAIN](#), [GPIOx_PUSH_PULL](#), and [SET_BIT](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.101.2.4 MGPIOx_vSetPinOutputSpeed()

```

void MGPIOx_vSetPinOutputSpeed (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputSpeed )
```

Sets a certain pin's output speed on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8OutputSpeed</i>	The output speed to apply on the pin

Definition at line 142 of file [GPIO_program.c](#).

```

00143 {
00144     switch(A_u8PortID)
00145     {
00146         case GPIO_PORTA:
00147             CLR_BITS(GPIOA->OSPEEDRx, 0b11, A_u8PinID, 2);
00148             SET_BITS(GPIOA->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00149             break;
00150
00151         case GPIO_PORTB:
00152             CLR_BITS(GPIOB->OSPEEDRx, 0b11, A_u8PinID, 2);
00153             SET_BITS(GPIOB->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00154             break;
00155
00156         case GPIO_PORTC:
00157             CLR_BITS(GPIOC->OSPEEDRx, 0b11, A_u8PinID, 2);
00158             SET_BITS(GPIOC->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00159             break;
00160     }
00161 }
00162
00163 }
```

References [CLR_BITS](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITS](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.101.2.5 MGPIox_vSetPinInputPullType()

```
void MGPIox_vSetPinInputPullType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8InputPullType )
```

Sets a certain pin's input pull type on a specific port.

Parameters

in	A_u8PortID	The port that the pin belongs to
in	A_u8PinID	The pin to update its mode
in	A_u8InputPullType	The input pull type to apply on the pin

Definition at line 168 of file [GPIO_program.c](#).

```
00169 {
00170
00171     switch (A_u8PortID)
00172     {
00173         case GPIO_PORTA:
00174             CLR_BITS(GPIOA->PUPDRx, 0b11, A_u8PinID, 2);
00175             SET_BITS(GPIOA->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00176             break;
00177
00178         case GPIO_PORTB:
00179             CLR_BITS(GPIOB->PUPDRx, 0b11, A_u8PinID, 2);
00180             SET_BITS(GPIOB->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00181             break;
00182
00183         case GPIO_PORTC:
00184             CLR_BITS(GPIOC->PUPDRx, 0b11, A_u8PinID, 2);
00185             SET_BITS(GPIOC->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00186             break;
00187     }
00188 }
00189 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITs](#).

Referenced by [MGPIox_vInit\(\)](#).

7.101.2.6 MGPIox_u8GetPinValue()

```
u8_t MGPIox_u8GetPinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Gets the value currently on a certain pin.

Parameters

in	A_u8PortID	The port that the pin belongs to
in	A_u8PinID	The pin to update its mode

Returns

The current value on the pin

Definition at line 194 of file [GPIO_program.c](#).

```
00195 {
00196     u8_t L_u8PinValue = INITIAL_ZERO;
00197     switch (A_u8PortID)
00198     {
00199         case GPIO_PORTA:
00200             L_u8PinValue = GET_BIT(GPIOA->IDRx, A_u8PinID);
00201             break;
00202         case GPIO_PORTB:
00203             L_u8PinValue = GET_BIT(GPIOB->IDRx, A_u8PinID);
00204             break;
00205         case GPIO_PORTC:
00206             L_u8PinValue = GET_BIT(GPIOC->IDRx, A_u8PinID);
00207             break;
00208     }
00209     return L_u8PinValue;
00210 }
00211
00212 }
```

References [GET_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [INITIAL_ZERO](#).

7.101.2.7 MGPIox_vSetValue()

```
void MGPIox_vSetValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8PinValue )
```

Sets a certain pin's output value on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8PinValue</i>	The set value to set on the pin

Definition at line 221 of file [GPIO_program.c](#).

```
00222 {
00223     switch (A_u8PinValue)
00224     {
00225         case GPIOx_HIGH:
00226             switch (A_u8PortID)
00227             {
00228                 case GPIO_PORTA:
00229                     SET_BIT(GPIOA->ODRx, A_u8PinID);
00230                     break;
00231
00232                 case GPIO_PORTB:
00233                     SET_BIT(GPIOB->ODRx, A_u8PinID);
00234                     break;
00235
00236                 case GPIO_PORTC:
00237                     SET_BIT(GPIOC->ODRx, A_u8PinID);
00238                     break;
00239             }
00240             break;
00241
00242 }
```

```

00243     case GPIOx_LOW:
00244         switch( A_u8PortID )
00245     {
00246         case GPIO_PORTA:
00247             CLR_BIT(GPIOA->ODRx, A_u8PinID);
00248             break;
00249
00250         case GPIO_PORTB:
00251             CLR_BIT(GPIOB->ODRx, A_u8PinID);
00252             break;
00253
00254         case GPIO_PORTC:
00255             CLR_BIT(GPIOC->ODRx, A_u8PinID);
00256             break;
00257     }
00258     break;
00259 }
00260 }
00261 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), [GPIOx_LOW](#), and [SET_BIT](#).

Referenced by [HBUZZER_vSoundOff\(\)](#), [HBUZZER_vSoundOn\(\)](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), and [HDCM_vStopMotor\(\)](#).

7.101.2.8 MGPIOp_vSetResetAtomic()

```

void MGPIOp_vSetResetAtomic (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8SetResetPinValue )
```

Resets a certain pin's output value on a specific port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8SetResetPinValue</i>	The reset value to set on the pin

Definition at line 266 of file [GPIO_program.c](#).

```

00267 {
00268
00269     switch (A_u8PortID)
00270     {
00271
00272         case GPIO_PORTA:
00273
00274             switch (A_u8SetResetPinValue)
00275             {
00276                 case GPIOx_HIGH:
00277                     GPIOA->BSRRx = (1 << A_u8PinID);
00278                     break;
00279
00280                 case GPIOx_LOW:
00281                     GPIOA->BSRRx = (1 << (A_u8PinID + 16));
00282                     break;
00283             }
00284
00285             break;
00286
00287         case GPIO_PORTB:
00288
00289             switch (A_u8SetResetPinValue)
00290             {
00291                 case GPIOx_HIGH:
```

```

00292     GPIOB->BSRRx = (1 << A_u8PinID);
00293     break;
00294
00295     case GPIOx_LOW:
00296     GPIOB->BSRRx = (1 << (A_u8PinID + 16));
00297     break;
00298 }
00299
00300     break;
00301
00302     case GPIO_PORTC:
00303
00304     switch (A_u8SetResetPinValue)
00305     {
00306         case GPIOx_HIGH:
00307         GPIOC->BSRRx = (1 << A_u8PinID);
00308         break;
00309
00310         case GPIOx_LOW:
00311         GPIOC->BSRRx = (1 << (A_u8PinID + 16));
00312         break;
00313     }
00314
00315     break;
00316 }
00317
00318 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), and [GPIOx_LOW](#).

Referenced by [HULTSNC_vTrigger\(\)](#).

7.101.2.9 MGPIOp_vSetAlternateFunctionON()

```
void MGPIOp_vSetAlternateFunctionON (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8AFID )
```

Applies an alternative function on a certain pin.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to update its mode
in	<i>A_u8AFID</i>	The alternative function to apply on the pin

Definition at line 323 of file [GPIO_program.c](#).

```

00324 {
00325
00326     switch (A_u8PortID)
00327     {
00328         case GPIO_PORTA:
00329             switch (A_u8PinID)
00330             {
00331                 case GPIOx_PIN0:
00332                 case GPIOx_PIN1:
00333                 case GPIOx_PIN2:
00334                 case GPIOx_PIN3:
00335                 case GPIOx_PIN4:
00336                 case GPIOx_PIN5:
00337                 case GPIOx_PIN6:
00338                 case GPIOx_PIN7:
00339                     CLR_BITS(GPIOA->AFRLx, 0b1111, A_u8PinID, 4);
00340                     SET_BITS(GPIOA->AFRLx, A_u8AFID, A_u8PinID, 4);
00341                     break;
```

```

00342             case GPIOx_PIN8:
00343             case GPIOx_PIN9:
00344             case GPIOx_PIN10:
00345             case GPIOx_PIN11:
00346             case GPIOx_PIN12:
00347             case GPIOx_PIN13:
00348             case GPIOx_PIN14:
00349             case GPIOx_PIN15:
00350                 CLR_BITS(GPIOA->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00351                 SET_BITS(GPIOA->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00352                 break;
00353             }
00354         break;
00356
00357     case GPIO_PORTB:
00358         switch (A_u8PinID)
00359     {
00360         case GPIOx_PIN0:
00361         case GPIOx_PIN1:
00362         case GPIOx_PIN2:
00363         case GPIOx_PIN3:
00364         case GPIOx_PIN4:
00365         case GPIOx_PIN5:
00366         case GPIOx_PIN6:
00367         case GPIOx_PIN7:
00368             CLR_BITS(GPIOB->AFRLx, 0b1111, A_u8PinID, 4);
00369             SET_BITS(GPIOB->AFRLx, A_u8AFID, A_u8PinID, 4);
00370             break;
00371
00372         case GPIOx_PIN8:
00373         case GPIOx_PIN9:
00374         case GPIOx_PIN10:
00375         case GPIOx_PIN11:
00376         case GPIOx_PIN12:
00377         case GPIOx_PIN13:
00378         case GPIOx_PIN14:
00379         case GPIOx_PIN15:
00380             CLR_BITS(GPIOB->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00381             SET_BITS(GPIOB->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00382             break;
00383     }
00384     break;
00385
00386     case GPIO_PORTC:
00387         switch (A_u8PinID)
00388     {
00389         case GPIOx_PIN0:
00390         case GPIOx_PIN1:
00391         case GPIOx_PIN2:
00392         case GPIOx_PIN3:
00393         case GPIOx_PIN4:
00394         case GPIOx_PIN5:
00395         case GPIOx_PIN6:
00396         case GPIOx_PIN7:
00397             CLR_BITS(GPIOC->AFRLx, 0b1111, A_u8PinID, 4);
00398             SET_BITS(GPIOC->AFRLx, A_u8AFID, A_u8PinID, 4);
00399             break;
00400
00401         case GPIOx_PIN8:
00402         case GPIOx_PIN9:
00403         case GPIOx_PIN10:
00404         case GPIOx_PIN11:
00405         case GPIOx_PIN12:
00406         case GPIOx_PIN13:
00407         case GPIOx_PIN14:
00408         case GPIOx_PIN15:
00409             CLR_BITS(GPIOC->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00410             SET_BITS(GPIOC->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00411             break;
00412     }
00413     break;
00414 }
00415
00416 }
```

References [CLR_BITS](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_PIN0](#), [GPIOx_PIN1](#), [GPIOx_PIN10](#), [GPIOx_PIN11](#), [GPIOx_PIN12](#), [GPIOx_PIN13](#), [GPIOx_PIN14](#), [GPIOx_PIN15](#), [GPIOx_PIN2](#), [GPIOx_PIN3](#), [GPIOx_PIN4](#), [GPIOx_PIN5](#), [GPIOx_PIN6](#), [GPIOx_PIN7](#), [GPIOx_PIN8](#), [GPIOx_PIN9](#), and [SET_BITS](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.101.2.10 MGPIOx_vInit()

```
void MGPIOx_vInit (
    MGPIOx_ConfigType * A_xPinConfig )
```

Initialize the GPIO with a certain configuration.

Parameters

in	A_xPinConfig	The initialization configuration for the GPIO
----	--------------	---

Definition at line 421 of file [GPIO_program.c](#).

```
00422 {
00423
00424     MGPIOx_vSetPinMode           (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->Mode
00425     );
00426     MGPIOx_vSetPinOutputType    (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputType
00427     );
00428     MGPIOx_vSetPinOutputSpeed   (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputSpeed
00429     );
00430     MGPIOx_vSetPinInputPullType (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->InputType
00431     );
00432     MGPIOx_vSetAlternateFunctionON (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->AF_Type
00433     );
00434 }
```

References [MGPIOx_ConfigType::AF_Type](#), [MGPIOx_ConfigType::InputType](#), [MGPIOx_vSetAlternateFunctionON\(\)](#), [MGPIOx_vSetPinInputPullType\(\)](#), [MGPIOx_vSetPinMode\(\)](#), [MGPIOx_vSetPinOutputSpeed\(\)](#), [MGPIOx_vSetPinOutputType\(\)](#), [MGPIOx_ConfigType::Mode](#), [MGPIOx_ConfigType::OutputSpeed](#), [MGPIOx_ConfigType::OutputType](#), [MGPIOx_ConfigType::Pin](#), and [MGPIOx_ConfigType::Port](#).

Referenced by [HBUZZER_vInit\(\)](#), [HDCM_vInitMotor\(\)](#), [HULTSNC_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.101.2.11 MGPIOx_vTogglePinValue()

```
void MGPIOx_vTogglePinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Toggles a certain's pin's value on a certain port.

Parameters

in	A_u8PortID	The port that the pin belongs to
in	A_u8PinID	The pin to toggle its value

Definition at line 435 of file [GPIO_program.c](#).

```
00436 {
00437
00438     switch (A_u8PortID)
00439     {
00440         case GPIO_PORTA:
00441             TOGGLE_BIT(GPIOA->ODRx, A_u8PinID);
00442             break;
00443
00444         case GPIO_PORTB:
00445             TOGGLE_BIT(GPIOB->ODRx, A_u8PinID);
00446             break;
```

```

00447
00448     case GPIO_PORTC:
00449         TOGGLE_BIT(GPIOC->ODRx, A_u8PinID);
00450         break;
00451     }
00452
00453 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [TOGGLE_BIT](#).

Referenced by [HBUZZER_vToggleSound\(\)](#).

7.101.2.12 GPIO_vSetNibbleLowValue()

```

void GPIO_vSetNibbleLowValue (
    u8_t A_u8PortID,
    u16_t A_u16PortValue )
```

Sets a certain port's output value.

Parameters

in	<i>A_u8PortID</i>	The port to out the value on
in	<i>A_u16PortValue</i>	The value to set on the port

Definition at line 458 of file [GPIO_program.c](#).

```

00459 {
00460
00461     switch(A_u8PortID)
00462     {
00463
00464         case GPIO_PORTA : GPIOA->ODRx = ( ((GPIOA->ODRx & PORTA_BIT_MANIPULATION) | (A_u16PortValue))
00465             << 0 ) ; break; //Start writing from PIN0 to PIN7.
00466
00467         case GPIO_PORTB : GPIOB->ODRx = ( ((GPIOB->ODRx & PORTB_BIT_MANIPULATION) | (A_u16PortValue))
00468             << 5 ) ; break; //Start writing from PIN5 to PIN12.
00469
00470         case GPIO_PORTC : GPIOC->ODRx ; break; // You shouldn't write on this port.
00471
00472     default :
00473         break;
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [PORTA_BIT_MANIPULATION](#), and [PORTB_BIT_MANIPULATION](#).

7.102 GPIO_program.c

[Go to the documentation of this file.](#)

```

00001 /*************************************************************************/
00002 /* Include headers */
00003 /*************************************************************************/
00004 #include "../../LIB/LSTD_TYPES.h"
00005 #include "../../LIB/LSTD_COMPILER.h"
00006 #include "../../LIB/LSTD_VALUES.h"
00007 #include "../../LIB/LSTD_BITMATH.h"
00008
00009 #include "GPIO_interface.h"
00010 #include "GPIO_private.h"
00011 #include "GPIO_config.h"
```

```

00020
00021 /***** Functions' implementations *****/
00022 /* Functions' implementations */
00023 /***** Functions' implementations *****/
00024
00025 void MGPIOx_vLockedPins(void)
00026 {
00027     VAR(volatile u32_t) L_u32LockGPIOA = INITIAL_ZERO ;
00028
00029     /* Lock key write sequence */
00030
00031     /* WR LCKR[16] = '1' + LCKR[13,14,15] = ◆1◆ */
00032     L_u32LockGPIOA = ( (1UL << LCKK_BIT_POS) | (GPIOA_PIN_POS) ) ;
00033
00034     GPIOA->LCKRx = L_u32LockGPIOA ;
00035
00036     /* WR LCKR[16] = '0' + LCKR[13,14,15] should not change*/
00037     GPIOA->LCKRx = (GPIOA_PIN_POS) ;
00038
00039     /* WR LCKR[16] = '1'+ LCKR[13,14,15] should not change*/
00040     GPIOA->LCKRx = L_u32LockGPIOA ;
00041
00042     /* RD LCKR */
00043     L_u32LockGPIOA = GPIOA->LCKRx ;
00044
00045
00046     VAR(volatile u32_t) L_u32LockGPIOB = INITIAL_ZERO ;
00047
00048     /* Lock key write sequence */
00049
00050     /* WR LCKR[16] = '1' + LCKR[2,3,4,11] = ◆1◆ */
00051     L_u32LockGPIOB = ( (1UL << LCKK_BIT_POS) | (GPIOB_PIN_POS) ) ;
00052
00053     GPIOB->LCKRx = L_u32LockGPIOB ;
00054
00055     /* WR LCKR[16] = '0' + LCKR[2,3,4,11] should not change*/
00056     GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00057
00058     /* WR LCKR[16] = '1'+ LCKR[2,3,4,11] should not change*/
00059     GPIOB->LCKRx = L_u32LockGPIOB ;
00060
00061     /* RD LCKR */
00062     L_u32LockGPIOB = GPIOB->LCKRx ;
00063
00064 }
00065
00066 }
00067
00068 /***** Functions' implementations *****/
00069 /***** Functions' implementations *****/
00070
00071 void MGPIOx_vSetPinMode(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8Mode)
00072 {
00073
00074     switch (A_u8PortID)
00075     {
00076         case GPIO_PORTA:
00077             CLR_BITS(GPIOA->MODERx, 0b11, A_u8PinID, 2);
00078             SET_BITS(GPIOA->MODERx, A_u8Mode, A_u8PinID, 2);
00079             break;
00080
00081         case GPIO_PORTB:
00082             CLR_BITS(GPIOB->MODERx, 0b11, A_u8PinID, 2);
00083             SET_BITS(GPIOB->MODERx, A_u8Mode, A_u8PinID, 2);
00084             break;
00085
00086         case GPIO_PORTC:
00087             CLR_BITS(GPIOC->MODERx, 0b11, A_u8PinID, 2);
00088             SET_BITS(GPIOC->MODERx, A_u8Mode, A_u8PinID, 2);
00089             break;
00090     }
00091
00092 }
00093
00094 /***** Functions' implementations *****/
00095 /***** Functions' implementations *****/
00096
00097 void MGPIOx_vSetPinOutputType(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8OutputType)
00098 {
00099
00100     switch (A_u8OutputType)
00101     {
00102         case GPIOx_OPENDRAIN:
00103             switch (A_u8PortID)
00104             {
00105                 case GPIO_PORTA:
00106                     SET_BIT(GPIOA->OTYPERx, A_u8PinID);
00107                     break;

```

```

00108
00109     case GPIO_PORTB:
00110         SET_BIT(GPIOB->OTYPERx, A_u8PinID);
00111         break;
00112
00113     case GPIO_PORTC:
00114         SET_BIT(GPIOC->OTYPERx, A_u8PinID);
00115         break;
00116     }
00117     break;
00118
00119 case GPIO_PUSH_PULL:
00120     switch(A_u8PortID)
00121     {
00122         case GPIO_PORTA:
00123             CLR_BIT(GPIOA->OTYPERx, A_u8PinID);
00124             break;
00125
00126         case GPIO_PORTB:
00127             CLR_BIT(GPIOB->OTYPERx, A_u8PinID);
00128             break;
00129
00130         case GPIO_PORTC:
00131             CLR_BIT(GPIOC->OTYPERx, A_u8PinID);
00132             break;
00133     }
00134     break;
00135 }
00136
00137 }
00138
00139 /*****
00140 *****/
00141
00142 void MGPIox_vSetPinOutputSpeed(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8OutputSpeed)
00143 {
00144
00145     switch(A_u8PortID)
00146     {
00147         case GPIO_PORTA:
00148             CLR_BITS(GPIOA->OSPEEDRx, 0b11, A_u8PinID, 2);
00149             SET_BITS(GPIOA->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00150             break;
00151
00152         case GPIO_PORTB:
00153             CLR_BITS(GPIOB->OSPEEDRx, 0b11, A_u8PinID, 2);
00154             SET_BITS(GPIOB->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00155             break;
00156
00157         case GPIO_PORTC:
00158             CLR_BITS(GPIOC->OSPEEDRx, 0b11, A_u8PinID, 2);
00159             SET_BITS(GPIOC->OSPEEDRx, A_u8OutputSpeed, A_u8PinID, 2);
00160             break;
00161     }
00162
00163 }
00164
00165 /*****
00166 *****/
00167
00168 void MGPIox_vSetPinInputPullType(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8InputPullType)
00169 {
00170
00171     switch (A_u8PortID)
00172     {
00173         case GPIO_PORTA:
00174             CLR_BITS(GPIOA->PUPDRx, 0b11, A_u8PinID, 2);
00175             SET_BITS(GPIOA->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00176             break;
00177
00178         case GPIO_PORTB:
00179             CLR_BITS(GPIOB->PUPDRx, 0b11, A_u8PinID, 2);
00180             SET_BITS(GPIOB->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00181             break;
00182
00183         case GPIO_PORTC:
00184             CLR_BITS(GPIOC->PUPDRx, 0b11, A_u8PinID, 2);
00185             SET_BITS(GPIOC->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00186             break;
00187     }
00188
00189 }
00190
00191 /*****
00192 *****/
00193
00194 u8_t MGPIox_u8GetValue(u8_t A_u8PortID, u8_t A_u8PinID)

```

```

00195 {
00196     u8_t L_u8PinValue = INITIAL_ZERO;
00197
00198     switch (A_u8PortID)
00199     {
00200         case GPIO_PORTA:
00201             L_u8PinValue = GET_BIT(GPIOA->IDRx, A_u8PinID);
00202             break;
00203
00204         case GPIO_PORTB:
00205             L_u8PinValue = GET_BIT(GPIOB->IDRx, A_u8PinID);
00206             break;
00207
00208         case GPIO_PORTC:
00209             L_u8PinValue = GET_BIT(GPIOC->IDRx, A_u8PinID);
00210             break;
00211     }
00212 }
00213
00214     return L_u8PinValue;
00215
00216 }
00217
00218 /*****
00219 *****/
00220
00221 void MGPIox_vSetValue(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8PinValue)
00222 {
00223
00224     switch (A_u8PinValue)
00225     {
00226         case GPIOx_HIGH:
00227             switch (A_u8PortID)
00228             {
00229                 case GPIO_PORTA:
00230                     SET_BIT(GPIOA->ODRx, A_u8PinID);
00231                     break;
00232
00233                 case GPIO_PORTB:
00234                     SET_BIT(GPIOB->ODRx, A_u8PinID);
00235                     break;
00236
00237                 case GPIO_PORTC:
00238                     SET_BIT(GPIOC->ODRx, A_u8PinID);
00239                     break;
00240             }
00241             break;
00242
00243         case GPIOx_LOW:
00244             switch (A_u8PortID)
00245             {
00246                 case GPIO_PORTA:
00247                     CLR_BIT(GPIOA->ODRx, A_u8PinID);
00248                     break;
00249
00250                 case GPIO_PORTB:
00251                     CLR_BIT(GPIOB->ODRx, A_u8PinID);
00252                     break;
00253
00254                 case GPIO_PORTC:
00255                     CLR_BIT(GPIOC->ODRx, A_u8PinID);
00256                     break;
00257             }
00258             break;
00259     }
00260
00261 }
00262
00263 /*****
00264 *****/
00265
00266 void MGPIox_vSetResetAtomic(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8SetResetPinValue)
00267 {
00268
00269     switch (A_u8PortID)
00270     {
00271
00272         case GPIO_PORTA:
00273
00274             switch (A_u8SetResetPinValue)
00275             {
00276                 case GPIOx_HIGH:
00277                     GPIOA->BSRRx = (1 << A_u8PinID);
00278                     break;
00279
00280                 case GPIOx_LOW:
00281                     GPIOA->BSRRx = (1 << (A_u8PinID + 16));
00282             }
00283
00284         }
00285
00286     }
00287
00288 }
00289

```

```

00282         break;
00283     }
00284
00285     break;
00286
00287     case GPIO_PORTB:
00288
00289         switch (A_u8SetResetPinValue)
00290     {
00291         case GPIOx_HIGH:
00292             GPIOB->BSRRx = (1 << A_u8PinID);
00293             break;
00294
00295         case GPIOx_LOW:
00296             GPIOB->BSRRx = (1 << (A_u8PinID + 16));
00297             break;
00298     }
00299
00300     break;
00301
00302     case GPIO_PORTC:
00303
00304         switch (A_u8SetResetPinValue)
00305     {
00306         case GPIOx_HIGH:
00307             GPIOC->BSRRx = (1 << A_u8PinID);
00308             break;
00309
00310         case GPIOx_LOW:
00311             GPIOC->BSRRx = (1 << (A_u8PinID + 16));
00312             break;
00313     }
00314
00315     break;
00316 }
00317
00318 }
00319
00320 /***** *****/
00321 /***** *****/
00322
00323 void MGPIox_vSetAlternateFunctionON(u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8AFID)
00324 {
00325
00326     switch (A_u8PortID)
00327     {
00328         case GPIO_PORTA:
00329             switch (A_u8PinID)
00330         {
00331             case GPIOx_PIN0:
00332             case GPIOx_PIN1:
00333             case GPIOx_PIN2:
00334             case GPIOx_PIN3:
00335             case GPIOx_PIN4:
00336             case GPIOx_PIN5:
00337             case GPIOx_PIN6:
00338             case GPIOx_PIN7:
00339                 CLR_BITS(GPIOA->AFRLx, 0b1111, A_u8PinID, 4);
00340                 SET_BITS(GPIOA->AFRLx, A_u8AFID, A_u8PinID, 4);
00341                 break;
00342
00343             case GPIOx_PIN8:
00344             case GPIOx_PIN9:
00345             case GPIOx_PIN10:
00346             case GPIOx_PIN11:
00347             case GPIOx_PIN12:
00348             case GPIOx_PIN13:
00349             case GPIOx_PIN14:
00350             case GPIOx_PIN15:
00351                 CLR_BITS(GPIOA->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00352                 SET_BITS(GPIOA->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00353                 break;
00354         }
00355         break;
00356
00357         case GPIO_PORTB:
00358             switch (A_u8PinID)
00359         {
00360             case GPIOx_PIN0:
00361             case GPIOx_PIN1:
00362             case GPIOx_PIN2:
00363             case GPIOx_PIN3:
00364             case GPIOx_PIN4:
00365             case GPIOx_PIN5:
00366             case GPIOx_PIN6:
00367             case GPIOx_PIN7:
00368                 CLR_BITS(GPIOB->AFRLx, 0b1111, A_u8PinID, 4);

```

```

00369     SET_BITS(GPIOB->AFRLx, A_u8AFID, A_u8PinID, 4);
00370     break;
00371
00372     case GPIOx_PIN8:
00373     case GPIOx_PIN9:
00374     case GPIOx_PIN10:
00375     case GPIOx_PIN11:
00376     case GPIOx_PIN12:
00377     case GPIOx_PIN13:
00378     case GPIOx_PIN14:
00379     case GPIOx_PIN15:
00380         CLR_BITS(GPIOB->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00381         SET_BITS(GPIOB->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00382         break;
00383     }
00384     break;
00385
00386     case GPIO_PORTC:
00387         switch (A_u8PinID)
00388     {
00389         case GPIOx_PIN0:
00390         case GPIOx_PIN1:
00391         case GPIOx_PIN2:
00392         case GPIOx_PIN3:
00393         case GPIOx_PIN4:
00394         case GPIOx_PIN5:
00395         case GPIOx_PIN6:
00396         case GPIOx_PIN7:
00397             CLR_BITS(GPIOC->AFRLx, 0b1111, A_u8PinID, 4);
00398             SET_BITS(GPIOC->AFRLx, A_u8AFID, A_u8PinID, 4);
00399             break;
00400
00401         case GPIOx_PIN8:
00402         case GPIOx_PIN9:
00403         case GPIOx_PIN10:
00404         case GPIOx_PIN11:
00405         case GPIOx_PIN12:
00406         case GPIOx_PIN13:
00407         case GPIOx_PIN14:
00408         case GPIOx_PIN15:
00409             CLR_BITS(GPIOC->AFRHx, 0b1111, (A_u8PinID - 8), 4);
00410             SET_BITS(GPIOC->AFRHx, A_u8AFID, (A_u8PinID - 8), 4);
00411             break;
00412     }
00413     break;
00414 }
00415
00416 }
00417
00418 /*****
00419 *****/
00420
00421 void MGPIOx_vInit(MGPIOx_ConfigType * A_xPinConfig)
00422 {
00423
00424     MGPIOx_vSetPinMode          (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->Mode
00425 );
00426     MGPIOx_vSetPinOutputType    (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputType
00427 );
00428     MGPIOx_vSetPinOutputSpeed   (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputSpeed
00429 );
00430     MGPIOx_vSetPinInputPullType (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->InputType
00431 );
00432     MGPIOx_vSetAlternateFunctionON (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->AF_Type
00433 );
00434
00435 /*****
00436 *****/
00437
00438 void MGPIOx_vTogglePinValue(u8_t A_u8PortID, u8_t A_u8PinID)
00439 {
00440     switch (A_u8PortID)
00441     {
00442         case GPIO_PORTA:
00443             TOGGLE_BIT(GPIOA->ODRx, A_u8PinID);
00444             break;
00445
00446         case GPIO_PORTB:
00447             TOGGLE_BIT(GPIOB->ODRx, A_u8PinID);
00448             break;
00449
00450         case GPIO_PORTC:
00451             TOGGLE_BIT(GPIOC->ODRx, A_u8PinID);
00452             break;
00453     }
00454 }
```

```

00451      }
00452
00453 }
00454
00455 /*****
00456 *****/
00457
00458 void GPIO_vSetNibbleLowValue( u8_t A_u8PortID, ul6_t A_ul6PortValue )
00459 {
00460
00461     switch(A_u8PortID)
00462     {
00463
00464         case GPIO_PORTA : GPIOA->ODRx = ( ((GPIOA->ODRx & PORTA_BIT_MANIPULATION) | (A_ul6PortValue))
00465             << 0 ) ; break; //Start writing from PIN0 to PIN7.
00466
00467         case GPIO_PORTB : GPIOB->ODRx = ( ((GPIOB->ODRx & PORTB_BIT_MANIPULATION) | (A_ul6PortValue))
00468             << 5 ) ; break; //Start writing from PIN5 to PIN12.
00469
00470         case GPIO_PORTC : GPIOC->ODRx ; break; // You shouldn't write on this port.
00471     }
00472
00473 }
00474
00475 *****/
00476 *****/
00477
00478
00479

```

7.103 NVIC_config.h

```

00001 /* FILENAME: NVIC_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 08/25/2022
00005 */
00006 #ifndef _NVIC_config_H
00007 #define _NVIC_config_H
00008
00009
00010
00011
00012
00013 #endif // _NVIC_config_H

```

7.104 COTS/MCAL/NVIC/NVIC_interface.h File Reference

This file contains the interface information and addresses for the NVIC module.

Macros

- #define _16GROUP_NoSub_Priorities (0b011)
16 groups and 0 sub-groups
- #define _8GROUP_2Sub_Priorities (0b100)
8 groups and 2 sub-groups
- #define _4GROUP_4Sub_Priorities (0b101)
4 groups and 4 sub-groups
- #define _2GROUP_8Sub_Priorities (0b110)
2 groups and 8 sub-groups
- #define NoGROUP_16Sub_Priorities (0b111)
0 groups and 16 sub-groups
- #define GROUP_4BITS (0b011)

- #define GROUP_3BITS (0b100)
8 groups and 2 sub-groups
- #define GROUP_2BITS (0b101)
4 groups and 4 sub-groups
- #define GROUP_1BITS (0b110)
2 groups and 8 sub-groups
- #define GROUP_0BITS (0b111)
0 groups and 16 sub-groups
- #define NO_GROUP_PRIORITY (0)
No group priority.
- #define GROUP_PRIORITY_0 (0)
Priority: 0.
- #define GROUP_PRIORITY_1 (1)
Priority: 1.
- #define GROUP_PRIORITY_2 (2)
Priority: 2.
- #define GROUP_PRIORITY_3 (3)
Priority: 3.
- #define GROUP_PRIORITY_4 (4)
Priority: 4.
- #define GROUP_PRIORITY_5 (5)
Priority: 5.
- #define GROUP_PRIORITY_6 (6)
Priority: 6.
- #define GROUP_PRIORITY_7 (7)
Priority: 7.
- #define GROUP_PRIORITY_8 (8)
Priority: 8.
- #define GROUP_PRIORITY_9 (9)
Priority: 9.
- #define GROUP_PRIORITY_10 (10)
Priority: 10.
- #define GROUP_PRIORITY_11 (11)
Priority: 11.
- #define GROUP_PRIORITY_12 (12)
Priority: 12.
- #define GROUP_PRIORITY_13 (13)
Priority: 13.
- #define GROUP_PRIORITY_14 (14)
Priority: 14.
- #define GROUP_PRIORITY_15 (15)
Priority: 15.
- #define NO_SUB_PRIORITY (0)
No sub-group priority.
- #define SUB_PRIORITY_0 (0)
Priority: 0.
- #define SUB_PRIORITY_1 (1)
Priority: 1.
- #define SUB_PRIORITY_2 (2)
Priority: 2.

- #define **SUB_PRIORITY_3** (3)
Priority: 3.
- #define **SUB_PRIORITY_4** (4)
Priority: 4.
- #define **SUB_PRIORITY_5** (5)
Priority: 5.
- #define **SUB_PRIORITY_6** (6)
Priority: 6.
- #define **SUB_PRIORITY_7** (7)
Priority: 7.
- #define **SUB_PRIORITY_8** (8)
Priority: 8.
- #define **SUB_PRIORITY_9** (9)
Priority: 9.
- #define **SUB_PRIORITY_10** (10)
Priority: 10.
- #define **SUB_PRIORITY_11** (11)
Priority: 11.
- #define **SUB_PRIORITY_12** (12)
Priority: 12.
- #define **SUB_PRIORITY_13** (13)
Priority: 13.
- #define **SUB_PRIORITY_14** (14)
Priority: 14.
- #define **SUB_PRIORITY_15** (15)
Priority: 15.
- #define **WWDG** (0)
Window Watchdog (WWDG) Interrupt.
- #define **EXTI16** (1)
EXTI Line 16 interrupt / PVD through EXTI line detection interrupt.
- #define **EXTI21** (2)
EXTI Line 21 interrupt / Tamper andTimeStamp interrupts through the EXTI line.
- #define **EXTI22** (3)
EXTI Line 22 interrupt / RTC (Real-time clock) wakeup interrupt through the EXTI line.
- #define **FLASH** (4)
Flash global interrupt.
- #define **RCC** (5)
RCC global interrupt.
- #define **EXTI0** (6)
EXTI Line 0 interrupt.
- #define **EXTI1** (7)
EXTI Line 1 interrupt.
- #define **EXTI2** (8)
EXTI Line 2 interrupt.
- #define **EXTI3** (9)
EXTI Line 3 interrupt.
- #define **EXTI4** (10)
EXTI Line 4 interrupt.
- #define **DMA1_STREAM0** (11)
DMA1 (Direct Memory Access) Stream 0 global interrupt.
- #define **DMA1_STREAM1** (12)

- #define DMA1_STREAM2 (13)

DMA1 (Direct Memory Access) Steam 1 global interrupt.
- #define DMA1_STREAM3 (14)

DMA1 (Direct Memory Access) Steam 2 global interrupt.
- #define DMA1_STREAM4 (15)

DMA1 (Direct Memory Access) Steam 3 global interrupt.
- #define DMA1_STREAM5 (16)

DMA1 (Direct Memory Access) Steam 4 global interrupt.
- #define DMA1_STREAM6 (17)

DMA1 (Direct Memory Access) Steam 5 global interrupt.
- #define ADC (18)

ADC (Analog-To-Digital Converter) ADC1 global interrupt.
- #define EXTI9 (23)

EXTI Lines [9:5] interrupts.
- #define TIM1_BRK_TIM9 (24)

TIM1 (Timer 1) break interrupt and TIM9 (Timer 9) global interrupt.
- #define TIM1_UP_TIM10 (25)

TIM1 (Timer 1) update interrupt and TIM10 (Timer 10) global interrupt.
- #define TIM1_TRG_COM_TIM11 (26)

TIM1 (Timer 1) trigger and commutation itnerrupts and TIM11 (Timer 11) global interrupts.
- #define TIM1_CC (27)

TIM1 (Timer 1) capture compare interrupt.
- #define TIM2 (28)

TIM2 (Timer 2) global interrupt.
- #define TIM3 (29)

TIM3 (Timer 3) global interrupt.
- #define TIM4 (30)

TIM4 (Timer 4) global interrupt.
- #define I2C1_EV (31)

I2C1 (Inter-integrated Circuit 1) event interrupt.
- #define I2C1_ER (32)

I2C1 (Inter-integrated Circuit 1) error interrupt.
- #define I2C2_EV (33)

I2C2 (Inter-integrated Circuit 2) event interrupt.
- #define I2C2_ER (34)

I2C2 (Inter-integrated Circuit 2) error interrupt.
- #define SPI1 (35)

SPI1 (Serial Peripheral Interface 1) global interrupt.
- #define SPI2 (36)

SPI2 (Serial Peripheral Interface 2) global interrupt.
- #define USART1 (37)

USART1 (Universal Synchronous/Asynchronous Receiver/Transmitter 1) global interrupt.
- #define USART2 (38)

USART2 (Universal Synchronous/Asynchronous Receiver/Transmitter 2) global interrupt.
- #define EXTI15_10 (30)

EXTI Line[15:10] interrupts.
- #define EXTI17 (41)

EXTI Line 17 interrupt / RTC Alarms (A and B) through EXTI line interrupt.
- #define EXTI18 (42)

EXTI Line 18 interrupt / USB On-The-Go FS Wakeup through EXTI line interrupt.

- `#define DMA1_STREAM7` (47)
DMA1 (Direct Memory Access) Steam 7 global interrupt.
- `#define SDIO` (49)
SDIO (Secure Digital Input Output) global interrupt.
- `#define TIM5` (50)
TIM5 (Timer 5) global interrupt.
- `#define SPI3` (51)
SPI3 (Serial Peripheral Interface 3) global interrupt.
- `#define DMA2_STREAM0` (56)
DMA2 (Direct Memory Access 2) Steam 0 global interrupt.
- `#define DMA2_STREAM1` (57)
DMA2 (Direct Memory Access 2) Steam 1 global interrupt.
- `#define DMA2_STREAM2` (58)
DMA2 (Direct Memory Access 2) Steam 2 global interrupt.
- `#define DMA2_STREAM3` (59)
DMA2 (Direct Memory Access 2) Steam 3 global interrupt.
- `#define DMA2_STREAM4` (60)
DMA2 (Direct Memory Access 2) Steam 4 global interrupt.
- `#define OTG_FS` (67)
USB On The Go FS global interrupt.
- `#define DMA2_STREAM5` (68)
DMA2 (Direct Memory Access 2) Steam 5 global interrupt.
- `#define DMA2_STREAM6` (69)
DMA2 (Direct Memory Access 2) Steam 6 global interrupt.
- `#define DMA2_STREAM7` (70)
DMA2 (Direct Memory Access 2) Steam 7 global interrupt.
- `#define USART6` (71)
USART6 (Universal Synchronous/Asynchronous Receiver/Transmitter 6) global interrupt.
- `#define I2C3_EV` (72)
I2C3 (Inter-integrated Circuit 3) event interrupt.
- `#define I2C3_ER` (73)
I2C3 (Inter-integrated Circuit 3) error interrupt.
- `#define FPU` (81)
FPU (Floating Point Unit) global interrupt.
- `#define SPI4` (84)
SPI4 (Serial Peripheral Interface 4) global interrupt.

Functions

- `void MNVIC_vEnablePeriphral (u8_t A_u8INTID)`
Enable the interrupt of a certain peripheral in NVIC.
- `void MNVIC_vDisablePeriphral (u8_t A_u8INTID)`
Disable the interrupt of a certain peripheral in NVIC.
- `void MNVIC_vSetPendingFlag (u8_t A_u8INTID)`
Set the pending flag of the peripheral.
- `void MNVIC_vClearPendingFlag (u8_t A_u8INTID)`
Clear the pending flag of the peripheral.
- `u8_t MNVIC_u8GetActive (u8_t A_u8INTID)`
Get the current state of the interrupt active flag of a certain peripheral in NVIC.
- `void MNVIC_vSetPriorityConfig (u8_t A_u8PriorityOption)`

- Configures the group and sub-group priority configuration.*
- void [MNVIC_vSetPriority \(s8_t A_s8INTID, u8_t A_u8GroupPriority, u8_t A_u8SubPriority\)](#)
Set the group and the sub-group priorities for the required interrupt.
 - [u32_t NVIC_GetPriority \(s8_t A_s8INTID\)](#)
Get a certain interrupt's priority.

7.104.1 Detailed Description

This file contains the interface information and addresses for the NVIC module.

Author

Ali El Bana

Version

2.0

Date

08/25/2022

Definition in file [NVIC_interface.h](#).

7.104.2 Function Documentation

7.104.2.1 MNVIC_vEnablePeriphral()

```
void MNVIC_vEnablePeriphral (
    u8_t A_u8INTID )
```

Enable the interrupt of a certain peripheral in NVIC.

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vDisablePeriphral](#)

Definition at line 33 of file [NVIC_program.c](#).

```
00034 {
00035     MNVIC->ISERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00036 }
```

References [MNVIC](#).

7.104.2.2 MNVIC_vDisablePeriphral()

```
void MNVIC_vDisablePeriphral (
    u8_t A_u8INTID )
```

Disable the interrupt of a certain peripheral in NVIC.

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vEnablePeriphral](#)

Definition at line 41 of file [NVIC_program.c](#).

```
00042 {
00043     MNVIC->ICERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00044 }
```

References [MNVIC](#).

7.104.2.3 MNVIC_vSetPendingFlag()

```
void MNVIC_vSetPendingFlag (
    u8_t A_u8INTID )
```

Set the pending flag of the peripheral.

Note

Used mostly for testing

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vClearPendingFlag](#)

Definition at line 49 of file [NVIC_program.c](#).

```
00050 {
00051     MNVIC->ISPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00052 }
```

References [MNVIC](#).

7.104.2.4 MNVIC_vClearPendingFlag()

```
void MNVIC_vClearPendingFlag (
    u8_t A_u8INTID )
```

Clear the pending flag of the peripheral.

Note

Used mostly for testing

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vSetPendingFlag](#)

Definition at line 57 of file [NVIC_program.c](#).

```
00058 {
00059     MNVIC->ICPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00060 }
```

References [MNVIC](#).

7.104.2.5 MNVIC_u8GetActive()

```
u8_t MNVIC_u8GetActive (
    u8_t A_u8INTID )
```

Get the current state of the interrupt active flag of a certain peripheral in NVIC.

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

Returns

The current state of the interrupt active flag

Definition at line 65 of file [NVIC_program.c](#).

```
00066 {
00067     u8_t L_u8Active = INITIAL_ZERO;
00068     L_u8Active = GET_BIT((MNVIC->IABRx[A_u8INTID / REGISTER_SIZE]), (A_u8INTID % REGISTER_SIZE));
00069     return L_u8Active;
00070 }
```

References [GET_BIT](#), [INITIAL_ZERO](#), and [MNVIC](#).

7.104.2.6 MNVIC_vSetPriorityConfig()

```
void MNVIC_vSetPriorityConfig (
    u8_t A_u8PriorityOption )
```

Configures the group and sub-group priority configuration.

Parameters

in	A_u8PriorityOption	The peripheral's interrupt ID
----	--------------------	-------------------------------

Definition at line 75 of file [NVIC_program.c](#).

```
00076 {
00077     GS_u32GroupConf = (VECTKEY_PASSWORD | (A_u8PriorityOption << 8));
00078     MSCB->AIRCR = GS_u32GroupConf;
00079 }
```

References [MSCB](#), and [VECTKEY_PASSWORD](#).

7.104.2.7 MNVIC_vSetPriority()

```
void MNVIC_vSetPriority (
    s8_t A_s8INTID,
    u8_t A_u8GroupPriority,
    u8_t A_u8SubPriority )
```

Set the group and the sub-group priorities for the required interrupt.

Parameters

in	A_s8INTID	The port that the pin belongs to
in	A_u8GroupPriority	The pin to update its mode
in	A_u8SubPriority	The alternative function to apply on the pin

See also

[Group priorities](#)

[Sub-Group priorities](#)

Definition at line 84 of file [NVIC_program.c](#).

```
00085 {
00086     u8_t L_u8Priority = INITIAL_ZERO;
00087     L_u8Priority = A_u8SubPriority | (A_u8GroupPriority << ((GS_u32GroupConf - 0x05FA0300) / 0x100));
00088
00089     // Core Peripheral
00090     if (A_s8INTID < 0)
00091     {
00092         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00093         {
00094             A_s8INTID += 3;
00095             MSCB->SHPR1 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00096         }
00097         else if (A_s8INTID == SV_CALL)
00098         {
00099             A_s8INTID += 7;
00100             MSCB->SHPR2 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00101     }
```

```

00101      }
00102      else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00103      {
00104          A_s8INTID += 8;
00105          MSCB->SHPR3 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00106      }
00107      else
00108      {
00109          /* Do nothing */
00110      }
00111  // External Peripheral
00112  else if (A_s8INTID >= 0)
00113  {
00114      MNVIC->IPRx[A_s8INTID] = (L_u8Priority << 4);
00115  }
00116  else
00117  {
00118      /* Do nothing */
00119  }
00120 }
00121 }
```

References [BUS_FAULT](#), [INITIAL_ZERO](#), [MEMORY_MANAGE](#), [MNVIC](#), [MSCB](#), [PEND_SV](#), [SV_CALL](#), [SYSTICK](#), and [USAGE_FAULT](#).

7.104.2.8 NVIC_GetPriority()

```

u32_t NVIC_GetPriority (
    s8_t A_s8INTID )
```

Get a certain interrupt's priority.

Parameters

in	A_s8INTID	
----	-----------	--

Returns

The priority for the corresponding interrupt

Definition at line 126 of file [NVIC_program.c](#).

```

00127 {
00128     u32_t L_u32ThePriorityIS = INITIAL_ZERO;
00129
00130     // Core Peripheral
00131     if (A_s8INTID < 0)
00132     {
00133         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00134         {
00135             A_s8INTID += 3;
00136             L_u32ThePriorityIS = MSCB->SHPR1 >> ((8 * A_s8INTID) + 4);
00137         }
00138         else if (A_s8INTID == SV_CALL)
00139         {
00140             A_s8INTID += 7;
00141             L_u32ThePriorityIS = MSCB->SHPR2 >> ((8 * A_s8INTID) + 4);
00142         }
00143         else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00144         {
00145             A_s8INTID += 8;
00146             L_u32ThePriorityIS = MSCB->SHPR3 >> ((8 * A_s8INTID) + 4);
00147         }
00148         else
00149         {
00150             /* Do nothing */
00151         }
00152     }
00153 // External Peripheral
```

```

00154     else if (A_s8INTID >= 0)
00155     {
00156         L_u32ThePriorityIS = (MNVIC->IPRx[A_s8INTID] >> 4);
00157     }
00158     else
00159     {
00160         /* Do nothing */
00161     }
00162
00163     return L_u32ThePriorityIS;
00164 }
```

References [BUS_FAULT](#), [INITIAL_ZERO](#), [MEMORY_MANAGE](#), [MNVIC](#), [MSCB](#), [PEND_SV](#), [SV_CALL](#), [SYSTICK](#), and [USAGE_FAULT](#).

7.105 NVIC_interface.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
00010 #ifndef _NVIC_interface_H
00011 #define _NVIC_interface_H
00012
00013 /***** Functions prototypes *****/
00014 /*                                     */
00015 /*****                                     */
00016
00022 void MNVIC_vEnablePeriphral(u8_t A_u8INTID);
00023
00029 void MNVIC_vDisablePeriphral(u8_t A_u8INTID);
00030
00037 void MNVIC_vSetPendingFlag(u8_t A_u8INTID);
00038
00045 void MNVIC_vClearPendingFlag(u8_t A_u8INTID);
00046
00052 u8_t MNVIC_u8GetActive(u8_t A_u8INTID);
00053
00058 void MNVIC_vSetPriorityConfig(u8_t A_u8PriorityOption);
00059
00068 void MNVIC_vSetPriority(s8_t A_s8INTID, u8_t A_u8GroupPriority, u8_t A_u8SubPriority);
00069
00075 u32_t NVIC_GetPriority(s8_t A_s8INTID);
00076
00077 /***** Interfacing macros *****/
00078 /*                                     */
00079 /*****                                     */
00080
00097 #define _16GROUP_NoSub_Priorities (0b011)
00105 #define _8GROUP_2Sub_Priorities (0b100)
00113 #define _4GROUP_4Sub_Priorities (0b101)
00121 #define _2GROUP_8Sub_Priorities (0b110)
00129 #define NoGROUP_16Sub_Priorities (0b111)
00147 #define GROUP_4BITS (0b011)
00154 #define GROUP_3BITS (0b100)
00161 #define GROUP_2BITS (0b101)
00168 #define GROUP_1BITS (0b110)
00175 #define GROUP_0BITS (0b111)
00189 #define NO_GROUP_PRIORITY (0)
00194 #define GROUP_PRIORITY_0 (0)
00199 #define GROUP_PRIORITY_1 (1)
00204 #define GROUP_PRIORITY_2 (2)
00209 #define GROUP_PRIORITY_3 (3)
00214 #define GROUP_PRIORITY_4 (4)
00219 #define GROUP_PRIORITY_5 (5)
00224 #define GROUP_PRIORITY_6 (6)
00229 #define GROUP_PRIORITY_7 (7)
00234 #define GROUP_PRIORITY_8 (8)
00239 #define GROUP_PRIORITY_9 (9)
00244 #define GROUP_PRIORITY_10 (10)
00249 #define GROUP_PRIORITY_11 (11)
00254 #define GROUP_PRIORITY_12 (12)
00259 #define GROUP_PRIORITY_13 (13)
00264 #define GROUP_PRIORITY_14 (14)
00269 #define GROUP_PRIORITY_15 (15)
00285 #define NO_SUB_PRIORITY (0)
00290 #define SUB_PRIORITY_0 (0)
00295 #define SUB_PRIORITY_1 (1)
00300 #define SUB_PRIORITY_2 (2)
00305 #define SUB_PRIORITY_3 (3)
00310 #define SUB_PRIORITY_4 (4)
```

```

00315 #define SUB_PRIORITY_5 (5)
00320 #define SUB_PRIORITY_6 (6)
00325 #define SUB_PRIORITY_7 (7)
00330 #define SUB_PRIORITY_8 (8)
00335 #define SUB_PRIORITY_9 (9)
00340 #define SUB_PRIORITY_10 (10)
00345 #define SUB_PRIORITY_11 (11)
00350 #define SUB_PRIORITY_12 (12)
00355 #define SUB_PRIORITY_13 (13)
00360 #define SUB_PRIORITY_14 (14)
00365 #define SUB_PRIORITY_15 (15)
00379 #define WWDG (0)
00384 #define EXTI16 (1)
00389 #define EXTI21 (2)
00394 #define EXTI22 (3)
00399 #define FLASH (4)
00404 #define RCC (5)
00409 #define EXTI0 (6)
00414 #define EXTI1 (7)
00419 #define EXTI2 (8)
00424 #define EXTI3 (9)
00429 #define EXTI4 (10)
00434 #define DMA1_STREAM0 (11)
00439 #define DMA1_STREAM1 (12)
00444 #define DMA1_STREAM2 (13)
00449 #define DMA1_STREAM3 (14)
00454 #define DMA1_STREAM4 (15)
00459 #define DMA1_STREAM5 (16)
00464 #define DMA1_STREAM6 (17)
00469 #define ADC (18)
00474 #define EXTI9 (23)
00479 #define TIM1_BRK_TIM9 (24)
00484 #define TIM1_UP_TIM10 (25)
00489 #define TIM1_TRG_COM_TIM11 (26)
00494 #define TIM1_CC (27)
00499 #define TIM2 (28)
00504 #define TIM3 (29)
00509 #define TIM4 (30)
00514 #define I2C1_EV (31)
00519 #define I2C1_ER (32)
00524 #define I2C2_EV (33)
00529 #define I2C2_ER (34)
00534 #define SPI1 (35)
00539 #define SPI2 (36)
00544 #define USART1 (37)
00549 #define USART2 (38)
00554 #define EXTI15_10 (30)
00559 #define EXTI17 (41)
00564 #define EXTI18 (42)
00569 #define DMA1_STREAM7 (47)
00574 #define SDIO (49)
00579 #define TIM5 (50)
00584 #define SPI3 (51)
00589 #define DMA2_STREAM0 (56)
00594 #define DMA2_STREAM1 (57)
00599 #define DMA2_STREAM2 (58)
00604 #define DMA2_STREAM3 (59)
00609 #define DMA2_STREAM4 (60)
00614 #define OTG_FS (67)
00619 #define DMA2_STREAM5 (68)
00624 #define DMA2_STREAM6 (69)
00629 #define DMA2_STREAM7 (70)
00634 #define USART6 (71)
00639 #define I2C3_EV (72)
00644 #define I2C3_ER (73)
00649 #define FPU (81)
00654 #define SPI4 (84)
00657 #endif // _NVIC_interface_H

```

7.106 COTS/MCAL/NVIC/NVIC_private.h File Reference

This file contains the registers information and addresses for the NVIC module.

Data Structures

- struct [NVIC_MemoryMapType](#)
NVIC configuration structure for NVIC memory map.
- struct [SCB_MemoryMapType](#)
System control block memory map structure.

Macros

- `#define NVIC_BASE_ADDRESS (0xE000E100)`
NVIC base address.
- `#define SCB_BASE_ADDRESS (0xE000ED00)`
SCB base address.
- `#define MSCB ((volatile P2VAR(SCB_MemoryMapType))(SCB_BASE_ADDRESS))`
- `#define MNVIC ((volatile P2VAR(NVIC_MemoryMapType))(NVIC_BASE_ADDRESS))`
- `#define MNVIC_STIR *((volatile 32 *) (0xE000EF00))`
- `#define VECTKEY_PASSWORD (0x05FA0000)`
VECTKEY Password.
- `#define PEND_SV (-6)`
- `#define SYSTICK (-5)`
- `#define SV_CALL (-4)`
- `#define MEMORY_MANAGE (-3)`
- `#define BUS_FAULT (-2)`
- `#define USAGE_FAULT (-1)`

7.106.1 Detailed Description

This file contains the registers information and addresses for the NVIC module.

Author

Ali El Bana

Version

2.0

Date

08/22/2022

Definition in file [NVIC_private.h](#).

7.106.2 Macro Definition Documentation

7.106.2.1 MNVIC_STIR

```
#define MNVIC_STIR *((volatile 32 *) (0xE000EF00))
```

Definition at line [243](#) of file [NVIC_private.h](#).

7.106.2.2 VECTKEY_PASSWORD

```
#define VECTKEY_PASSWORD (0x05FA0000)
```

VECTKEY Password.

This is the VECTKEY field password that must provides on every write attempt on the AIRCR register

See also

[SCB_MemoryMapType::AIRCR](#)

Definition at line 252 of file [NVIC_private.h](#).

7.107 NVIC_private.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
0010 #ifndef _NVIC_private_H
0011 #define _NVIC_private_H
0012
0018 typedef struct
0019 {
0027     u32_t ISERx[8];
0031     u32_t RESERVED0x[24];
0039     u32_t ICERx[8];
0043     u32_t RSERVED1x[24];
0053     u32_t ISPRx[8];
0057     u32_t RESERVED2x[24];
0067     u32_t ICPRx[8];
0071     u32_t RESERVED3x[24];
0077     u32_t IABRx[8];
0081     u32_t RESERVED4x[56];
0089     u8_t IPRx[240];
0090
0091 } NVIC_MemoryMapType;
0092
00100 typedef struct
00101 {
00106     u32_t CPUID;
00111     u32_t ICSR;
00116     u32_t VTOR;
00123     u32_t AIRCR;
00128     u32_t SCR;
00136     u32_t CCR;
00143     u32_t SHPR1;
00150     u32_t SHPR2;
00157     u32_t SHPR3;
00165     u32_t SHCSR;
00176     u32_t CFSR;
00183     u32_t HFSR;
00187     u32_t RESERVED;
00192     u32_t MMFAR;
00197     u32_t BFAR;
00198 } SCB_MemoryMapType;
00199
00212 #define NVIC_BASE_ADDRESS (0xE000E100)
00213
00219 #define SCB_BASE_ADDRESS (0xE000ED00)
00234 #define MSCB ((volatile P2VAR(SCB_MemoryMapType)) (SCB_BASE_ADDRESS))
00240 #define MNVIC ((volatile P2VAR(NVIC_MemoryMapType)) (NVIC_BASE_ADDRESS))
00243 #define MNVIC_STIR *((volatile 32 *) (0xE000EF00))
00244
00252 #define VECTKEY_PASSWORD (0x05FA0000)
00253
00266 #define PEND_SV (-6)
00272 #define SYSTICK (-5)
00278 #define SV_CALL (-4)
00284 #define MEMORY_MANAGE (-3)
00290 #define BUSFAULT (-2)
00296 #define USAGE_FAULT (-1)
00299 #endif // _NVIC_private_H
```

7.108 COTS/MCAL/NVIC/NVIC_program.c File Reference

This file contains the source code of the interfacing for the NVIC modules.

```
#include "../../../LIB/LSTD_TYPES.h"
#include "../../../LIB/LSTD_COMPILER.h"
#include "../../../LIB/LSTD_VALUES.h"
#include "../../../LIB/LSTD_BITMATH.h"
#include "NVIC_interface.h"
#include "NVIC_private.h"
#include "NVIC_config.h"
```

Macros

- #define REGISTER_SIZE (32)

Functions

- void MNVIC_vEnablePeriphral (u8_t A_u8INTID)
Enable the interrupt of a certain peripheral in NVIC.
- void MNVIC_vDisablePeriphral (u8_t A_u8INTID)
Disable the interrupt of a certain peripheral in NVIC.
- void MNVIC_vSetPendingFlag (u8_t A_u8INTID)
Set the pending flag of the peripheral.
- void MNVIC_vClearPendingFlag (u8_t A_u8INTID)
Clear the pending flag of the peripheral.
- u8_t MNVIC_u8GetActive (u8_t A_u8INTID)
Get the current state of the interrupt active flag of a certain peripheral in NVIC.
- void MNVIC_vSetPriorityConfig (u8_t A_u8PriorityOption)
Configures the group and sub-group priority configuration.
- void MNVIC_vSetPriority (s8_t A_s8INTID, u8_t A_u8GroupPriority, u8_t A_u8SubPriority)
Set the group and the sub-group priorities for the required interrupt.
- u32_t NVIC_GetPriority (s8_t A_s8INTID)
Get a certain interrupt's priority.

7.108.1 Detailed Description

This file contains the source code of the interfacing for the NVIC modules.

Author

Ali El Bana

Version

2.0

Date

08/25/2022

Definition in file [NVIC_program.c](#).

7.108.2 Macro Definition Documentation

7.108.2.1 REGISTER_SIZE

```
#define REGISTER_SIZE (32)
```

Definition at line 27 of file [NVIC_program.c](#).

7.108.3 Function Documentation

7.108.3.1 MNVIC_vEnablePeriphral()

```
void MNVIC_vEnablePeriphral (
    u8_t A_u8INTID )
```

Enable the interrupt of a certain peripheral in NVIC.

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vDisablePeriphral](#)

Definition at line 33 of file [NVIC_program.c](#).

```
00034 {
00035     MNVIC->ISERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00036 }
```

References [MNVIC](#).

7.108.3.2 MNVIC_vDisablePeriphral()

```
void MNVIC_vDisablePeriphral (
    u8_t A_u8INTID )
```

Disable the interrupt of a certain peripheral in NVIC.

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vEnablePeriphral](#)

Definition at line 41 of file [NVIC_program.c](#).

```
00042 {  
00043     MNVIC->ICERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));  
00044 }
```

References [MNVIC](#).

7.108.3.3 MNVIC_vSetPendingFlag()

```
void MNVIC_vSetPendingFlag (  
    u8_t A_u8INTID )
```

Set the pending flag of the peripheral.

Note

Used mostly for testing

Parameters

in	A_u8INTID	The peripheral's interrupt ID
----	-----------	-------------------------------

See also

[MNVIC_vClearPendingFlag](#)

Definition at line 49 of file [NVIC_program.c](#).

```
00050 {  
00051     MNVIC->ISPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));  
00052 }
```

References [MNVIC](#).

7.108.3.4 MNVIC_vClearPendingFlag()

```
void MNVIC_vClearPendingFlag (  
    u8_t A_u8INTID )
```

Clear the pending flag of the peripheral.

Note

Used mostly for testing

Parameters

in	<i>A_u8INTID</i>	The peripheral's interrupt ID
----	------------------	-------------------------------

See also

[MNVIC_vSetPendingFlag](#)

Definition at line 57 of file [NVIC_program.c](#).

```
00058 {
00059     MNVIC->ICPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00060 }
```

References [MNVIC](#).

7.108.3.5 MNVIC_u8GetActive()

```
u8_t MNVIC_u8GetActive (
    u8_t A_u8INTID )
```

Get the current state of the interrupt active flag of a certain peripheral in NVIC.

Parameters

in	<i>A_u8INTID</i>	The peripheral's interrupt ID
----	------------------	-------------------------------

Returns

The current state of the interrupt active flag

Definition at line 65 of file [NVIC_program.c](#).

```
00066 {
00067     u8_t L_u8Active = INITIAL_ZERO;
00068     L_u8Active = GET_BIT((MNVIC->IABRx[A_u8INTID / REGISTER_SIZE]), (A_u8INTID % REGISTER_SIZE));
00069     return L_u8Active;
00070 }
```

References [GET_BIT](#), [INITIAL_ZERO](#), and [MNVIC](#).

7.108.3.6 MNVIC_vSetPriorityConfig()

```
void MNVIC_vSetPriorityConfig (
    u8_t A_u8PriorityOption )
```

Configures the group and sub-group priority configuration.

Parameters

in	<i>A_u8PriorityOption</i>	The peripheral's interrupt ID
----	---------------------------	-------------------------------

Definition at line 75 of file [NVIC_program.c](#).

```
00076 {
00077     GS_u32GroupConf = (VECTKEY_PASSWORD | (A_u8PriorityOption << 8));
00078     MSCB->AIRCR = GS_u32GroupConf;
00079 }
```

References [MSCB](#), and [VECTKEY_PASSWORD](#).

7.108.3.7 MNVIC_vSetPriority()

```
void MNVIC_vSetPriority (
    s8_t A_s8INTID,
    u8_t A_u8GroupPriority,
    u8_t A_u8SubPriority )
```

Set the group and the sub-group priorities for the required interrupt.

Parameters

in	<i>A_s8INTID</i>	The port that the pin belongs to
in	<i>A_u8GroupPriority</i>	The pin to update its mode
in	<i>A_u8SubPriority</i>	The alternative function to apply on the pin

See also

[Group priorities](#)

[Sub-Group priorities](#)

Definition at line 84 of file [NVIC_program.c](#).

```
00085 {
00086     u8_t L_u8Priority = INITIAL_ZERO;
00087     L_u8Priority = A_u8SubPriority | (A_u8GroupPriority << ((GS_u32GroupConf - 0x05FA0300) / 0x100));
00088
00089     // Core Peripheral
00090     if (A_s8INTID < 0)
00091     {
00092         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00093         {
00094             A_s8INTID += 3;
00095             MSCB->SHPR1 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00096         }
00097         else if (A_s8INTID == SV_CALL)
00098         {
00099             A_s8INTID += 7;
00100             MSCB->SHPR2 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00101         }
00102         else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00103         {
00104             A_s8INTID += 8;
00105             MSCB->SHPR3 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00106         }
00107         else
00108         {
00109             /* Do nothing */
00110         }
00111     }
00112     // External Peripheral
00113     else if (A_s8INTID >= 0)
00114     {
00115         MNVIC->IPRx[A_s8INTID] = (L_u8Priority << 4);
00116     }
00117     else
00118     {
00119         /* Do nothing */
00120     }
```

```
00121 }
```

References [BUS_FAULT](#), [INITIAL_ZERO](#), [MEMORY_MANAGE](#), [MNVIC](#), [MSCB](#), [PEND_SV](#), [SV_CALL](#), [SYSTICK](#), and [USAGE_FAULT](#).

7.108.3.8 NVIC_GetPriority()

```
u32_t NVIC_GetPriority (
    s8_t A_s8INTID )
```

Get a certain interrupt's priority.

Parameters

in	A_s8INTID	
----	-----------	--

Returns

The priority for the corresponding interrupt

Definition at line 126 of file [NVIC_program.c](#).

```
00127 {
00128     u32_t L_u32ThePriorityIS = INITIAL_ZERO;
00129
00130     // Core Peripheral
00131     if (A_s8INTID < 0)
00132     {
00133         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00134         {
00135             A_s8INTID += 3;
00136             L_u32ThePriorityIS = MSCB->SHPR1 >> ((8 * A_s8INTID) + 4);
00137         }
00138         else if (A_s8INTID == SV_CALL)
00139         {
00140             A_s8INTID += 7;
00141             L_u32ThePriorityIS = MSCB->SHPR2 >> ((8 * A_s8INTID) + 4);
00142         }
00143         else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00144         {
00145             A_s8INTID += 8;
00146             L_u32ThePriorityIS = MSCB->SHPR3 >> ((8 * A_s8INTID) + 4);
00147         }
00148         else
00149         {
00150             /* Do nothing */
00151         }
00152     }
00153     // External Peripheral
00154     else if (A_s8INTID >= 0)
00155     {
00156         L_u32ThePriorityIS = (MNVIC->IPRx[A_s8INTID] >> 4);
00157     }
00158     else
00159     {
00160         /* Do nothing */
00161     }
00162
00163     return L_u32ThePriorityIS;
00164 }
```

References [BUS_FAULT](#), [INITIAL_ZERO](#), [MEMORY_MANAGE](#), [MNVIC](#), [MSCB](#), [PEND_SV](#), [SV_CALL](#), [SYSTICK](#), and [USAGE_FAULT](#).

7.109 NVIC_program.c

[Go to the documentation of this file.](#)

```

00001
00009 /***** Include headers *****/
0010 /*           Include headers           */
0011 /***** Include headers *****/
0012
0013 #include "../../LIB/LSTD_TYPES.h"
0014 #include "../../LIB/LSTD_COMPILER.h"
0015 #include "../../LIB/LSTD_VALUES.h"
0016 #include "../../LIB/LSTD_BITMATH.h"
0017 #include "NVIC_interface.h"
0018 #include "NVIC_private.h"
0019 #include "NVIC_config.h"
0020
0021 /***** Global functions *****/
0022 /*           Global functions           */
0023 /***** Global functions *****/
0024
0025 STATIC VAR(u32_t) GS_u32GroupConf = INITIAL_ZERO;
0026
0027 #define REGISTER_SIZE (32)
0028
0029 /***** Functions' implementations *****/
0030 /*           Functions' implementations           */
0031 /***** Functions' implementations *****/
0032
0033 void MNVIC_vEnablePeriphral(u8_t A_u8INTID)
0034 {
0035     MNVIC->ISERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
0036 }
0037
0038
0039 /*****
0040
0041 void MNVIC_vDisablePeriphral(u8_t A_u8INTID)
0042 {
0043     MNVIC->ICERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
0044 }
0045
0046
0047 /*****
0048
0049 void MNVIC_vSetPendingFlag(u8_t A_u8INTID)
0050 {
0051     MNVIC->ISPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
0052 }
0053
0054
0055 /*****
0056
0057 void MNVIC_vClearPendingFlag(u8_t A_u8INTID)
0058 {
0059     MNVIC->ICPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
0060 }
0061
0062
0063 /*****
0064
0065 u8_t MNVIC_u8GetActive(u8_t A_u8INTID)
0066 {
0067     u8_t L_u8Active = INITIAL_ZERO;
0068     L_u8Active = GET_BIT((MNVIC->IABRx[A_u8INTID / REGISTER_SIZE]), (A_u8INTID % REGISTER_SIZE));
0069     return L_u8Active;
0070 }
0071
0072
0073 /*****
0074
0075 void MNVIC_vSetPriorityConfig(u8_t A_u8PriorityOption)
0076 {
0077     GS_u32GroupConf = (VECTKEY_PASSWORD | (A_u8PriorityOption << 8));
0078     MSCB->AIRCR = GS_u32GroupConf;
0079 }

```

```

00080
00081     /*****
00082     *****/
00083
00084 void MNVIC_vSetPriority(s8_t A_s8INTID, u8_t A_u8GroupPriority, u8_t A_u8SubPriority)
00085 {
00086     u8_t L_u8Priority = INITIAL_ZERO;
00087     L_u8Priority = A_u8SubPriority | (A_u8GroupPriority << ((GS_u32GroupConf - 0x05FA0300) / 0x100));
00088
00089     // Core Peripheral
00090     if (A_s8INTID < 0)
00091     {
00092         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00093         {
00094             A_s8INTID += 3;
00095             MSCB->SHPR1 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00096         }
00097         else if (A_s8INTID == SV_CALL)
00098         {
00099             A_s8INTID += 7;
00100            MSCB->SHPR2 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00101        }
00102        else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00103        {
00104            A_s8INTID += 8;
00105            MSCB->SHPR3 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00106        }
00107        else
00108        {
00109            /* Do nothing */
00110        }
00111    }
00112    // External Peripheral
00113    else if (A_s8INTID >= 0)
00114    {
00115        MNVIC->IPRx[A_s8INTID] = (L_u8Priority << 4);
00116    }
00117    else
00118    {
00119        /* Do nothing */
00120    }
00121 }
00122
00123     *****/
00124     *****/
00125
00126 u32_t NVIC_GetPriority(s8_t A_s8INTID)
00127 {
00128     u32_t L_u32ThePriorityIS = INITIAL_ZERO;
00129
00130     // Core Peripheral
00131     if (A_s8INTID < 0)
00132     {
00133         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00134         {
00135             A_s8INTID += 3;
00136             L_u32ThePriorityIS = MSCB->SHPR1 >> ((8 * A_s8INTID) + 4);
00137         }
00138         else if (A_s8INTID == SV_CALL)
00139         {
00140             A_s8INTID += 7;
00141             L_u32ThePriorityIS = MSCB->SHPR2 >> ((8 * A_s8INTID) + 4);
00142         }
00143         else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00144         {
00145             A_s8INTID += 8;
00146             L_u32ThePriorityIS = MSCB->SHPR3 >> ((8 * A_s8INTID) + 4);
00147         }
00148         else
00149         {
00150             /* Do nothing */
00151         }
00152     }
00153     // External Peripheral
00154     else if (A_s8INTID >= 0)
00155     {
00156         L_u32ThePriorityIS = (MNVIC->IPRx[A_s8INTID] >> 4);
00157     }
00158     else
00159     {
00160         /* Do nothing */
00161     }
00162 }
```

```
00163     return L_u32ThePriorityIS;
00164 }
00165
00166
00167 //*****
```

7.110 COTS/MCAL/RCC/MRCC_config.h File Reference

This file contains the RCC configurations.

Macros

- #define PLLI2S DISABLE
- #define PLL DISABLE
- #define CSS DISABLE
- #define HSEBYP NOTBYBASED
- #define HSE_EN DISABLE
- #define HSI_EN ENABLE
- #define PLLQ Equal_2
- #define PLLSRC HSE
- #define PLLP PLLCLK
- #define PLLN Equal_200
- #define PLLM Equal_2
- #define MCO2 SYSCLK
- #define MCO2PRE NoDivision
- #define MCO1PRE NoDivision
- #define I2SSRC PLLI2SCLK
- #define MCO1 HSE
- #define RTCPRE HSEby2
- #define PPRE2 NoDivision
- #define PPRE1 NoDivision
- #define HPRE SYSCLKby2
- #define SWS HSI
- #define SW HSI
- #define DMA2CLK DISABLE
- #define DMA1CLK DISABLE
- #define CRCCLK DISABLE
- #define GPIOHCLK DISABLE
- #define GPIOECLK DISABLE
- #define GPIODCLK DISABLE
- #define GPIOCCLK DISABLE
- #define GPIOBCLK DISABLE
- #define GPIOACLK DISABLE
- #define OTGFS DISABLE
- #define PWREN DISABLE
- #define I2C3EN DISABLE
- #define I2C2EN DISABLE
- #define I2C1EN DISABLE
- #define USART2EN DISABLE
- #define SPI3EN DISABLE
- #define SPI2EN DISABLE
- #define WWDGEN DISABLE

- #define [TIM5EN](#) DISABLE
- #define [TIM4EN](#) DISABLE
- #define [TIM3EN](#) DISABLE
- #define [TIM2EN](#) DISABLE
- #define [TIM11EN](#) DISABLE
- #define [TIM10EN](#) DISABLE
- #define [TIM9EN](#) DISABLE
- #define [SYSCFGEN](#) DISABLE
- #define [SPI4EN](#) DISABLE
- #define [SPI1EN](#) DISABLE
- #define [SDIOEN](#) DISABLE
- #define [ADC1EN](#) DISABLE
- #define [USART6EN](#) DISABLE
- #define [USART1EN](#) DISABLE
- #define [TIM1EN](#) DISABLE

7.110.1 Detailed Description

This file contains the RCC configurations.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

11/9/2022

Definition in file [MRCC_config.h](#).

7.110.2 Macro Definition Documentation

7.110.2.1 PLLI2S

#define [PLLI2S](#) DISABLE

Definition at line [21](#) of file [MRCC_config.h](#).

7.110.2.2 PLL

```
#define PLL DISABLE
```

Definition at line 29 of file [MRCC_config.h](#).

7.110.2.3 CSS

```
#define CSS DISABLE
```

Definition at line 37 of file [MRCC_config.h](#).

7.110.2.4 HSEBYP

```
#define HSEBYP NOTBYBASED
```

Definition at line 45 of file [MRCC_config.h](#).

7.110.2.5 HSE_EN

```
#define HSE_EN DISABLE
```

Definition at line 53 of file [MRCC_config.h](#).

7.110.2.6 HSI_EN

```
#define HSI_EN ENABLE
```

Definition at line 61 of file [MRCC_config.h](#).

7.110.2.7 PLLQ

```
#define PLLQ Equal_2
```

Definition at line 84 of file [MRCC_config.h](#).

7.110.2.8 PLLSRC

```
#define PLLSRC HSE
```

Definition at line [92](#) of file [MRCC_config.h](#).

7.110.2.9 PLLP

```
#define PLLP PLLCLK
```

Definition at line [102](#) of file [MRCC_config.h](#).

7.110.2.10 PLLN

```
#define PLLN Equal_200
```

Definition at line [173](#) of file [MRCC_config.h](#).

7.110.2.11 PLLM

```
#define PLLM Equal_2
```

Definition at line [240](#) of file [MRCC_config.h](#).

7.110.2.12 MCO2

```
#define MCO2 SYSCLK
```

Definition at line [253](#) of file [MRCC_config.h](#).

7.110.2.13 MCO2PRE

```
#define MCO2PRE NoDivision
```

Definition at line [264](#) of file [MRCC_config.h](#).

7.110.2.14 MCO1PRE

```
#define MCO1PRE NoDivision
```

Definition at line 275 of file [MRCC_config.h](#).

7.110.2.15 I2SSRC

```
#define I2SSRC PLLI2SCLK
```

Definition at line 283 of file [MRCC_config.h](#).

7.110.2.16 MCO1

```
#define MCO1 HSE
```

Definition at line 293 of file [MRCC_config.h](#).

7.110.2.17 RTCPRE

```
#define RTCPRE HSEby2
```

Definition at line 332 of file [MRCC_config.h](#).

7.110.2.18 PPREG

```
#define PPREG NoDivision
```

Definition at line 343 of file [MRCC_config.h](#).

7.110.2.19 PPREF

```
#define PPREF NoDivision
```

Definition at line 354 of file [MRCC_config.h](#).

7.110.2.20 HPRE

```
#define HPRE SYSCLKby2
```

Definition at line 369 of file [MRCC_config.h](#).

7.110.2.21 SWS

```
#define SWS HSI
```

Definition at line 378 of file [MRCC_config.h](#).

7.110.2.22 SW

```
#define SW HSI
```

Definition at line 387 of file [MRCC_config.h](#).

7.110.2.23 DMA2CLK

```
#define DMA2CLK DISABLE
```

Definition at line 398 of file [MRCC_config.h](#).

7.110.2.24 DMA1CLK

```
#define DMA1CLK DISABLE
```

Definition at line 406 of file [MRCC_config.h](#).

7.110.2.25 CRCCLK

```
#define CRCCLK DISABLE
```

Definition at line 414 of file [MRCC_config.h](#).

7.110.2.26 GPIOHCLK

```
#define GPIOHCLK DISABLE
```

Definition at line 422 of file [MRCC_config.h](#).

7.110.2.27 GPIOECLK

```
#define GPIOECLK DISABLE
```

Definition at line 430 of file [MRCC_config.h](#).

7.110.2.28 GPIODCLK

```
#define GPIODCLK DISABLE
```

Definition at line 438 of file [MRCC_config.h](#).

7.110.2.29 GPIOCCLK

```
#define GPIOCCLK DISABLE
```

Definition at line 446 of file [MRCC_config.h](#).

7.110.2.30 GPIOBCLK

```
#define GPIOBCLK DISABLE
```

Definition at line 454 of file [MRCC_config.h](#).

7.110.2.31 GPIOACLK

```
#define GPIOACLK DISABLE
```

Definition at line 462 of file [MRCC_config.h](#).

7.110.2.32 OTGFS

```
#define OTGFS DISABLE
```

Definition at line [473](#) of file [MRCC_config.h](#).

7.110.2.33 PWREN

```
#define PWREN DISABLE
```

Definition at line [484](#) of file [MRCC_config.h](#).

7.110.2.34 I2C3EN

```
#define I2C3EN DISABLE
```

Definition at line [492](#) of file [MRCC_config.h](#).

7.110.2.35 I2C2EN

```
#define I2C2EN DISABLE
```

Definition at line [500](#) of file [MRCC_config.h](#).

7.110.2.36 I2C1EN

```
#define I2C1EN DISABLE
```

Definition at line [508](#) of file [MRCC_config.h](#).

7.110.2.37 USART2EN

```
#define USART2EN DISABLE
```

Definition at line [516](#) of file [MRCC_config.h](#).

7.110.2.38 SPI3EN

```
#define SPI3EN DISABLE
```

Definition at line 524 of file [MRCC_config.h](#).

7.110.2.39 SPI2EN

```
#define SPI2EN DISABLE
```

Definition at line 532 of file [MRCC_config.h](#).

7.110.2.40 WWDGEN

```
#define WWDGEN DISABLE
```

Definition at line 540 of file [MRCC_config.h](#).

7.110.2.41 TIM5EN

```
#define TIM5EN DISABLE
```

Definition at line 548 of file [MRCC_config.h](#).

7.110.2.42 TIM4EN

```
#define TIM4EN DISABLE
```

Definition at line 556 of file [MRCC_config.h](#).

7.110.2.43 TIM3EN

```
#define TIM3EN DISABLE
```

Definition at line 564 of file [MRCC_config.h](#).

7.110.2.44 TIM2EN

```
#define TIM2EN DISABLE
```

Definition at line [572](#) of file [MRCC_config.h](#).

7.110.2.45 TIM11EN

```
#define TIM11EN DISABLE
```

Definition at line [583](#) of file [MRCC_config.h](#).

7.110.2.46 TIM10EN

```
#define TIM10EN DISABLE
```

Definition at line [591](#) of file [MRCC_config.h](#).

7.110.2.47 TIM9EN

```
#define TIM9EN DISABLE
```

Definition at line [599](#) of file [MRCC_config.h](#).

7.110.2.48 SYSCFGEN

```
#define SYSCFGEN DISABLE
```

Definition at line [607](#) of file [MRCC_config.h](#).

7.110.2.49 SPI4EN

```
#define SPI4EN DISABLE
```

Definition at line [615](#) of file [MRCC_config.h](#).

7.110.2.50 SPI1EN

```
#define SPI1EN DISABLE
```

Definition at line [623](#) of file [MRCC_config.h](#).

7.110.2.51 SDIOEN

```
#define SDIOEN DISABLE
```

Definition at line [631](#) of file [MRCC_config.h](#).

7.110.2.52 ADC1EN

```
#define ADC1EN DISABLE
```

Definition at line [639](#) of file [MRCC_config.h](#).

7.110.2.53 USART6EN

```
#define USART6EN DISABLE
```

Definition at line [647](#) of file [MRCC_config.h](#).

7.110.2.54 USART1EN

```
#define USART1EN DISABLE
```

Definition at line [655](#) of file [MRCC_config.h](#).

7.110.2.55 TIM1EN

```
#define TIM1EN DISABLE
```

Definition at line [663](#) of file [MRCC_config.h](#).

7.111 MRCC_config.h

[Go to the documentation of this file.](#)

```

00001
00009 #ifndef MCAL_RCC_MRCC_CONFIG_H_
0010 #define MCAL_RCC_MRCC_CONFIG_H_
0011
0012
0013 /***** RCC_CR configurations *****/
0014 /*          RCC_CR configurations          */
0015 /***** RCC_CR configurations *****/
0016
0017 /*options:
0018 *ENABLE
0019 *DISABLE
0020 */
0021 #define PLLI2S DISABLE
0022
0023 /***** CSS configurations *****/
0024
0025 /*options:
0026 *ENABLE
0027 *DISABLE
0028 */
0029 #define PLL DISABLE
0030
0031 /***** HSEBYP configurations *****/
0032
0033 /*options:
0034 *ENABLE
0035 *DISABLE
0036 */
0037 #define CSS DISABLE
0038
0039 /***** HSE_EN configurations *****/
0040
0041 /*options:
0042 *BYBASED
0043 *NOTBYBASED
0044 */
0045 #define HSEBYP NOTBYBASED
0046
0047 /***** HSE_EN configurations *****/
0048
0049 /*options:
0050 *ENABLE
0051 *DISABLE
0052 */
0053 #define HSE_EN DISABLE
0054
0055 /***** HSI_EN configurations *****/
0056
0057 /*options:
0058 *ENABLE
0059 *DISABLE
0060 */
0061 #define HSI_EN ENABLE
0062
0063
0064 // RCC_PLLCFGR configurations //
0065
0066 /***** PLLQ configurations *****/
0067
0068 /*options:
0069 *Equal_2
0070 *Equal_3
0071 *Equal_4
0072 *Equal_5
0073 *Equal_6
0074 *Equal_7
0075 *Equal_8
0076 *Equal_9
0077 *Equal_10
0078 *Equal_11
0079 *Equal_12
0080 *Equal_13
0081 *Equal_14
0082 *Equal_15
0083 */
0084 #define PLLQ Equal_2
0085
0086 /***** HSI configurations *****/
0087
0088 /*options:
0089 *HSI

```

```
00090 *HSE
00091 */
00092 #define PLLSRC HSE
00093
00094 /***** */
00095
00096 /*options:
00097 *Equal_2
00098 *Equal_4
00099 *Equal_6
00100 *Equal_8
00101 */
00102 #define PLLP PLLCLK
00103
00104 /***** */
00105
00106 /*options:
00107 *Equal_192
00108 *Equal_193
00109 *Equal_194
00110 *Equal_195
00111 *Equal_196
00112 *Equal_197
00113 *Equal_198
00114 *Equal_199
00115 *Equal_200
00116 *Equal_201
00117 *Equal_202
00118 *Equal_203
00119 *Equal_204
00120 *Equal_205
00121 *Equal_206
00122 *Equal_207
00123 *Equal_208
00124 *Equal_209
00125 *Equal_210
00126
00127 *Equal_300
00128 *Equal_301
00129 *Equal_302
00130 *Equal_303
00131 *Equal_304
00132 *Equal_305
00133 *Equal_306
00134 *Equal_307
00135 *Equal_308
00136 *Equal_309
00137 *Equal_310
00138
00139 *Equal_400
00140 *Equal_401
00141 *Equal_402
00142 *Equal_403
00143 *Equal_404
00144 *Equal_405
00145 *Equal_406
00146 *Equal_407
00147 *Equal_408
00148 *Equal_409
00149 *Equal_410
00150 *Equal_411
00151 *Equal_412
00152 *Equal_413
00153 *Equal_414
00154 *Equal_415
00155 *Equal_416
00156 *Equal_417
00157 *Equal_418
00158 *Equal_419
00159 *Equal_420
00160 *Equal_421
00161 *Equal_422
00162 *Equal_423
00163 *Equal_424
00164 *Equal_425
00165 *Equal_426
00166 *Equal_427
00167 *Equal_428
00168 *Equal_429
00169 *Equal_430
00170 *Equal_431
00171 *Equal_432
00172 */
00173 #define PLLN Equal_200
00174
00175 /***** */
00176
```

```
00177 /*options:  
00178 *Equal_2  
00179 *Equal_4  
00180 *Equal_5  
00181 *Equal_6  
00182 *Equal_7  
00183 *Equal_8  
00184 *Equal_9  
00185 *Equal_10  
00186 *Equal_11  
00187 *Equal_12  
00188 *Equal_13  
00189 *Equal_14  
00190 *Equal_15  
00191 *Equal_16  
00192 *Equal_17  
00193 *Equal_18  
00194 *Equal_19  
00195 *Equal_20  
00196 *Equal_21  
00197 *Equal_22  
00198 *Equal_23  
00199 *Equal_24  
00200 *Equal_25  
00201 *Equal_26  
00202 *Equal_27  
00203 *Equal_28  
00204 *Equal_29  
00205 *Equal_30  
00206 *Equal_31  
00207 *Equal_32  
00208 *Equal_33  
00209 *Equal_34  
00210 *Equal_35  
00211 *Equal_36  
00212 *Equal_37  
00213 *Equal_38  
00214 *Equal_39  
00215 *Equal_40  
00216 *Equal_41  
00217 *Equal_42  
00218 *Equal_43  
00219 *Equal_44  
00220 *Equal_45  
00221 *Equal_46  
00222 *Equal_47  
00223 *Equal_48  
00224 *Equal_49  
00225 *Equal_50  
00226 *Equal_51  
00227 *Equal_52  
00228 *Equal_53  
00229 *Equal_54  
00230 *Equal_55  
00231 *Equal_56  
00232 *Equal_57  
00233 *Equal_58  
00234 *Equal_59  
00235 *Equal_60  
00236 *Equal_61  
00237 *Equal_62  
00238 *Equal_63  
00239 */  
00240 #define PLLM Equal_2  
00241  
00242  
00243 /*****  
00244 // RCC_CFGR configurations //  
00245 *****/  
00246  
00247 /*options:  
00248 *SYSCLK  
00249 *PLL12SCLK  
00250 *HSE  
00251 *PLLCLK  
00252 */  
00253 #define MCO2 SYSCLK  
00254  
00255 /*****  
00256  
00257 /*options:  
00258 *NoDivision  
00259 *DivisionBy2  
00260 *DivisionBy3  
00261 *DivisionBy4  
00262 *DivisionBy5  
00263 */
```

```
00264 #define MCO2PRE NoDivision
00265
00266 /******
00267 *****/
00268 /*options:
00269 *NoDivision
00270 *DivisionBy2
00271 *DivisionBy3
00272 *DivisionBy4
00273 *DivisionBy5
00274 */
00275 #define MCO1PRE NoDivision
00276
00277 /******
00278 *****/
00279 /*options:
00280 *PLL12SCLK
00281 *I2S_CKIN
00282 */
00283 #define I2SSRC PLL12SCLK
00284
00285 /******
00286 *****/
00287 /*options:
00288 *HSI
00289 *LSE
00290 *HSE
00291 *PLLCLK
00292 */
00293 #define MCO1 HSE
00294
00295 /******
00296 *****/
00297 /*options:
00298 *NoCLK0
00299 *NoCLK1
00300 *HSEby2
00301 *HSEby3
00302 *HSEby4
00303 *HSEby5
00304 *HSEby6
00305 *HSEby7
00306 *HSEby8
00307 *HSEby9
00308 *HSEby10
00309 *HSEby11
00310 *HSEby12
00311 *HSEby13
00312 *HSEby14
00313 *HSEby15
00314 *HSEby16
00315 *HSEby17
00316 *HSEby18
00317 *HSEby19
00318 *HSEby20
00319 *HSEby21
00320 *HSEby22
00321 *HSEby22
00322 *HSEby23
00323 *HSEby24
00324 *HSEby25
00325 *HSEby26
00326 *HSEby27
00327 *HSEby28
00328 *HSEby29
00329 *HSEby30
00330 *HSEby31
00331 */
00332 #define RTCPRE HSEby2
00333
00334 /******
00335 *****/
00336 /*options:
00337 *NoDivision
00338 *AHBby2
00339 *AHBby4
00340 *AHBby8
00341 *AHBby16
00342 */
00343 #define PPRE2 NoDivision
00344
00345 /******
00346 *****/
00347 /*options:
00348 *NoDivision
00349 *AHBby2
00350 *AHBby4
```

```
00351 *AHBby8
00352 *AHBby16
00353 */
00354 #define PPRE1 NoDivision
00355
00356 /***** */
00357
00358 /*options:
00359 *NoDivision
00360 *SYSCLKby2
00361 *SYSCLKby4
00362 *SYSCLKby8
00363 *SYSCLKby16
00364 *SYSCLKby64
00365 *SYSCLKby128
00366 *SYSCLKby256
00367 *SYSCLKby512
00368 */
00369 #define HPRE SYSCLKby2
00370
00371 /***** */
00372
00373 /*options:
00374 *HSI
00375 *HSE
00376 *PLLCLK
00377 */
00378 #define SWS HSI
00379
00380 /***** */
00381
00382 /*options:
00383 *HSI
00384 *HSE
00385 *PLLCLK
00386 */
00387 #define SW HSI
00388
00389
00390 /***** */
00391 // RCC_AHB1ENR configurations //
00392 /***** */
00393
00394 /*options:
00395 *ENABLE
00396 *DISABLE
00397 */
00398 #define DMA2CLK DISABLE
00399
00400 /***** */
00401
00402 /*options:
00403 *ENABLE
00404 *DISABLE
00405 */
00406 #define DMA1CLK DISABLE
00407
00408 /***** */
00409
00410 /*options:
00411 *ENABLE
00412 *DISABLE
00413 */
00414 #define CRCCLK DISABLE
00415
00416 /***** */
00417
00418 /*options:
00419 *ENABLE
00420 *DISABLE
00421 */
00422 #define GPIOHCLK DISABLE
00423
00424 /***** */
00425
00426 /*options:
00427 *ENABLE
00428 *DISABLE
00429 */
00430 #define GPIOECLK DISABLE
00431
00432 /***** */
00433
00434 /*options:
00435 *ENABLE
00436 *DISABLE
00437 */
```

```
00438 #define GPIODCLK DISABLE
00439
00440 /*****
00441
00442 /*options:
00443 *ENABLE
00444 *DISABLE
00445 */
00446 #define GPIOCCLK DISABLE
00447
00448 /*****
00449
00450 /*options:
00451 *ENABLE
00452 *DISABLE
00453 */
00454 #define GPIOBCLK DISABLE
00455
00456 /*****
00457
00458 /*options:
00459 *ENABLE
00460 *DISABLE
00461 */
00462 #define GPIOACLK DISABLE
00463
00464
00465 /*****
00466           // RCC_AHB2ENR configurations //
00467 /*****
00468
00469 /*options:
00470 *ENABLE
00471 *DISABLE
00472 */
00473 #define OTGFS DISABLE
00474
00475
00476 /*****
00477           // RCC_APB1ENR configurations //
00478 /*****
00479
00480 /*options:
00481 *ENABLE
00482 *DISABLE
00483 */
00484 #define PWREN DISABLE
00485
00486 /*****
00487
00488 /*options:
00489 *ENABLE
00490 *DISABLE
00491 */
00492 #define I2C3EN DISABLE
00493
00494 /*****
00495
00496 /*options:
00497 *ENABLE
00498 *DISABLE
00499 */
00500 #define I2C2EN DISABLE
00501
00502 /*****
00503
00504 /*options:
00505 *ENABLE
00506 *DISABLE
00507 */
00508 #define I2C1EN DISABLE
00509
00510 /*****
00511
00512 /*options:
00513 *ENABLE
00514 *DISABLE
00515 */
00516 #define USART2EN DISABLE
00517
00518 /*****
00519
00520 /*options:
00521 *ENABLE
00522 *DISABLE
00523 */
00524 #define SPI3EN DISABLE
```

```
00525
00526 /****** */
00527
00528 /*options:
00529 *ENABLE
00530 *DISABLE
00531 */
00532 #define SPI2EN DISABLE
00533
00534 /****** */
00535
00536 /*options:
00537 *ENABLE
00538 *DISABLE
00539 */
00540 #define WWDGEN DISABLE
00541
00542 /****** */
00543
00544 /*options:
00545 *ENABLE
00546 *DISABLE
00547 */
00548 #define TIM5EN DISABLE
00549
00550 /****** */
00551
00552 /*options:
00553 *ENABLE
00554 *DISABLE
00555 */
00556 #define TIM4EN DISABLE
00557
00558 /****** */
00559
00560 /*options:
00561 *ENABLE
00562 *DISABLE
00563 */
00564 #define TIM3EN DISABLE
00565
00566 /****** */
00567
00568 /*options:
00569 *ENABLE
00570 *DISABLE
00571 */
00572 #define TIM2EN DISABLE
00573
00574
00575 /****** */
00576 // RCC_APB2ENR configurations //
00577 /****** */
00578
00579 /*options:
00580 *ENABLE
00581 *DISABLE
00582 */
00583 #define TIM11EN DISABLE
00584
00585 /****** */
00586
00587 /*options:
00588 *ENABLE
00589 *DISABLE
00590 */
00591 #define TIM10EN DISABLE
00592
00593 /****** */
00594
00595 /*options:
00596 *ENABLE
00597 *DISABLE
00598 */
00599 #define TIM9EN DISABLE
00600
00601 /****** */
00602
00603 /*options:
00604 *ENABLE
00605 *DISABLE
00606 */
00607 #define SYSCFGEN DISABLE
00608
00609 /****** */
00610
00611 /*options:
```

```
00612 /*ENABLE
00613 *DISABLE
00614 */
00615 #define SPI4EN DISABLE
00616
00617 /*****
00618
00619 /*options:
00620 *ENABLE
00621 *DISABLE
00622 */
00623 #define SPI1EN DISABLE
00624
00625 *****/
00626
00627 /*options:
00628 *ENABLE
00629 *DISABLE
00630 */
00631 #define SDIOEN DISABLE
00632
00633 *****/
00634
00635 /*options:
00636 *ENABLE
00637 *DISABLE
00638 */
00639 #define ADC1EN DISABLE
00640
00641 *****/
00642
00643 /*options:
00644 *ENABLE
00645 *DISABLE
00646 */
00647 #define USART6EN DISABLE
00648
00649 *****/
00650
00651 /*options:
00652 *ENABLE
00653 *DISABLE
00654 */
00655 #define USART1EN DISABLE
00656
00657 *****/
00658
00659 /*options:
00660 *ENABLE
00661 *DISABLE
00662 */
00663 #define TIM1EN DISABLE
00664
00665 *****/
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682 #endif /* MCAL_RCC_MRCC_CONFIG_H */
```

7.112 COTS/MCAL/RCC/MRCC_interface.h File Reference

This file contains the interfacing information for the RCC module.

Macros

- `#define RCC_AHB1 1`

- #define RCC_AHB2 2
AHB2 Bus.
- #define RCC_APB1 3
APB1 Bus.
- #define RCC_APB2 4
APB2 Bus.
- #define RCC_AHB1LPENR 5
peripheral clock enable in low power mode register
- #define AHB1ENR_DMA2EN 22
Enable CLK on DMA2 peripheral.
- #define AHB1ENR_DMA1EN 21
Enable CLK on DMA1 peripheral.
- #define AHB1ENR_CRCEN 12
Enable CLK on CRC peripheral.
- #define AHB1ENR_GPIOHEN 7
Enable CLK on GPIOH peripheral.
- #define AHB1ENR_GPIOEEN 4
Enable CLK on GPIOE peripheral.
- #define AHB1ENR_GPIODEN 3
Enable CLK on GPIOD peripheral.
- #define AHB1ENR_GPIOCEN 2
Enable CLK on GPIOC peripheral.
- #define AHB1ENR_GPIOBEN 1
Enable CLK on GPIOB peripheral.
- #define AHB1ENR_GPIOAEN 0
Enable CLK on GPIOA peripheral.
- #define AHB2ENR_OTGFSEN 7
Enable CLK on OTGFS peripheral.
- #define APB1ENR_PWREN 28
Enable CLK on PWR peripheral.
- #define APB1ENR_I2C3EN 23
Enable CLK on I2C3 peripheral.
- #define APB1ENR_I2C2EN 22
Enable CLK on I2C2 peripheral.
- #define APB1ENR_I2C1EN 21
Enable CLK on I2C1 peripheral.
- #define APB1ENR_USART2EN 17
Enable CLK on USART2 peripheral.
- #define APB1ENR_SPI3EN 15
Enable CLK on SPI3 peripheral.
- #define APB1ENR_SPI2EN 14
Enable CLK on SPI2 peripheral.
- #define APB1ENR_WWDGEN 11
Enable CLK on WWD peripheral.
- #define APB1ENR_TIM5EN 3
Enable CLK on TIM5 peripheral.
- #define APB1ENR_TIM4EN 2
Enable CLK on TIM4 peripheral.
- #define APB1ENR_TIM3EN 1
Enable CLK on TIM3 peripheral.

- `#define APB1ENR_TIM2EN 0`
Enable CLK on TIM2 peripheral.
- `#define APB2ENR_TIM11EN 18`
Enable CLK on TIM11 peripheral.
- `#define APB2ENR_TIM10EN 17`
Enable CLK on TIM10 peripheral.
- `#define APB2ENR_TIM9EN 16`
Enable CLK on TIM9 peripheral.
- `#define APB2ENR_SYSCFGGEN 14`
Enable CLK on SYSCFG peripheral.
- `#define APB2ENR_SPI4EN 13`
Enable CLK on SPI4 peripheral.
- `#define APB2ENR_SPI1EN 12`
Enable CLK on SPI1 peripheral.
- `#define APB2ENR_SDIOEN 11`
Enable CLK on SDIO peripheral.
- `#define APB2ENR_ADC1EN 8`
Enable CLK on ADC1 peripheral.
- `#define APB2ENR_USART6EN 5`
Enable CLK on USART6 peripheral.
- `#define APB2ENR_USART1EN 4`
Enable CLK on USART1 peripheral.
- `#define APB2ENR_TIM1EN 0`
Enable CLK on TIM1 peripheral.
- `#define AHB1LPENR_FLITFLPEN 15`
Enable CLK on FLITFLP peripheral.

Functions

- `void MRCC_vInit (void)`
Initialize the RCC with a certain configurations.
- `void MRCC_vEnablePeriphralCLK (u32_t A_u32BusID, u32_t A_u32PeriphralID)`
Enabling the CLK on a specific Peripheral.
- `void MRCC_vDisablePeriphralCLK (u32_t A_u32BusID, u32_t A_u32PeriphralID)`
Disabling the CLK on a specific Peripheral.

7.112.1 Detailed Description

This file contains the interfacing information for the RCC module.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

11/9/2022

Definition in file [MRCC_interface.h](#).

7.112.2 Function Documentation

7.112.2.1 MRCC_vInit()

```
void MRCC_vInit (
    void )
```

Initialize the RCC with a certain configurations.

Definition at line 29 of file [MRCC_program.c](#).

```
00030 {
00031     // PLLI2S (ON/OFF).
00032 #if PLLI2S == ENABLE
00033     SET_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00036
00037 #elif PLLI2S == DISABLE
00038     CLR_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00040
00041 #endif
00042
00043
00044     // PLL (ON/OFF).
00045 #if PLL == ENABLE
00046     SET_BIT( RCC->CR, RCC_CR_PLLON ) ;
00048
00049 #elif PLL == DISABLE
00050     CLR_BIT( RCC->CR, RCC_CR_PLLON ) ;
00052
00053 #endif
00054
00055
00056     // CSS (ON/OFF).
00057 #if CSS == ENABLE
00058     SET_BIT( RCC->CR, RCC_CR_CSSON ) ;
00060
00061 #elif CSS == DISABLE
00062     CLR_BIT( RCC->CR, RCC_CR_CSSON ) ;
00064
00065 #endif
00066
00067
00068     // HSEBYP.
00069 #if HSEBYP == BYBASED
00070     SET_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00072
00073 #elif HSEBYP == NOTBYBASED
00074     CLR_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00076
00077 #endif
00078
00079
00080     // Select CLK switch (HSI/HSE/PLL).
00081 #if SW == HSI
00082
00083     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00084     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00085
00086 #elif SW == HSE
00087
00088     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00089     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00090
00091 #elif SW == PLLCLK
00092
00093     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00094     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00095
```

```

00096 #endif
00097
00098     // Select CLK switch status (HSI/HSE/PLL).
00099 #if SWS == HSI
00100         CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00101         CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00103
00104 #elif SWS == HSE
00105
00106         SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00107         CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00108
00109 #elif SWS == PLLCLK
00110
00111         CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00112         SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00113
00114 #endif
00115
00116
00117     // MCO2 selection:
00118 #if MCO2 == SYSCLK
00119
00120         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00121         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00122
00123 #elif MCO2 == PLLI2SCLK
00124
00125         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00126         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00127
00128 #elif MCO2 == HSE
00129
00130         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00131         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00132
00133 #elif MCO2 == PLLCLK
00134
00135         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00136         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00137
00138 #endif
00139
00140
00141     // MCO2 prescaler:
00142 #if MCO2PRE == NoDivision
00143
00144         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00145         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00146         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00147
00148 #elif MCO2PRE == DivisionBy2
00149
00150         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00151         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00152         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00153
00154 #elif MCO2PRE == DivisionBy3
00155
00156         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00157         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00158         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00159
00160 #elif MCO2PRE == DivisionBy4
00161
00162         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00163         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00164         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00165
00166 #elif MCO2PRE == DivisionBy5
00167
00168         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00169         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00170         SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00171
00172 #endif
00173
00174
00175     // MCO1 selection:
00176 #if MCO1 == HSI
00177
00178         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00179         CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00180
00181 #elif MCO1 == LSE
00182

```

```

00183     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00184     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00185
00186 #elif MCO1 == HSE
00187
00188     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00189     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00190
00191 #elif MCO1 == PLLCLK
00192
00193     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00194     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00195
00196 #endif
00197
00198
00199 // MCO1 prescaler:
00200 #if MCO1PRE == NoDivision
00201
00202     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00203     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00204     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00205
00206 #elif MCO1PRE == DivisionBy2
00207
00208     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00209     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00210     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00211
00212 #elif MCO1PRE == DivisionBy3
00213
00214     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00215     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00216     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00217
00218 #elif MCO1PRE == DivisionBy4
00219
00220     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00221     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00222     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00223
00224 #elif MCO1PRE == DivisionBy5
00225
00226     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00227     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00228     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00229
00230 #endif
00231
00232
00233 // AHB prescalers:
00234 #if HPRE == NoDivision
00235
00236     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00237     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00238     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00239     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00240
00241 #elif HPRE == SYSCLKby2
00242
00243     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00244     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00245     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00246     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00247
00248 #elif HPRE == SYSCLKby4
00249
00250     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00251     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00252     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00253     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00254
00255 #elif HPRE == SYSCLKby8
00256
00257     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00258     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00259     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00260     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00261
00262 #elif HPRE == SYSCLKby16
00263
00264     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00265     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00266     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00267     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00268
00269 #elif HPRE == SYSCLKby64

```

```

00270
00271     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00272     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00273     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00274     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00275
00276 #elif HPRE == SYSCLKby128
00277
00278     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00279     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00280     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00281     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00282
00283 #elif HPRE == SYSCLKby256
00284
00285     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00286     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00287     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00288     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00289
00290 #elif HPRE == SYSCLKby512
00291
00292     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00293     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00294     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00295     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00296
00297 #endif
00298
00299
00300 // APB1 prescalers:
00301 #if PPREG == NoDivision
00302
00303     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00304     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00305     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00306
00307 #elif PPREG == AHBby2
00308
00309     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00310     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00311     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00312
00313 #elif PPREG == AHBby4
00314
00315     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00316     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00317     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00318
00319 #elif PPREG == AHBby8
00320
00321     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00322     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00323     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00324
00325 #elif PPREG == AHBby16
00326
00327     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00328     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00329     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00330
00331 #endif
00332
00333 // APB2 prescalers:
00334 #if PPREG == NoDivision
00335
00336     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00337     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00338     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00339
00340 #elif PPREG == AHBby2
00341
00342     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00343     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00344     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00345
00346 #elif PPREG == AHBby4
00347
00348     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00349     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00350     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00351
00352 #elif PPREG == AHBby8
00353
00354     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00355     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00356     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;

```

```

00357
00358 #elif PPRE2 == AHBby16
00359
00360     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00361     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00362     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00363
00364 #endif
00365
00366
00367 // PLL configurations:
00368
00369
00370
00371     // Enable the selected CLK (HSI ON/HSE ON/PLL ON): //
00372     // HSE (ON/OFF).
00373 #if HSE_EN == ENABLE
00374
00375     SET_BIT( RCC->CR, RCC_CR_HSEON ) ;
00376
00377 #elif HSE_EN == DISABLE
00378
00379     CLR_BIT( RCC->CR, RCC_CR_HSEON ) ;
00380
00381 #endif
00382
00383
00384     // HSI (ON/OFF).
00385 #if HSI_EN == ENABLE
00386
00387     SET_BIT( RCC->CR, RCC_CR_HSION ) ;
00388
00389 #elif HSI_EN == DISABLE
00390
00391     CLR_BIT( RCC->CR, RCC_CR_HSION ) ;
00392
00393 #endif
00394
00395 }

```

References [CLR_BIT](#), [RCC](#), and [SET_BIT](#).

7.112.2.2 MRCC_vEnablePeriphralCLK()

```

void MRCC_vEnablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )

```

Enabling the CLK on a specific Peripheral.

Parameters

in	<i>A_u32BusID</i>	Bus ID.
in	<i>A_u32PeriphralID</i>	Peripheral ID.

Definition at line 400 of file [MRCC_program.c](#).

```

00401 {
00402     switch( A_u32BusID )
00403     {
00404         case RCC_AHB1 :
00405             SET_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00406             break ;
00407         case RCC_AHB2 :
00408             SET_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00409             break ;
00410         case RCC_APB1 :
00411             SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00412             break ;
00413         case RCC_APB2 :
00414             SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00415             break ;
00416     }
00417 }

```

```

00415     SET_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00416
00417     break ;
00418
00419
00420
00421     case RCC_APB1 :
00422
00423         SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00424
00425     break ;
00426
00427
00428     case RCC_APB2 :
00429
00430         SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00431
00432     break ;
00433
00434     case RCC_AHB1LPENR:
00435
00436         SET_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00437
00438     break ;
00439
00440     default:
00441
00442         // Error wrong Bus ID
00443
00444     break ;
00445
00446 }
00447
00448 }
```

References [RCC](#), [RCC_AHB1](#), [RCC_AHB1LPENR](#), [RCC_AHB2](#), [RCC_APB1](#), [RCC_APB2](#), and [SET_BIT](#).

Referenced by [MADC_vEnable\(\)](#), [MADC_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.112.2.3 MRCC_vDisablePeriphralCLK()

```
void MRCC_vDisablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )
```

Disabling the CLK on a specific Peripheral.

Parameters

in	<i>A_u32BusID</i>	Bus ID.
in	<i>A_u32PeriphralID</i>	Peripheral ID.

Definition at line 453 of file [MRCC_program.c](#).

```

00454 {
00455
00456     switch( A_u32BusID )
00457     {
00458
00459     case RCC_AHB1 :
00460
00461         CLR_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00462
00463     break ;
00464
00465
00466     case RCC_AHB2 :
00467
00468         CLR_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00469 }
```

```

00470
00471     break ;
00472
00473
00474     case RCC_APB1 :
00475
00476         CLR_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00477
00478     break ;
00479
00480
00481     case RCC_APB2 :
00482
00483         CLR_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00484
00485     break ;
00486
00487
00488     case RCC_AHB1LPENR:
00489
00490         CLR_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00491
00492     break ;
00493
00494
00495     default:
00496
00497     // Error wrong Bus ID
00498
00499     break ;
00500
00501 }
00502
00503 }
```

References [CLR_BIT](#), [RCC](#), [RCC_AHB1](#), [RCC_AHB1LPENR](#), [RCC_AHB2](#), [RCC_APB1](#), and [RCC_APB2](#).

Referenced by [MADC_vDisable\(\)](#).

7.113 MRCC_interface.h

[Go to the documentation of this file.](#)

```

00001 /* Header file guard */
00010 #ifndef MCAL_RCC_MRCC_INTERFACE_H_
00011 #define MCAL_RCC_MRCC_INTERFACE_H_
00012
00013 /*****+
00014 *          Functions prototypes
00015 +*****/
00016
00020 void MRCC_vInit(void);
00021
00027 void MRCC_vEnablePeriphralCLK(u32_t A_u32BusID, u32_t A_u32PeriphralID);
00028
00034 void MRCC_vDisablePeriphralCLK(u32_t A_u32BusID, u32_t A_u32PeriphralID);
00035
00036 /*****+
00037 *          Interfacing macros
00038 +*****/
00039
00051 #define RCC_AHB1 1
00052
00058 #define RCC_AHB2 2
00059
00065 #define RCC_APB1 3
00066
00072 #define RCC_APB2 4
00073
00079 #define RCC_AHB1LPENR 5
00080
00094 #define AHB1ENR_DMA2EN 22
00095
00101 #define AHB1ENR_DMA1EN 21
00102
00108 #define AHB1ENR_CRCEN 12
00109
00115 #define AHB1ENR_GPIOHEN 7
00116
```

```

00122 #define AHB1ENR_GPIOEEN 4
00123
00129 #define AHB1ENR_GPIODEN 3
00130
00136 #define AHB1ENR_GPIOCEN 2
00137
00143 #define AHB1ENR_GPIOBEN 1
00144
00150 #define AHB1ENR_GPIOAEN 0
00151
00157 #define AHB2ENR_OTGFSEN 7
00158
00164 #define APB1ENR_PWREN 28
00165
00171 #define APB1ENR_I2C3EN 23
00172
00178 #define APB1ENR_I2C2EN 22
00179
00185 #define APB1ENR_I2C1EN 21
00186
00192 #define APB1ENR_USART2EN 17
00193
00199 #define APB1ENR_SPI3EN 15
00200
00206 #define APB1ENR_SPI2EN 14
00207
00213 #define APB1ENR_WWDGEN 11
00214
00220 #define APB1ENR_TIM5EN 3
00221
00227 #define APB1ENR_TIM4EN 2
00228
00234 #define APB1ENR_TIM3EN 1
00235
00241 #define APB1ENR_TIM2EN 0
00242
00248 #define APB2ENR_TIM11EN 18
00249
00255 #define APB2ENR_TIM10EN 17
00256
00262 #define APB2ENR_TIM9EN 16
00263
00269 #define APB2ENR_SYSCFGEN 14
00270
00276 #define APB2ENR_SPI4EN 13
00277
00283 #define APB2ENR_SPI1EN 12
00284
00290 #define APB2ENR_SDIOEN 11
00291
00297 #define APB2ENR_ADC1EN 8
00298
00304 #define APB2ENR_USART6EN 5
00305
00311 #define APB2ENR_USART1EN 4
00312
00318 #define APB2ENR_TIM1EN 0
00319
00325 #define AHB1LPENR_FLITFLPEN 15
00326
00329 #endif /* MCAL_RCC_MRCC_INTERFACE_H_ */

```

7.114 COTS/MCAL/RCC/MRCC_private.h File Reference

This file contains the registers information and addresses for the RCC module.

Data Structures

- struct [RCC_MemoryMapType](#)
RCC declaration structure for its registers.

Macros

- #define RCC_BASE_ADDRESS 0x40023800
- #define RCC (volatile P2VAR(RCC_MemoryMapType)) (RCC_BASE_ADDRESS))
- #define RCC_CR_PLLI2SRDY 27
- #define RCC_CR_PLLI2SON 26
- #define RCC_CR_PLLRDY 25
- #define RCC_CR_PLLON 24
- #define RCC_CR_CSSON 19
- #define RCC_CR_HSEBYP 18
- #define RCC_CR_HSERDY 17
- #define RCC_CR_HSEON 16
- #define RCC_CR_HSIRDY 1
- #define RCC_CR_HSION 0
- #define RCC_PLLCFGR_PLLQ_b0 24
- #define RCC_PLLCFGR_PLLQ_b1 25
- #define RCC_PLLCFGR_PLLQ_b2 26
- #define RCC_PLLCFGR_PLLQ_b3 27
- #define RCC_PLLCFGR_PLLSRC 22
- #define RCC_PLLCFGR_PLLP_b0 16
- #define RCC_PLLCFGR_PLLP_b1 17
- #define RCC_PLLCFGR_PLLN_b0 6
- #define RCC_PLLCFGR_PLLN_b1 7
- #define RCC_PLLCFGR_PLLN_b2 8
- #define RCC_PLLCFGR_PLLN_b3 9
- #define RCC_PLLCFGR_PLLN_b4 10
- #define RCC_PLLCFGR_PLLN_b5 11
- #define RCC_PLLCFGR_PLLN_b6 12
- #define RCC_PLLCFGR_PLLN_b7 13
- #define RCC_PLLCFGR_PLLN_b8 14
- #define RCC_PLLCFGR_PLLM_b0 0
- #define RCC_PLLCFGR_PLLM_b1 1
- #define RCC_PLLCFGR_PLLM_b2 2
- #define RCC_PLLCFGR_PLLM_b3 3
- #define RCC_PLLCFGR_PLLM_b4 4
- #define RCC_PLLCFGR_PLLM_b5 5
- #define RCC_CFGR_MOC2_b0 30
- #define RCC_CFGR_MOC2_b1 31
- #define RCC_CFGR_MOC2PRE_b0 27
- #define RCC_CFGR_MOC2PRE_b1 28
- #define RCC_CFGR_MOC2PRE_b2 29
- #define RCC_CFGR_MOC1PRE_b0 24
- #define RCC_CFGR_MOC1PRE_b1 25
- #define RCC_CFGR_MOC1PRE_b2 26
- #define RCC_CFGR_I2SSRC 23
- #define RCC_CFGR_MOC1_b0 21
- #define RCC_CFGR_MOC1_b1 22
- #define RCC_CFGR_RTCPRE_b0 16
- #define RCC_CFGR_RTCPRE_b1 17
- #define RCC_CFGR_RTCPRE_b2 18
- #define RCC_CFGR_RTCPRE_b3 19
- #define RCC_CFGR_RTCPRE_b4 20
- #define RCC_CFGR_PPREG2_b0 13
- #define RCC_CFGR_PPREG2_b1 14
- #define RCC_CFGR_PPREG2_b2 15

- #define RCC_CFGR_PPRE1_b0 10
- #define RCC_CFGR_PPRE1_b1 11
- #define RCC_CFGR_PPRE1_b2 12
- #define RCC_CFGR_HPRE_b0 4
- #define RCC_CFGR_HPRE_b1 5
- #define RCC_CFGR_HPRE_b2 6
- #define RCC_CFGR_HPRE_b3 7
- #define RCC_CFGR_SWS_b0 2
- #define RCC_CFGR_SWS_b1 3
- #define RCC_CFGR_SW_b0 0
- #define RCC_CFGR_SW_b1 1
- #define RCC_AHB1ENR_DMA2EN 22
- #define RCC_AHB1ENR_DMA1EN 21
- #define RCC_AHB1ENR_CRCEN 12
- #define RCC_AHB1ENR_GPIOHEN 7
- #define RCC_AHB1ENR_GPIOEEN 4
- #define RCC_AHB1ENR_GPIODEN 3
- #define RCC_AHB1ENR_GPIOCEN 2
- #define RCC_AHB1ENR_GPIOBEN 1
- #define RCC_AHB1ENR_GPIOAEN 0
- #define RCC_AHB2ENR_OTGFSEN 7
- #define RCC_APB1ENR_PWREN 28
- #define RCC_APB1ENR_I2C3EN 23
- #define RCC_APB1ENR_I2C2EN 22
- #define RCC_APB1ENR_I2C1EN 21
- #define RCC_APB1ENR_USART2EN 17
- #define RCC_APB1ENR_SPI3EN 15
- #define RCC_APB1ENR_SPI2EN 14
- #define RCC_APB1ENR_WWDGEN 11
- #define RCC_APB1ENR_TIM5EN 3
- #define RCC_APB1ENR_TIM4EN 2
- #define RCC_APB1ENR_TIM3EN 1
- #define RCC_APB1ENR_TIM2EN 0
- #define RCC_APB2ENR_TIM11EN 18
- #define RCC_APB2ENR_TIM10EN 17
- #define RCC_APB2ENR_TIM9EN 16
- #define RCC_APB2ENR_SYSCFGEN 14
- #define RCC_APB2ENR_SPI4EN 13
- #define RCC_APB2ENR_SPI1EN 12
- #define RCC_APB2ENR_SDIOEN 11
- #define RCC_APB2ENR_ADC1EN 8
- #define RCC_APB2ENR_USART6EN 5
- #define RCC_APB2ENR_USART1EN 4
- #define RCC_APB2ENR_TIM1EN 0
- #define RCC_AHB1LPENR_FLITFLPEN 15
- #define ENABLE 1
- #define DISABLE 2
- #define BYBASED 1
- #define NOTBYBASED 2
- #define SYSCLK 1
- #define PLLI2SCLK 5
- #define HSE 3
- #define PLLCLK 4
- #define NoDivision 1
- #define DivisionBy2 2

- #define DivisionBy3 3
- #define DivisionBy4 4
- #define DivisionBy5 5
- #define I2S_CKIN 2
- #define HSI 1
- #define LSE 2
- #define PLLCLK 4
- #define SYSCLKby2 2
- #define SYSCLKby4 3
- #define SYSCLKby8 4
- #define SYSCLKby16 5
- #define SYSCLKby64 6
- #define SYSCLKby128 7
- #define SYSCLKby256 8
- #define SYSCLKby512 9
- #define NoCLK0 1
- #define NoCLK1 2
- #define HSEby2 3
- #define HSEby3 4
- #define HSEby4 5
- #define HSEby5 6
- #define HSEby6 7
- #define HSEby7 8
- #define HSEby8 9
- #define HSEby9 10
- #define HSEby10 11
- #define HSEby11 12
- #define HSEby12 13
- #define HSEby13 14
- #define HSEby14 15
- #define HSEby15 16
- #define HSEby16 17
- #define HSEby17 18
- #define HSEby18 19
- #define HSEby19 20
- #define HSEby20 21
- #define HSEby21 22
- #define HSEby22 23
- #define HSEby23 24
- #define HSEby24 25
- #define HSEby25 26
- #define HSEby26 27
- #define HSEby27 28
- #define HSEby28 39
- #define HSEby29 30
- #define HSEby30 31
- #define HSEby31 32
- #define AHBby2 1
- #define AHBby4 2
- #define AHBby8 3
- #define AHBby16 4
- #define Equal_0 0
- #define Equal_1 1
- #define Equal_2 2
- #define Equal_3 3

- #define Equal_4 4
- #define Equal_5 5
- #define Equal_6 6
- #define Equal_7 7
- #define Equal_8 8
- #define Equal_9 9
- #define Equal_10 10
- #define Equal_11 11
- #define Equal_12 12
- #define Equal_13 13
- #define Equal_14 14
- #define Equal_15 15
- #define Equal_16 16
- #define Equal_17 17
- #define Equal_18 18
- #define Equal_19 19
- #define Equal_20 20
- #define Equal_21 21
- #define Equal_22 22
- #define Equal_23 23
- #define Equal_24 24
- #define Equal_25 25
- #define Equal_26 26
- #define Equal_27 27
- #define Equal_28 28
- #define Equal_29 29
- #define Equal_30 30
- #define Equal_31 31
- #define Equal_32 32
- #define Equal_33 33
- #define Equal_34 34
- #define Equal_35 35
- #define Equal_36 36
- #define Equal_37 37
- #define Equal_38 38
- #define Equal_39 39
- #define Equal_40 40
- #define Equal_41 41
- #define Equal_42 42
- #define Equal_43 43
- #define Equal_44 44
- #define Equal_45 45
- #define Equal_46 46
- #define Equal_47 47
- #define Equal_48 48
- #define Equal_49 49
- #define Equal_50 50
- #define Equal_51 51
- #define Equal_52 52
- #define Equal_53 53
- #define Equal_54 54
- #define Equal_55 55
- #define Equal_56 56
- #define Equal_57 57
- #define Equal_58 58

- #define Equal_59 59
- #define Equal_60 60
- #define Equal_61 61
- #define Equal_62 62
- #define Equal_63 63

7.114.1 Detailed Description

This file contains the registers information and addresses for the RCC module.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

11/9/2022

Definition in file [MRCC_private.h](#).

7.114.2 Macro Definition Documentation

7.114.2.1 RCC_BASE_ADDRESS

```
#define RCC_BASE_ADDRESS 0x40023800
```

RCC Base Address

Definition at line [211](#) of file [MRCC_private.h](#).

7.114.2.2 RCC

```
#define RCC ( (volatile P2VAR(RCC_MemoryMapType) ) (RCC_BASE_ADDRESS) )
```

RCC register

Definition at line [218](#) of file [MRCC_private.h](#).

7.114.2.3 RCC_CR_PLLI2SRDY

```
#define RCC_CR_PLLI2SRDY 27
```

Definition at line [225](#) of file [MRCC_private.h](#).

7.114.2.4 RCC_CR_PLLI2SON

```
#define RCC_CR_PLLI2SON 26
```

Definition at line [226](#) of file [MRCC_private.h](#).

7.114.2.5 RCC_CR_PLLRDY

```
#define RCC_CR_PLLRDY 25
```

Definition at line [227](#) of file [MRCC_private.h](#).

7.114.2.6 RCC_CR_PLLON

```
#define RCC_CR_PLLON 24
```

Definition at line [228](#) of file [MRCC_private.h](#).

7.114.2.7 RCC_CR_CSSON

```
#define RCC_CR_CSSON 19
```

Definition at line [229](#) of file [MRCC_private.h](#).

7.114.2.8 RCC_CR_HSEBYP

```
#define RCC_CR_HSEBYP 18
```

Definition at line [230](#) of file [MRCC_private.h](#).

7.114.2.9 RCC_CR_HSERDY

```
#define RCC_CR_HSERDY 17
```

Definition at line 231 of file [MRCC_private.h](#).

7.114.2.10 RCC_CR_HSEON

```
#define RCC_CR_HSEON 16
```

Definition at line 232 of file [MRCC_private.h](#).

7.114.2.11 RCC_CR_HSIRDY

```
#define RCC_CR_HSIRDY 1
```

Definition at line 233 of file [MRCC_private.h](#).

7.114.2.12 RCC_CR_HSION

```
#define RCC_CR_HSION 0
```

Definition at line 234 of file [MRCC_private.h](#).

7.114.2.13 RCC_PLLCFGR_PLLQ_b0

```
#define RCC_PLLCFGR_PLLQ_b0 24
```

Definition at line 237 of file [MRCC_private.h](#).

7.114.2.14 RCC_PLLCFGR_PLLQ_b1

```
#define RCC_PLLCFGR_PLLQ_b1 25
```

Definition at line 238 of file [MRCC_private.h](#).

7.114.2.15 RCC_PLLCFGR_PLLQ_b2

```
#define RCC_PLLCFGR_PLLQ_b2 26
```

Definition at line [239](#) of file [MRCC_private.h](#).

7.114.2.16 RCC_PLLCFGR_PLLQ_b3

```
#define RCC_PLLCFGR_PLLQ_b3 27
```

Definition at line [240](#) of file [MRCC_private.h](#).

7.114.2.17 RCC_PLLCFGR_PLLSRC

```
#define RCC_PLLCFGR_PLLSRC 22
```

Definition at line [241](#) of file [MRCC_private.h](#).

7.114.2.18 RCC_PLLCFGR_PLLP_b0

```
#define RCC_PLLCFGR_PLLP_b0 16
```

Definition at line [242](#) of file [MRCC_private.h](#).

7.114.2.19 RCC_PLLCFGR_PLLP_b1

```
#define RCC_PLLCFGR_PLLP_b1 17
```

Definition at line [243](#) of file [MRCC_private.h](#).

7.114.2.20 RCC_PLLCFGR_PLLN_b0

```
#define RCC_PLLCFGR_PLLN_b0 6
```

Definition at line [244](#) of file [MRCC_private.h](#).

7.114.2.21 RCC_PLLCFGR_PLLN_b1

```
#define RCC_PLLCFGR_PLLN_b1 7
```

Definition at line [245](#) of file [MRCC_private.h](#).

7.114.2.22 RCC_PLLCFGR_PLLN_b2

```
#define RCC_PLLCFGR_PLLN_b2 8
```

Definition at line [246](#) of file [MRCC_private.h](#).

7.114.2.23 RCC_PLLCFGR_PLLN_b3

```
#define RCC_PLLCFGR_PLLN_b3 9
```

Definition at line [247](#) of file [MRCC_private.h](#).

7.114.2.24 RCC_PLLCFGR_PLLN_b4

```
#define RCC_PLLCFGR_PLLN_b4 10
```

Definition at line [248](#) of file [MRCC_private.h](#).

7.114.2.25 RCC_PLLCFGR_PLLN_b5

```
#define RCC_PLLCFGR_PLLN_b5 11
```

Definition at line [249](#) of file [MRCC_private.h](#).

7.114.2.26 RCC_PLLCFGR_PLLN_b6

```
#define RCC_PLLCFGR_PLLN_b6 12
```

Definition at line [250](#) of file [MRCC_private.h](#).

7.114.2.27 RCC_PLLCFGR_PLLN_b7

```
#define RCC_PLLCFGR_PLLN_b7 13
```

Definition at line 251 of file [MRCC_private.h](#).

7.114.2.28 RCC_PLLCFGR_PLLN_b8

```
#define RCC_PLLCFGR_PLLN_b8 14
```

Definition at line 252 of file [MRCC_private.h](#).

7.114.2.29 RCC_PLLCFGR_PLLM_b0

```
#define RCC_PLLCFGR_PLLM_b0 0
```

Definition at line 253 of file [MRCC_private.h](#).

7.114.2.30 RCC_PLLCFGR_PLLM_b1

```
#define RCC_PLLCFGR_PLLM_b1 1
```

Definition at line 254 of file [MRCC_private.h](#).

7.114.2.31 RCC_PLLCFGR_PLLM_b2

```
#define RCC_PLLCFGR_PLLM_b2 2
```

Definition at line 255 of file [MRCC_private.h](#).

7.114.2.32 RCC_PLLCFGR_PLLM_b3

```
#define RCC_PLLCFGR_PLLM_b3 3
```

Definition at line 256 of file [MRCC_private.h](#).

7.114.2.33 RCC_PLLCFGR_PLLM_b4

```
#define RCC_PLLCFGR_PLLM_b4 4
```

Definition at line [257](#) of file [MRCC_private.h](#).

7.114.2.34 RCC_PLLCFGR_PLLM_b5

```
#define RCC_PLLCFGR_PLLM_b5 5
```

Definition at line [258](#) of file [MRCC_private.h](#).

7.114.2.35 RCC_CFGR_MOC2_b0

```
#define RCC_CFGR_MOC2_b0 30
```

Definition at line [261](#) of file [MRCC_private.h](#).

7.114.2.36 RCC_CFGR_MOC2_b1

```
#define RCC_CFGR_MOC2_b1 31
```

Definition at line [262](#) of file [MRCC_private.h](#).

7.114.2.37 RCC_CFGR_MOC2PRE_b0

```
#define RCC_CFGR_MOC2PRE_b0 27
```

Definition at line [263](#) of file [MRCC_private.h](#).

7.114.2.38 RCC_CFGR_MOC2PRE_b1

```
#define RCC_CFGR_MOC2PRE_b1 28
```

Definition at line [264](#) of file [MRCC_private.h](#).

7.114.2.39 RCC_CFGR_MOC2PRE_b2

```
#define RCC_CFGR_MOC2PRE_b2 29
```

Definition at line [265](#) of file [MRCC_private.h](#).

7.114.2.40 RCC_CFGR_MOC1PRE_b0

```
#define RCC_CFGR_MOC1PRE_b0 24
```

Definition at line [266](#) of file [MRCC_private.h](#).

7.114.2.41 RCC_CFGR_MOC1PRE_b1

```
#define RCC_CFGR_MOC1PRE_b1 25
```

Definition at line [267](#) of file [MRCC_private.h](#).

7.114.2.42 RCC_CFGR_MOC1PRE_b2

```
#define RCC_CFGR_MOC1PRE_b2 26
```

Definition at line [268](#) of file [MRCC_private.h](#).

7.114.2.43 RCC_CFGR_I2SSRC

```
#define RCC_CFGR_I2SSRC 23
```

Definition at line [269](#) of file [MRCC_private.h](#).

7.114.2.44 RCC_CFGR_MOC1_b0

```
#define RCC_CFGR_MOC1_b0 21
```

Definition at line [270](#) of file [MRCC_private.h](#).

7.114.2.45 RCC_CFGR_MOC1_b1

```
#define RCC_CFGR_MOC1_b1 22
```

Definition at line [271](#) of file [MRCC_private.h](#).

7.114.2.46 RCC_CFGR_RTCPRE_b0

```
#define RCC_CFGR_RTCPRE_b0 16
```

Definition at line [272](#) of file [MRCC_private.h](#).

7.114.2.47 RCC_CFGR_RTCPRE_b1

```
#define RCC_CFGR_RTCPRE_b1 17
```

Definition at line [273](#) of file [MRCC_private.h](#).

7.114.2.48 RCC_CFGR_RTCPRE_b2

```
#define RCC_CFGR_RTCPRE_b2 18
```

Definition at line [274](#) of file [MRCC_private.h](#).

7.114.2.49 RCC_CFGR_RTCPRE_b3

```
#define RCC_CFGR_RTCPRE_b3 19
```

Definition at line [275](#) of file [MRCC_private.h](#).

7.114.2.50 RCC_CFGR_RTCPRE_b4

```
#define RCC_CFGR_RTCPRE_b4 20
```

Definition at line [276](#) of file [MRCC_private.h](#).

7.114.2.51 RCC_CFGR_PPREG_b0

```
#define RCC_CFGR_PPREG_b0 13
```

Definition at line [277](#) of file [MRCC_private.h](#).

7.114.2.52 RCC_CFGR_PPREG_b1

```
#define RCC_CFGR_PPREG_b1 14
```

Definition at line [278](#) of file [MRCC_private.h](#).

7.114.2.53 RCC_CFGR_PPREG_b2

```
#define RCC_CFGR_PPREG_b2 15
```

Definition at line [279](#) of file [MRCC_private.h](#).

7.114.2.54 RCC_CFGR_PPREG1_b0

```
#define RCC_CFGR_PPREG1_b0 10
```

Definition at line [280](#) of file [MRCC_private.h](#).

7.114.2.55 RCC_CFGR_PPREG1_b1

```
#define RCC_CFGR_PPREG1_b1 11
```

Definition at line [281](#) of file [MRCC_private.h](#).

7.114.2.56 RCC_CFGR_PPREG1_b2

```
#define RCC_CFGR_PPREG1_b2 12
```

Definition at line [282](#) of file [MRCC_private.h](#).

7.114.2.57 RCC_CFGR_HPREG_b0

```
#define RCC_CFGR_HPREG_b0 4
```

Definition at line [283](#) of file [MRCC_private.h](#).

7.114.2.58 RCC_CFGR_HPREG_b1

```
#define RCC_CFGR_HPREG_b1 5
```

Definition at line [284](#) of file [MRCC_private.h](#).

7.114.2.59 RCC_CFGR_HPREG_b2

```
#define RCC_CFGR_HPREG_b2 6
```

Definition at line [285](#) of file [MRCC_private.h](#).

7.114.2.60 RCC_CFGR_HPREG_b3

```
#define RCC_CFGR_HPREG_b3 7
```

Definition at line [286](#) of file [MRCC_private.h](#).

7.114.2.61 RCC_CFGR_SWS_b0

```
#define RCC_CFGR_SWS_b0 2
```

Definition at line [287](#) of file [MRCC_private.h](#).

7.114.2.62 RCC_CFGR_SWS_b1

```
#define RCC_CFGR_SWS_b1 3
```

Definition at line [288](#) of file [MRCC_private.h](#).

7.114.2.63 RCC_CFGR_SW_b0

```
#define RCC_CFGR_SW_b0 0
```

Definition at line 289 of file [MRCC_private.h](#).

7.114.2.64 RCC_CFGR_SW_b1

```
#define RCC_CFGR_SW_b1 1
```

Definition at line 290 of file [MRCC_private.h](#).

7.114.2.65 RCC_AHB1ENR_DMA2EN

```
#define RCC_AHB1ENR_DMA2EN 22
```

Definition at line 293 of file [MRCC_private.h](#).

7.114.2.66 RCC_AHB1ENR_DMA1EN

```
#define RCC_AHB1ENR_DMA1EN 21
```

Definition at line 294 of file [MRCC_private.h](#).

7.114.2.67 RCC_AHB1ENR_CRCEN

```
#define RCC_AHB1ENR_CRCEN 12
```

Definition at line 295 of file [MRCC_private.h](#).

7.114.2.68 RCC_AHB1ENR_GPIOHEN

```
#define RCC_AHB1ENR_GPIOHEN 7
```

Definition at line 296 of file [MRCC_private.h](#).

7.114.2.69 RCC_AHB1ENR_GPIOEEN

```
#define RCC_AHB1ENR_GPIOEEN 4
```

Definition at line [297](#) of file [MRCC_private.h](#).

7.114.2.70 RCC_AHB1ENR_GPIODEN

```
#define RCC_AHB1ENR_GPIODEN 3
```

Definition at line [298](#) of file [MRCC_private.h](#).

7.114.2.71 RCC_AHB1ENR_GPIOCEN

```
#define RCC_AHB1ENR_GPIOCEN 2
```

Definition at line [299](#) of file [MRCC_private.h](#).

7.114.2.72 RCC_AHB1ENR_GPIOBEN

```
#define RCC_AHB1ENR_GPIOBEN 1
```

Definition at line [300](#) of file [MRCC_private.h](#).

7.114.2.73 RCC_AHB1ENR_GPIOAEN

```
#define RCC_AHB1ENR_GPIOAEN 0
```

Definition at line [301](#) of file [MRCC_private.h](#).

7.114.2.74 RCC_AHB2ENR_OTGFSEN

```
#define RCC_AHB2ENR_OTGFSEN 7
```

Definition at line [304](#) of file [MRCC_private.h](#).

7.114.2.75 RCC_APB1ENR_PWREN

```
#define RCC_APB1ENR_PWREN 28
```

Definition at line 307 of file [MRCC_private.h](#).

7.114.2.76 RCC_APB1ENR_I2C3EN

```
#define RCC_APB1ENR_I2C3EN 23
```

Definition at line 308 of file [MRCC_private.h](#).

7.114.2.77 RCC_APB1ENR_I2C2EN

```
#define RCC_APB1ENR_I2C2EN 22
```

Definition at line 309 of file [MRCC_private.h](#).

7.114.2.78 RCC_APB1ENR_I2C1EN

```
#define RCC_APB1ENR_I2C1EN 21
```

Definition at line 310 of file [MRCC_private.h](#).

7.114.2.79 RCC_APB1ENR_USART2EN

```
#define RCC_APB1ENR_USART2EN 17
```

Definition at line 311 of file [MRCC_private.h](#).

7.114.2.80 RCC_APB1ENR_SPI3EN

```
#define RCC_APB1ENR_SPI3EN 15
```

Definition at line 312 of file [MRCC_private.h](#).

7.114.2.81 RCC_APB1ENR_SPI2EN

```
#define RCC_APB1ENR_SPI2EN 14
```

Definition at line 313 of file [MRCC_private.h](#).

7.114.2.82 RCC_APB1ENR_WWDGEN

```
#define RCC_APB1ENR_WWDGEN 11
```

Definition at line 314 of file [MRCC_private.h](#).

7.114.2.83 RCC_APB1ENR_TIM5EN

```
#define RCC_APB1ENR_TIM5EN 3
```

Definition at line 315 of file [MRCC_private.h](#).

7.114.2.84 RCC_APB1ENR_TIM4EN

```
#define RCC_APB1ENR_TIM4EN 2
```

Definition at line 316 of file [MRCC_private.h](#).

7.114.2.85 RCC_APB1ENR_TIM3EN

```
#define RCC_APB1ENR_TIM3EN 1
```

Definition at line 317 of file [MRCC_private.h](#).

7.114.2.86 RCC_APB1ENR_TIM2EN

```
#define RCC_APB1ENR_TIM2EN 0
```

Definition at line 318 of file [MRCC_private.h](#).

7.114.2.87 RCC_APB2ENR_TIM11EN

```
#define RCC_APB2ENR_TIM11EN 18
```

Definition at line 321 of file [MRCC_private.h](#).

7.114.2.88 RCC_APB2ENR_TIM10EN

```
#define RCC_APB2ENR_TIM10EN 17
```

Definition at line 322 of file [MRCC_private.h](#).

7.114.2.89 RCC_APB2ENR_TIM9EN

```
#define RCC_APB2ENR_TIM9EN 16
```

Definition at line 323 of file [MRCC_private.h](#).

7.114.2.90 RCC_APB2ENR_SYSCFGEN

```
#define RCC_APB2ENR_SYSCFGEN 14
```

Definition at line 324 of file [MRCC_private.h](#).

7.114.2.91 RCC_APB2ENR_SPI4EN

```
#define RCC_APB2ENR_SPI4EN 13
```

Definition at line 325 of file [MRCC_private.h](#).

7.114.2.92 RCC_APB2ENR_SPI1EN

```
#define RCC_APB2ENR_SPI1EN 12
```

Definition at line 326 of file [MRCC_private.h](#).

7.114.2.93 RCC_APB2ENR_SDIOEN

```
#define RCC_APB2ENR_SDIOEN 11
```

Definition at line [327](#) of file [MRCC_private.h](#).

7.114.2.94 RCC_APB2ENR_ADC1EN

```
#define RCC_APB2ENR_ADC1EN 8
```

Definition at line [328](#) of file [MRCC_private.h](#).

7.114.2.95 RCC_APB2ENR_USART6EN

```
#define RCC_APB2ENR_USART6EN 5
```

Definition at line [329](#) of file [MRCC_private.h](#).

7.114.2.96 RCC_APB2ENR_USART1EN

```
#define RCC_APB2ENR_USART1EN 4
```

Definition at line [330](#) of file [MRCC_private.h](#).

7.114.2.97 RCC_APB2ENR_TIM1EN

```
#define RCC_APB2ENR_TIM1EN 0
```

Definition at line [331](#) of file [MRCC_private.h](#).

7.114.2.98 RCC_AHB1LPENR_FLITFLPEN

```
#define RCC_AHB1LPENR_FLITFLPEN 15
```

Definition at line [333](#) of file [MRCC_private.h](#).

7.114.2.99 ENABLE

```
#define ENABLE 1
```

Definition at line 340 of file [MRCC_private.h](#).

7.114.2.100 DISABLE

```
#define DISABLE 2
```

Definition at line 341 of file [MRCC_private.h](#).

7.114.2.101 BYBASED

```
#define BYBASED 1
```

Definition at line 343 of file [MRCC_private.h](#).

7.114.2.102 NOTBYBASED

```
#define NOTBYBASED 2
```

Definition at line 344 of file [MRCC_private.h](#).

7.114.2.103 SYSCLK

```
#define SYSCLK 1
```

Definition at line 346 of file [MRCC_private.h](#).

7.114.2.104 PLLI2SCLK

```
#define PLLI2SCLK 5
```

Definition at line 347 of file [MRCC_private.h](#).

7.114.2.105 HSE

```
#define HSE 3
```

Definition at line [348](#) of file [MRCC_private.h](#).

7.114.2.106 PLLCLK [1/2]

```
#define PLLCLK 4
```

Definition at line [362](#) of file [MRCC_private.h](#).

7.114.2.107 NoDivision

```
#define NoDivision 1
```

Definition at line [351](#) of file [MRCC_private.h](#).

7.114.2.108 DivisionBy2

```
#define DivisionBy2 2
```

Definition at line [352](#) of file [MRCC_private.h](#).

7.114.2.109 DivisionBy3

```
#define DivisionBy3 3
```

Definition at line [353](#) of file [MRCC_private.h](#).

7.114.2.110 DivisionBy4

```
#define DivisionBy4 4
```

Definition at line [354](#) of file [MRCC_private.h](#).

7.114.2.111 DivisionBy5

```
#define DivisionBy5 5
```

Definition at line 355 of file [MRCC_private.h](#).

7.114.2.112 I2S_CKIN

```
#define I2S_CKIN 2
```

Definition at line 358 of file [MRCC_private.h](#).

7.114.2.113 HSI

```
#define HSI 1
```

Definition at line 360 of file [MRCC_private.h](#).

7.114.2.114 LSE

```
#define LSE 2
```

Definition at line 361 of file [MRCC_private.h](#).

7.114.2.115 PLLCLK [2/2]

```
#define PLLCLK 4
```

Definition at line 362 of file [MRCC_private.h](#).

7.114.2.116 SYSCLKby2

```
#define SYSCLKby2 2
```

Definition at line 365 of file [MRCC_private.h](#).

7.114.2.117 SYSCLKby4

```
#define SYSCLKby4 3
```

Definition at line 366 of file [MRCC_private.h](#).

7.114.2.118 SYSCLKby8

```
#define SYSCLKby8 4
```

Definition at line 367 of file [MRCC_private.h](#).

7.114.2.119 SYSCLKby16

```
#define SYSCLKby16 5
```

Definition at line 368 of file [MRCC_private.h](#).

7.114.2.120 SYSCLKby64

```
#define SYSCLKby64 6
```

Definition at line 369 of file [MRCC_private.h](#).

7.114.2.121 SYSCLKby128

```
#define SYSCLKby128 7
```

Definition at line 370 of file [MRCC_private.h](#).

7.114.2.122 SYSCLKby256

```
#define SYSCLKby256 8
```

Definition at line 371 of file [MRCC_private.h](#).

7.114.2.123 SYSCLKby512

```
#define SYSCLKby512 9
```

Definition at line 372 of file [MRCC_private.h](#).

7.114.2.124 NoCLK0

```
#define NoCLK0 1
```

Definition at line 375 of file [MRCC_private.h](#).

7.114.2.125 NoCLK1

```
#define NoCLK1 2
```

Definition at line 376 of file [MRCC_private.h](#).

7.114.2.126 HSEby2

```
#define HSEby2 3
```

Definition at line 377 of file [MRCC_private.h](#).

7.114.2.127 HSEby3

```
#define HSEby3 4
```

Definition at line 378 of file [MRCC_private.h](#).

7.114.2.128 HSEby4

```
#define HSEby4 5
```

Definition at line 379 of file [MRCC_private.h](#).

7.114.2.129 HSEby5

```
#define HSEby5 6
```

Definition at line 380 of file [MRCC_private.h](#).

7.114.2.130 HSEby6

```
#define HSEby6 7
```

Definition at line 381 of file [MRCC_private.h](#).

7.114.2.131 HSEby7

```
#define HSEby7 8
```

Definition at line 382 of file [MRCC_private.h](#).

7.114.2.132 HSEby8

```
#define HSEby8 9
```

Definition at line 383 of file [MRCC_private.h](#).

7.114.2.133 HSEby9

```
#define HSEby9 10
```

Definition at line 384 of file [MRCC_private.h](#).

7.114.2.134 HSEby10

```
#define HSEby10 11
```

Definition at line 385 of file [MRCC_private.h](#).

7.114.2.135 HSEby11

```
#define HSEby11 12
```

Definition at line 386 of file [MRCC_private.h](#).

7.114.2.136 HSEby12

```
#define HSEby12 13
```

Definition at line 387 of file [MRCC_private.h](#).

7.114.2.137 HSEby13

```
#define HSEby13 14
```

Definition at line 388 of file [MRCC_private.h](#).

7.114.2.138 HSEby14

```
#define HSEby14 15
```

Definition at line 389 of file [MRCC_private.h](#).

7.114.2.139 HSEby15

```
#define HSEby15 16
```

Definition at line 390 of file [MRCC_private.h](#).

7.114.2.140 HSEby16

```
#define HSEby16 17
```

Definition at line 391 of file [MRCC_private.h](#).

7.114.2.141 HSEby17

```
#define HSEby17 18
```

Definition at line [392](#) of file [MRCC_private.h](#).

7.114.2.142 HSEby18

```
#define HSEby18 19
```

Definition at line [393](#) of file [MRCC_private.h](#).

7.114.2.143 HSEby19

```
#define HSEby19 20
```

Definition at line [394](#) of file [MRCC_private.h](#).

7.114.2.144 HSEby20

```
#define HSEby20 21
```

Definition at line [395](#) of file [MRCC_private.h](#).

7.114.2.145 HSEby21

```
#define HSEby21 22
```

Definition at line [396](#) of file [MRCC_private.h](#).

7.114.2.146 HSEby22

```
#define HSEby22 23
```

Definition at line [397](#) of file [MRCC_private.h](#).

7.114.2.147 HSEby23

```
#define HSEby23 24
```

Definition at line 398 of file [MRCC_private.h](#).

7.114.2.148 HSEby24

```
#define HSEby24 25
```

Definition at line 399 of file [MRCC_private.h](#).

7.114.2.149 HSEby25

```
#define HSEby25 26
```

Definition at line 400 of file [MRCC_private.h](#).

7.114.2.150 HSEby26

```
#define HSEby26 27
```

Definition at line 401 of file [MRCC_private.h](#).

7.114.2.151 HSEby27

```
#define HSEby27 28
```

Definition at line 402 of file [MRCC_private.h](#).

7.114.2.152 HSEby28

```
#define HSEby28 39
```

Definition at line 403 of file [MRCC_private.h](#).

7.114.2.153 HSEby29

```
#define HSEby29 30
```

Definition at line [404](#) of file [MRCC_private.h](#).

7.114.2.154 HSEby30

```
#define HSEby30 31
```

Definition at line [405](#) of file [MRCC_private.h](#).

7.114.2.155 HSEby31

```
#define HSEby31 32
```

Definition at line [406](#) of file [MRCC_private.h](#).

7.114.2.156 AHBby2

```
#define AHBby2 1
```

Definition at line [409](#) of file [MRCC_private.h](#).

7.114.2.157 AHBby4

```
#define AHBby4 2
```

Definition at line [410](#) of file [MRCC_private.h](#).

7.114.2.158 AHBby8

```
#define AHBby8 3
```

Definition at line [411](#) of file [MRCC_private.h](#).

7.114.2.159 AHBby16

```
#define AHBby16 4
```

Definition at line 412 of file [MRCC_private.h](#).

7.114.2.160 Equal_0

```
#define Equal_0 0
```

Definition at line 415 of file [MRCC_private.h](#).

7.114.2.161 Equal_1

```
#define Equal_1 1
```

Definition at line 416 of file [MRCC_private.h](#).

7.114.2.162 Equal_2

```
#define Equal_2 2
```

Definition at line 417 of file [MRCC_private.h](#).

7.114.2.163 Equal_3

```
#define Equal_3 3
```

Definition at line 418 of file [MRCC_private.h](#).

7.114.2.164 Equal_4

```
#define Equal_4 4
```

Definition at line 419 of file [MRCC_private.h](#).

7.114.2.165 Equal_5

```
#define Equal_5 5
```

Definition at line [420](#) of file [MRCC_private.h](#).

7.114.2.166 Equal_6

```
#define Equal_6 6
```

Definition at line [421](#) of file [MRCC_private.h](#).

7.114.2.167 Equal_7

```
#define Equal_7 7
```

Definition at line [422](#) of file [MRCC_private.h](#).

7.114.2.168 Equal_8

```
#define Equal_8 8
```

Definition at line [423](#) of file [MRCC_private.h](#).

7.114.2.169 Equal_9

```
#define Equal_9 9
```

Definition at line [424](#) of file [MRCC_private.h](#).

7.114.2.170 Equal_10

```
#define Equal_10 10
```

Definition at line [425](#) of file [MRCC_private.h](#).

7.114.2.171 Equal_11

```
#define Equal_11 11
```

Definition at line [426](#) of file [MRCC_private.h](#).

7.114.2.172 Equal_12

```
#define Equal_12 12
```

Definition at line [427](#) of file [MRCC_private.h](#).

7.114.2.173 Equal_13

```
#define Equal_13 13
```

Definition at line [428](#) of file [MRCC_private.h](#).

7.114.2.174 Equal_14

```
#define Equal_14 14
```

Definition at line [429](#) of file [MRCC_private.h](#).

7.114.2.175 Equal_15

```
#define Equal_15 15
```

Definition at line [430](#) of file [MRCC_private.h](#).

7.114.2.176 Equal_16

```
#define Equal_16 16
```

Definition at line [431](#) of file [MRCC_private.h](#).

7.114.2.177 Equal_17

```
#define Equal_17 17
```

Definition at line [432](#) of file [MRCC_private.h](#).

7.114.2.178 Equal_18

```
#define Equal_18 18
```

Definition at line [433](#) of file [MRCC_private.h](#).

7.114.2.179 Equal_19

```
#define Equal_19 19
```

Definition at line [434](#) of file [MRCC_private.h](#).

7.114.2.180 Equal_20

```
#define Equal_20 20
```

Definition at line [435](#) of file [MRCC_private.h](#).

7.114.2.181 Equal_21

```
#define Equal_21 21
```

Definition at line [436](#) of file [MRCC_private.h](#).

7.114.2.182 Equal_22

```
#define Equal_22 22
```

Definition at line [437](#) of file [MRCC_private.h](#).

7.114.2.183 Equal_23

```
#define Equal_23 23
```

Definition at line [438](#) of file [MRCC_private.h](#).

7.114.2.184 Equal_24

```
#define Equal_24 24
```

Definition at line [439](#) of file [MRCC_private.h](#).

7.114.2.185 Equal_25

```
#define Equal_25 25
```

Definition at line [440](#) of file [MRCC_private.h](#).

7.114.2.186 Equal_26

```
#define Equal_26 26
```

Definition at line [441](#) of file [MRCC_private.h](#).

7.114.2.187 Equal_27

```
#define Equal_27 27
```

Definition at line [442](#) of file [MRCC_private.h](#).

7.114.2.188 Equal_28

```
#define Equal_28 28
```

Definition at line [443](#) of file [MRCC_private.h](#).

7.114.2.189 Equal_29

```
#define Equal_29 29
```

Definition at line [444](#) of file [MRCC_private.h](#).

7.114.2.190 Equal_30

```
#define Equal_30 30
```

Definition at line [445](#) of file [MRCC_private.h](#).

7.114.2.191 Equal_31

```
#define Equal_31 31
```

Definition at line [446](#) of file [MRCC_private.h](#).

7.114.2.192 Equal_32

```
#define Equal_32 32
```

Definition at line [447](#) of file [MRCC_private.h](#).

7.114.2.193 Equal_33

```
#define Equal_33 33
```

Definition at line [448](#) of file [MRCC_private.h](#).

7.114.2.194 Equal_34

```
#define Equal_34 34
```

Definition at line [449](#) of file [MRCC_private.h](#).

7.114.2.195 Equal_35

```
#define Equal_35 35
```

Definition at line [450](#) of file [MRCC_private.h](#).

7.114.2.196 Equal_36

```
#define Equal_36 36
```

Definition at line [451](#) of file [MRCC_private.h](#).

7.114.2.197 Equal_37

```
#define Equal_37 37
```

Definition at line [452](#) of file [MRCC_private.h](#).

7.114.2.198 Equal_38

```
#define Equal_38 38
```

Definition at line [453](#) of file [MRCC_private.h](#).

7.114.2.199 Equal_39

```
#define Equal_39 39
```

Definition at line [454](#) of file [MRCC_private.h](#).

7.114.2.200 Equal_40

```
#define Equal_40 40
```

Definition at line [455](#) of file [MRCC_private.h](#).

7.114.2.201 Equal_41

```
#define Equal_41 41
```

Definition at line [456](#) of file [MRCC_private.h](#).

7.114.2.202 Equal_42

```
#define Equal_42 42
```

Definition at line [457](#) of file [MRCC_private.h](#).

7.114.2.203 Equal_43

```
#define Equal_43 43
```

Definition at line [458](#) of file [MRCC_private.h](#).

7.114.2.204 Equal_44

```
#define Equal_44 44
```

Definition at line [459](#) of file [MRCC_private.h](#).

7.114.2.205 Equal_45

```
#define Equal_45 45
```

Definition at line [460](#) of file [MRCC_private.h](#).

7.114.2.206 Equal_46

```
#define Equal_46 46
```

Definition at line [461](#) of file [MRCC_private.h](#).

7.114.2.207 Equal_47

```
#define Equal_47 47
```

Definition at line [462](#) of file [MRCC_private.h](#).

7.114.2.208 Equal_48

```
#define Equal_48 48
```

Definition at line [463](#) of file [MRCC_private.h](#).

7.114.2.209 Equal_49

```
#define Equal_49 49
```

Definition at line [464](#) of file [MRCC_private.h](#).

7.114.2.210 Equal_50

```
#define Equal_50 50
```

Definition at line [465](#) of file [MRCC_private.h](#).

7.114.2.211 Equal_51

```
#define Equal_51 51
```

Definition at line [466](#) of file [MRCC_private.h](#).

7.114.2.212 Equal_52

```
#define Equal_52 52
```

Definition at line [467](#) of file [MRCC_private.h](#).

7.114.2.213 Equal_53

```
#define Equal_53 53
```

Definition at line [468](#) of file [MRCC_private.h](#).

7.114.2.214 Equal_54

```
#define Equal_54 54
```

Definition at line [469](#) of file [MRCC_private.h](#).

7.114.2.215 Equal_55

```
#define Equal_55 55
```

Definition at line [470](#) of file [MRCC_private.h](#).

7.114.2.216 Equal_56

```
#define Equal_56 56
```

Definition at line [471](#) of file [MRCC_private.h](#).

7.114.2.217 Equal_57

```
#define Equal_57 57
```

Definition at line [472](#) of file [MRCC_private.h](#).

7.114.2.218 Equal_58

```
#define Equal_58 58
```

Definition at line [473](#) of file [MRCC_private.h](#).

7.114.2.219 Equal_59

```
#define Equal_59 59
```

Definition at line [474](#) of file [MRCC_private.h](#).

7.114.2.220 Equal_60

```
#define Equal_60 60
```

Definition at line [475](#) of file [MRCC_private.h](#).

7.114.2.221 Equal_61

```
#define Equal_61 61
```

Definition at line [476](#) of file [MRCC_private.h](#).

7.114.2.222 Equal_62

```
#define Equal_62 62
```

Definition at line [477](#) of file [MRCC_private.h](#).

7.114.2.223 Equal_63

```
#define Equal_63 63
```

Definition at line [478](#) of file [MRCC_private.h](#).

7.115 MRCC_private.h

[Go to the documentation of this file.](#)

```
00001 /* Header file guard */
00009 #ifndef MCAL_RCC_MRCC_PRIVATE_H_
00010 #define MCAL_RCC_MRCC_PRIVATE_H_
00012
00013
00014
00015 /***** Peripherals declaration *****/
00016 /* Peripherals declaration */
00017 /***** Peripherals declaration *****/
00018
00024 typedef struct
00025 {
00026
00030     u32_t CR          ;
00031
00035     u32_t PLLCFGR    ;
00036
00040     u32_t CFGR        ;
00041
00045     u32_t CIR         ;
00046
00050     u32_t AHB1RSTR   ;
00051
00055     u32_t AHB2RSTR   ;
00056
00060     u32_t Reserved1  ;
00061
00065     u32_t Reserved2  ;
00066
00070     u32_t APB1RSTR   ;
00071
00075     u32_t APB2RSTR   ;
00076
00080     u32_t Reserved3  ;
00081
00085     u32_t Reserved4  ;
00086
00090     u32_t AHB1ENR    ;
00091
00095     u32_t AHB2ENR    ;
00096
00100     u32_t Reserved5  ;
00101
00105     u32_t Reserved6  ;
00106
00110     u32_t APB1ENR    ;
00111
00115     u32_t APB2ENR    ;
00116
00120     u32_t Reserved7  ;
00121
00125     u32_t Reserved8  ;
00126
00130     u32_t AHB1LPENR  ;
00131
00135     u32_t AHB2LPENR  ;
00136
00140     u32_t Reserved9  ;
00141
00145     u32_t Reserved10 ;
00146
00150     u32_t APB1LPENR  ;
00151
00155     u32_t APB2LPENR  ;
00156
00160     u32_t Reserved11 ;
00161
00165     u32_t Reserved12 ;
00166
00170     u32_t BDCR        ;
00171
00175     u32_t CSR         ;
00176
00180     u32_t Reserved13  ;
00181
00185     u32_t Reserved14  ;
00186
00190     u32_t SSCGR       ;
00191
00195     u32_t PLLI2SCFGR  ;
00196
```

```

00200     u32_t DCKCFGR      ;
00201
00202
00203 } RCC_MemoryMapType ;
00204
00205
00211 #define RCC_BASE_ADDRESS 0x40023800
00212
00218 #define RCC ( (volatile P2VAR(RCC_MemoryMapType) ) (RCC_BASE_ADDRESS) ) // Is a pointer to the struct.
00219
00220
00221 /***** Peripheral registers *****/
00222 /* ***** Peripheral registers ***** */
00223 /***** Peripheral registers *****/
00224
00225 #define RCC_CR_PLLI2SRDY 27
00226 #define RCC_CR_PLLI2SON 26
00227 #define RCC_CR_PLLRDY 25
00228 #define RCC_CR_PLLON 24
00229 #define RCC_CR_CSSON 19
00230 #define RCC_CR_HSEBYP 18
00231 #define RCC_CR_HSERYDY 17
00232 #define RCC_CR_HSEON 16
00233 #define RCC_CR_HSIRDY 1
00234 #define RCC_CR_HSION 0
00235
00236
00237 #define RCC_PLLCFGRL_PLLQ_b0 24
00238 #define RCC_PLLCFGRL_PLLQ_b1 25
00239 #define RCC_PLLCFGRL_PLLQ_b2 26
00240 #define RCC_PLLCFGRL_PLLQ_b3 27
00241 #define RCC_PLLCFGRL_PLLSRC 22
00242 #define RCC_PLLCFGRL_PLLP_b0 16
00243 #define RCC_PLLCFGRL_PLLP_b1 17
00244 #define RCC_PLLCFGRL_PLLN_b0 6
00245 #define RCC_PLLCFGRL_PLLN_b1 7
00246 #define RCC_PLLCFGRL_PLLN_b2 8
00247 #define RCC_PLLCFGRL_PLLN_b3 9
00248 #define RCC_PLLCFGRL_PLLN_b4 10
00249 #define RCC_PLLCFGRL_PLLN_b5 11
00250 #define RCC_PLLCFGRL_PLLN_b6 12
00251 #define RCC_PLLCFGRL_PLLN_b7 13
00252 #define RCC_PLLCFGRL_PLLN_b8 14
00253 #define RCC_PLLCFGRL_PLLM_b0 0
00254 #define RCC_PLLCFGRL_PLLM_b1 1
00255 #define RCC_PLLCFGRL_PLLM_b2 2
00256 #define RCC_PLLCFGRL_PLLM_b3 3
00257 #define RCC_PLLCFGRL_PLLM_b4 4
00258 #define RCC_PLLCFGRL_PLLM_b5 5
00259
00260
00261 #define RCC_CFGR_MOC2_b0 30
00262 #define RCC_CFGR_MOC2_b1 31
00263 #define RCC_CFGR_MOC2PRE_b0 27
00264 #define RCC_CFGR_MOC2PRE_b1 28
00265 #define RCC_CFGR_MOC2PRE_b2 29
00266 #define RCC_CFGR_MOC1PRE_b0 24
00267 #define RCC_CFGR_MOC1PRE_b1 25
00268 #define RCC_CFGR_MOC1PRE_b2 26
00269 #define RCC_CFGR_I2SSRC 23
00270 #define RCC_CFGR_MOC1_b0 21
00271 #define RCC_CFGR_MOC1_b1 22
00272 #define RCC_CFGR_RTCPRE_b0 16
00273 #define RCC_CFGR_RTCPRE_b1 17
00274 #define RCC_CFGR_RTCPRE_b2 18
00275 #define RCC_CFGR_RTCPRE_b3 19
00276 #define RCC_CFGR_RTCPRE_b4 20
00277 #define RCC_CFGR_PPREG2_b0 13
00278 #define RCC_CFGR_PPREG2_b1 14
00279 #define RCC_CFGR_PPREG2_b2 15
00280 #define RCC_CFGR_PPREG1_b0 10
00281 #define RCC_CFGR_PPREG1_b1 11
00282 #define RCC_CFGR_PPREG1_b2 12
00283 #define RCC_CFGR_HPRE_b0 4
00284 #define RCC_CFGR_HPRE_b1 5
00285 #define RCC_CFGR_HPRE_b2 6
00286 #define RCC_CFGR_HPRE_b3 7
00287 #define RCC_CFGR_SWS_b0 2
00288 #define RCC_CFGR_SWS_b1 3
00289 #define RCC_CFGR_SW_b0 0
00290 #define RCC_CFGR_SW_b1 1
00291
00292
00293 #define RCC_AHB1ENR_DMA2EN 22
00294 #define RCC_AHB1ENR_DMA1EN 21
00295 #define RCC_AHB1ENR_CRCEN 12
00296 #define RCC_AHB1ENR_GPIOHEN 7

```

```
00297 #define RCC_AHB1ENR_GPIOEEN      4
00298 #define RCC_AHB1ENR_GPIODEN      3
00299 #define RCC_AHB1ENR_GPIOCEN      2
00300 #define RCC_AHB1ENR_GPIOBEN      1
00301 #define RCC_AHB1ENR_GPIOAEN      0
00302
00303
00304 #define RCC_AHB2ENR_OTGFSEN      7
00305
00306
00307 #define RCC_APB1ENR_PWREN        28
00308 #define RCC_APB1ENR_I2C3EN       23
00309 #define RCC_APB1ENR_I2C2EN       22
00310 #define RCC_APB1ENR_I2C1EN       21
00311 #define RCC_APB1ENR_USART2EN     17
00312 #define RCC_APB1ENR_SPI3EN       15
00313 #define RCC_APB1ENR_SPI2EN       14
00314 #define RCC_APB1ENR_WWDGEN       11
00315 #define RCC_APB1ENR_TIM5EN       3
00316 #define RCC_APB1ENR_TIM4EN       2
00317 #define RCC_APB1ENR_TIM3EN       1
00318 #define RCC_APB1ENR_TIM2EN       0
00319
00320
00321 #define RCC_APB2ENR_TIM11EN      18
00322 #define RCC_APB2ENR_TIM10EN      17
00323 #define RCC_APB2ENR_TIM9EN       16
00324 #define RCC_APB2ENR_SYSCFGEN     14
00325 #define RCC_APB2ENR_SPI4EN       13
00326 #define RCC_APB2ENR_SPI1EN       12
00327 #define RCC_APB2ENR_SDIOEN       11
00328 #define RCC_APB2ENR_ADC1EN       8
00329 #define RCC_APB2ENR_USART6EN     5
00330 #define RCC_APB2ENR_USART1EN     4
00331 #define RCC_APB2ENR_TIM1EN       0
00332
00333 #define RCC_AHB1LPENR_FLITFLPEN   15
00334
00335
00336 /****** Config.h Macros *****/
00337 /****** Config.h Macros *****/
00338 /****** Config.h Macros *****/
00339
00340 #define ENABLE    1
00341 #define DISABLE   2
00342
00343 #define BYBASED   1
00344 #define NOTBYBASED 2
00345
00346 #define SYSCLK     1
00347 #define PLLI2SCLK  5
00348 #define HSE        3
00349 #define PLLCLK     4
00350
00351 #define NoDivision 1
00352 #define DivisionBy2 2
00353 #define DivisionBy3 3
00354 #define DivisionBy4 4
00355 #define DivisionBy5 5
00356
00357
00358 #define I2S_CKIN   2
00359
00360 #define HSI        1
00361 #define LSE        2
00362 #define PLLCLK    4
00363
00364 // AHB prescalers:
00365 #define SYSCLKby2  2  // (0b1000)
00366 #define SYSCLKby4  3  // (0b1001)
00367 #define SYSCLKby8  4  // (0b1010)
00368 #define SYSCLKby16 5  // (0b1011)
00369 #define SYSCLKby64 6  // (0b1100)
00370 #define SYSCLKby128 7 // (0b1101)
00371 #define SYSCLKby256 8 // (0b1110)
00372 #define SYSCLKby512 9 // (0b1111)
00373
00374
00375 #define NoCLK0    1
00376 #define NoCLK1    2
00377 #define HSEby2    3
00378 #define HSEby3    4
00379 #define HSEby4    5
00380 #define HSEby5    6
00381 #define HSEby6    7
00382 #define HSEby7    8
00383 #define HSEby8    9
```

```
00384 #define HSEby9 10
00385 #define HSEby10 11
00386 #define HSEby11 12
00387 #define HSEby12 13
00388 #define HSEby13 14
00389 #define HSEby14 15
00390 #define HSEby15 16
00391 #define HSEby16 17
00392 #define HSEby17 18
00393 #define HSEby18 19
00394 #define HSEby19 20
00395 #define HSEby20 21
00396 #define HSEby21 22
00397 #define HSEby22 23
00398 #define HSEby23 24
00399 #define HSEby24 25
00400 #define HSEby25 26
00401 #define HSEby26 27
00402 #define HSEby27 28
00403 #define HSEby28 39
00404 #define HSEby29 30
00405 #define HSEby30 31
00406 #define HSEby31 32
00407
00408
00409 #define AHBby2 1
00410 #define AHBby4 2
00411 #define AHBby8 3
00412 #define AHBby16 4
00413
00414
00415 #define Equal_0 0
00416 #define Equal_1 1
00417 #define Equal_2 2
00418 #define Equal_3 3
00419 #define Equal_4 4
00420 #define Equal_5 5
00421 #define Equal_6 6
00422 #define Equal_7 7
00423 #define Equal_8 8
00424 #define Equal_9 9
00425 #define Equal_10 10
00426 #define Equal_11 11
00427 #define Equal_12 12
00428 #define Equal_13 13
00429 #define Equal_14 14
00430 #define Equal_15 15
00431 #define Equal_16 16
00432 #define Equal_17 17
00433 #define Equal_18 18
00434 #define Equal_19 19
00435 #define Equal_20 20
00436 #define Equal_21 21
00437 #define Equal_22 22
00438 #define Equal_23 23
00439 #define Equal_24 24
00440 #define Equal_25 25
00441 #define Equal_26 26
00442 #define Equal_27 27
00443 #define Equal_28 28
00444 #define Equal_29 29
00445 #define Equal_30 30
00446 #define Equal_31 31
00447 #define Equal_32 32
00448 #define Equal_33 33
00449 #define Equal_34 34
00450 #define Equal_35 35
00451 #define Equal_36 36
00452 #define Equal_37 37
00453 #define Equal_38 38
00454 #define Equal_39 39
00455 #define Equal_40 40
00456 #define Equal_41 41
00457 #define Equal_42 42
00458 #define Equal_43 43
00459 #define Equal_44 44
00460 #define Equal_45 45
00461 #define Equal_46 46
00462 #define Equal_47 47
00463 #define Equal_48 48
00464 #define Equal_49 49
00465 #define Equal_50 50
00466 #define Equal_51 51
00467 #define Equal_52 52
00468 #define Equal_53 53
00469 #define Equal_54 54
00470 #define Equal_55 55
```

```

00471 #define Equal_56      56
00472 #define Equal_57      57
00473 #define Equal_58      58
00474 #define Equal_59      59
00475 #define Equal_60      60
00476 #define Equal_61      61
00477 #define Equal_62      62
00478 #define Equal_63      63
00479
00480
00481
00482
00483
00484 #endif /* MCAL_RCC_MRCC_PRIVATE_H */

```

7.116 COTS/MCAL/RCC/MRCC_program.c File Reference

This file contains the source code of the interfacing for the RCC module.

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "MRCC_interface.h"
#include "MRCC_private.h"
#include "MRCC_config.h"

```

Functions

- void [MRCC_vInit](#) (void)
Initialize the RCC with a certain configurations.
- void [MRCC_vEnablePeriphralCLK](#) ([u32_t](#) A_u32BusID, [u32_t](#) A_u32PeriphralID)
Enabling the CLK on a specific Peripheral.
- void [MRCC_vDisablePeriphralCLK](#) ([u32_t](#) A_u32BusID, [u32_t](#) A_u32PeriphralID)
Disabling the CLK on a specific Peripheral.

7.116.1 Detailed Description

This file contains the source code of the interfacing for the RCC module.

Author

Ali El Bana & Mo Alaa

Version

2.0

Date

11/9/2022

Definition in file [MRCC_program.c](#).

7.116.2 Function Documentation

7.116.2.1 MRCC_vInit()

```
void MRCC_vInit (
    void )
```

Initialize the RCC with a certain configurations.

Definition at line 29 of file [MRCC_program.c](#).

```
00030 {
00031     // PLLI2S (ON/OFF).
00032 #if PLLI2S == ENABLE
00033     SET_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00036
00037 #elif PLLI2S == DISABLE
00038     CLR_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00040
00041 #endif
00042
00043
00044     // PLL (ON/OFF).
00045 #if PLL == ENABLE
00046     SET_BIT( RCC->CR, RCC_CR_PLLON ) ;
00048
00049 #elif PLL == DISABLE
00050     CLR_BIT( RCC->CR, RCC_CR_PLLON ) ;
00052
00053 #endif
00054
00055
00056     // CSS (ON/OFF).
00057 #if CSS == ENABLE
00058     SET_BIT( RCC->CR, RCC_CR_CSSON ) ;
00060
00061 #elif CSS == DISABLE
00062     CLR_BIT( RCC->CR, RCC_CR_CSSON ) ;
00064
00065 #endif
00066
00067
00068     // HSEBYP.
00069 #if HSEBYP == BYBASED
00070     SET_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00072
00073 #elif HSEBYP == NOTBYBASED
00074     CLR_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00076
00077 #endif
00078
00079
00080     // Select CLK switch (HSI/HSE/PLL).
00081 #if SW == HSI
00082
00083     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00084     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00085
00086 #elif SW == HSE
00087
00088     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00089     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00090
00091 #elif SW == PLLCLK
00092
00093     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00094     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00095
```

```

00096 #endif
00097
00098     // Select CLK switch status (HSI/HSE/PLL).
00099 #if SWS == HSI
00100         CLR_BIT( RCC->CFGGR, RCC_CFGR_SWS_b0 ) ;
00101         CLR_BIT( RCC->CFGGR, RCC_CFGR_SWS_b1 ) ;
00103
00104 #elif SWS == HSE
00105
00106         SET_BIT( RCC->CFGGR, RCC_CFGR_SWS_b0 ) ;
00107         CLR_BIT( RCC->CFGGR, RCC_CFGR_SWS_b1 ) ;
00108
00109 #elif SWS == PLLCLK
00110
00111         CLR_BIT( RCC->CFGGR, RCC_CFGR_SWS_b0 ) ;
00112         SET_BIT( RCC->CFGGR, RCC_CFGR_SWS_b1 ) ;
00113
00114 #endif
00115
00116
00117     // MCO2 selection:
00118 #if MCO2 == SYSCLK
00119
00120         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b0 ) ;
00121         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b1 ) ;
00122
00123 #elif MCO2 == PLLI2SCLK
00124
00125         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b0 ) ;
00126         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b1 ) ;
00127
00128 #elif MCO2 == HSE
00129
00130         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b0 ) ;
00131         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b1 ) ;
00132
00133 #elif MCO2 == PLLCLK
00134
00135         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b0 ) ;
00136         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2_b1 ) ;
00137
00138 #endif
00139
00140
00141     // MCO2 prescaler:
00142 #if MCO2PRE == NoDivision
00143
00144         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b0 ) ;
00145         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b1 ) ;
00146         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b2 ) ;
00147
00148 #elif MCO2PRE == DivisionBy2
00149
00150         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b0 ) ;
00151         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b1 ) ;
00152         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b2 ) ;
00153
00154 #elif MCO2PRE == DivisionBy3
00155
00156         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b0 ) ;
00157         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b1 ) ;
00158         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b2 ) ;
00159
00160 #elif MCO2PRE == DivisionBy4
00161
00162         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b0 ) ;
00163         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b1 ) ;
00164         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b2 ) ;
00165
00166 #elif MCO2PRE == DivisionBy5
00167
00168         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b0 ) ;
00169         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b1 ) ;
00170         SET_BIT( RCC->CFGGR, RCC_CFGR_MOC2PRE_b2 ) ;
00171
00172 #endif
00173
00174
00175     // MCO1 selection:
00176 #if MCO1 == HSI
00177
00178         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC1_b0 ) ;
00179         CLR_BIT( RCC->CFGGR, RCC_CFGR_MOC1_b1 ) ;
00180
00181 #elif MCO1 == LSE
00182

```

```

00183     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00184     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00185
00186 #elif MCO1 == HSE
00187
00188     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00189     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00190
00191 #elif MCO1 == PLLCLK
00192
00193     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00194     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00195
00196 #endif
00197
00198
00199 // MCO1 prescaler:
00200 #if MCO1PRE == NoDivision
00201
00202     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00203     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00204     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00205
00206 #elif MCO1PRE == DivisionBy2
00207
00208     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00209     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00210     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00211
00212 #elif MCO1PRE == DivisionBy3
00213
00214     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00215     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00216     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00217
00218 #elif MCO1PRE == DivisionBy4
00219
00220     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00221     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00222     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00223
00224 #elif MCO1PRE == DivisionBy5
00225
00226     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00227     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00228     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00229
00230 #endif
00231
00232
00233 // AHB prescalers:
00234 #if HPRE == NoDivision
00235
00236     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00237     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00238     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00239     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00240
00241 #elif HPRE == SYSCLKby2
00242
00243     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00244     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00245     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00246     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00247
00248 #elif HPRE == SYSCLKby4
00249
00250     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00251     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00252     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00253     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00254
00255 #elif HPRE == SYSCLKby8
00256
00257     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00258     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00259     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00260     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00261
00262 #elif HPRE == SYSCLKby16
00263
00264     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00265     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00266     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00267     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00268
00269 #elif HPRE == SYSCLKby64

```

```

00270
00271     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00272     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00273     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00274     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00275
00276 #elif HPRE == SYSCLKby128
00277
00278     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00279     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00280     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00281     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00282
00283 #elif HPRE == SYSCLKby256
00284
00285     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00286     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00287     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00288     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00289
00290 #elif HPRE == SYSCLKby512
00291
00292     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00293     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00294     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00295     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00296
00297 #endif
00298
00299
00300 // APB1 prescalers:
00301 #if PPREG == NoDivision
00302
00303     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00304     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00305     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00306
00307 #elif PPREG == AHBby2
00308
00309     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00310     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00311     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00312
00313 #elif PPREG == AHBby4
00314
00315     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00316     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00317     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00318
00319 #elif PPREG == AHBby8
00320
00321     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00322     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00323     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00324
00325 #elif PPREG == AHBby16
00326
00327     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00328     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00329     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00330
00331 #endif
00332
00333 // APB2 prescalers:
00334 #if PPREG == NoDivision
00335
00336     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00337     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00338     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00339
00340 #elif PPREG == AHBby2
00341
00342     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00343     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00344     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00345
00346 #elif PPREG == AHBby4
00347
00348     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00349     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00350     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00351
00352 #elif PPREG == AHBby8
00353
00354     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00355     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00356     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;

```

```

00357
00358 #elif PPRE2 == AHBby16
00359
00360     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00361     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00362     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00363
00364 #endif
00365
00366
00367 // PLL configurations:
00368
00369
00370
00371     // Enable the selected CLK (HSI ON/HSE ON/PLL ON): //
00372     // HSE (ON/OFF).
00373 #if HSE_EN == ENABLE
00374
00375     SET_BIT( RCC->CR, RCC_CR_HSEON ) ;
00376
00377 #elif HSE_EN == DISABLE
00378
00379     CLR_BIT( RCC->CR, RCC_CR_HSEON ) ;
00380
00381 #endif
00382
00383
00384     // HSI (ON/OFF).
00385 #if HSI_EN == ENABLE
00386
00387     SET_BIT( RCC->CR, RCC_CR_HSICON ) ;
00388
00389 #elif HSI_EN == DISABLE
00390
00391     CLR_BIT( RCC->CR, RCC_CR_HSICON ) ;
00392
00393 #endif
00394
00395 }

```

References [CLR_BIT](#), [RCC](#), and [SET_BIT](#).

7.116.2.2 MRCC_vEnablePeriphralCLK()

```

void MRCC_vEnablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )

```

Enabling the CLK on a specific Peripheral.

Parameters

in	<i>A_u32BusID</i>	Bus ID.
in	<i>A_u32PeriphralID</i>	Peripheral ID.

Definition at line 400 of file [MRCC_program.c](#).

```

00401 {
00402
00403     switch( A_u32BusID )
00404     {
00405
00406         case RCC_AHB1 :
00407
00408             SET_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00409
00410             break ;
00411
00412
00413         case RCC_AHB2 :
00414

```

```

00415     SET_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00416
00417     break ;
00418
00419
00420     case RCC_APB1 :
00421
00422         SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00423
00424     break ;
00425
00426
00427     case RCC_APB2 :
00428
00429         SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00430
00431     break ;
00432
00433     case RCC_AHB1LPENR:
00434
00435         SET_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00436
00437     break ;
00438
00439     default:
00440
00441         // Error wrong Bus ID
00442
00443     break ;
00444
00445 }
00446
00447 }
```

References [RCC](#), [RCC_AHB1](#), [RCC_AHB1LPENR](#), [RCC_AHB2](#), [RCC_APB1](#), [RCC_APB2](#), and [SET_BIT](#).

Referenced by [MADC_vEnable\(\)](#), [MADC_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.116.2.3 MRCC_vDisablePeriphralCLK()

```
void MRCC_vDisablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )
```

Disabling the CLK on a specific Peripheral.

Parameters

in	<i>A_u32BusID</i>	Bus ID.
in	<i>A_u32PeriphralID</i>	Peripheral ID.

Definition at line 453 of file [MRCC_program.c](#).

```

00454 {
00455
00456     switch( A_u32BusID )
00457     {
00458
00459         case RCC_AHB1 :
00460
00461             CLR_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00462
00463         break ;
00464
00465
00466         case RCC_AHB2 :
00467
00468             CLR_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00469 }
```

```

00470
00471     break ;
00472
00473
00474     case RCC_APB1 :
00475
00476         CLR_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00477
00478     break ;
00479
00480
00481     case RCC_APB2 :
00482
00483         CLR_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00484
00485     break ;
00486
00487
00488     case RCC_AHB1LPENR:
00489
00490         CLR_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00491
00492     break ;
00493
00494
00495     default:
00496
00497     // Error wrong Bus ID
00498
00499     break ;
00500
00501 }
00502
00503 }
```

References [CLR_BIT](#), [RCC](#), [RCC_AHB1](#), [RCC_AHB1LPENR](#), [RCC_AHB2](#), [RCC_APB1](#), and [RCC_APB2](#).

Referenced by [MADC_vDisable\(\)](#).

7.117 MRCC_program.c

[Go to the documentation of this file.](#)

```

00001 /*****
00012 /*           Include headers           */
00013 *****/
00014 #include "../../../LIB/LSTD_TYPES.h"
00015 #include "../../../LIB/LSTD_COMPILER.h"
00016 #include "../../../LIB/LSTD_VALUES.h"
00017 #include "../../../LIB/LSTD_BITMATH.h"
00018
00019 #include "MRCC_interface.h"
00020 #include "MRCC_private.h"
00021 #include "MRCC_config.h"
00022
00023
00024
00025 /*****
00026 /*           Functions implementations      */
00027 *****/
00028
00029 void MRCC_vInit( void )
00030 {
00031
00032     // PLLI2S (ON/OFF).
00033 #if PLLI2S == ENABLE
00034
00035     SET_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00036
00037 #elif PLLI2S == DISABLE
00038
00039     CLR_BIT( RCC->CR, RCC_CR_PLLI2SON ) ;
00040
00041 #endif
00042
00043
00044     // PLL (ON/OFF).
00045 #if PLL == ENABLE
00046
```

```

00047     SET_BIT( RCC->CR, RCC_CR_PLLON ) ;
00048
00049 #elif PLL == DISABLE
00050
00051     CLR_BIT( RCC->CR, RCC_CR_PLLON ) ;
00052
00053 #endif
00054
00055
00056 // CSS (ON/OFF).
00057 #if CSS == ENABLE
00058
00059     SET_BIT( RCC->CR, RCC_CR_CSSON ) ;
00060
00061 #elif CSS == DISABLE
00062
00063     CLR_BIT( RCC->CR, RCC_CR_CSSON ) ;
00064
00065 #endif
00066
00067
00068 // HSEBYP.
00069 #if HSEBYP == BYBASED
00070
00071     SET_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00072
00073 #elif HSEBYP == NOTBYBASED
00074
00075     CLR_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00076
00077 #endif
00078
00079
00080 // Select CLK switch (HSI/HSE/PLL).
00081 #if SW == HSI
00082
00083     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00084     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00085
00086 #elif SW == HSE
00087
00088     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00089     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00090
00091 #elif SW == PLLCLK
00092
00093     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00094     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00095
00096 #endif
00097
00098 // Select CLK switch status (HSI/HSE/PLL).
00099 #if SWS == HSI
00100
00101     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00102     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00103
00104 #elif SWS == HSE
00105
00106     SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00107     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00108
00109 #elif SWS == PLLCLK
00110
00111     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00112     SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00113
00114 #endif
00115
00116
00117 // MCO2 selection:
00118 #if MCO2 == SYSCLK
00119
00120     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00121     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00122
00123 #elif MCO2 == PLLI2SCLK
00124
00125     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00126     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00127
00128 #elif MCO2 == HSE
00129
00130     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00131     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00132
00133 #elif MCO2 == PLLCLK

```

```

00134     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00135     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00136
00137 #endif
00138
00139
00140
00141 // MCO2 prescaler:
00142 #if MCO2PRE == NoDivision
00143
00144     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00145     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00146     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00147
00148 #elif MCO2PRE == DivisionBy2
00149
00150     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00151     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00152     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00153
00154 #elif MCO2PRE == DivisionBy3
00155
00156     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00157     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00158     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00159
00160 #elif MCO2PRE == DivisionBy4
00161
00162     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00163     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00164     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00165
00166 #elif MCO2PRE == DivisionBy5
00167
00168     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00169     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00170     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00171
00172#endif
00173
00174
00175 // MCO1 selection:
00176 #if MCO1 == HSI
00177
00178     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00179     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00180
00181 #elif MCO1 == LSE
00182
00183     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00184     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00185
00186 #elif MCO1 == HSE
00187
00188     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00189     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00190
00191 #elif MCO1 == PLLCLK
00192
00193     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00194     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00195
00196#endif
00197
00198
00199 // MCO1 prescaler:
00200 #if MCO1PRE == NoDivision
00201
00202     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00203     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00204     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00205
00206 #elif MCO1PRE == DivisionBy2
00207
00208     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00209     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00210     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00211
00212 #elif MCO1PRE == DivisionBy3
00213
00214     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00215     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00216     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00217
00218 #elif MCO1PRE == DivisionBy4
00219
00220     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;

```

```

00221     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00222     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00223
00224 #elif MCO1PRE == DivisionBy5
00225
00226     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00227     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00228     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00229
00230 #endif
00231
00232
00233 // AHB prescalers:
00234 #if HPRE == NoDivision
00235
00236     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00237     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00238     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00239     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00240
00241 #elif HPRE == SYSCLKby2
00242
00243     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00244     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00245     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00246     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00247
00248 #elif HPRE == SYSCLKby4
00249
00250     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00251     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00252     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00253     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00254
00255 #elif HPRE == SYSCLKby8
00256
00257     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00258     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00259     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00260     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00261
00262 #elif HPRE == SYSCLKby16
00263
00264     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00265     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00266     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00267     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00268
00269 #elif HPRE == SYSCLKby64
00270
00271     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00272     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00273     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00274     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00275
00276 #elif HPRE == SYSCLKby128
00277
00278     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00279     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00280     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00281     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00282
00283 #elif HPRE == SYSCLKby256
00284
00285     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00286     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00287     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00288     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00289
00290 #elif HPRE == SYSCLKby512
00291
00292     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00293     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00294     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00295     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b3 ) ;
00296
00297 #endif
00298
00299
00300 // APB1 prescalers:
00301 #if PPREG == NoDivision
00302
00303     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00304     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00305     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00306
00307 #elif PPREG == AHBby2

```

```

00308
00309     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b0 ) ;
00310     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b1 ) ;
00311     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b2 ) ;
00312
00313 #elif PPRE1 == AHBby4
00314
00315     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b0 ) ;
00316     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b1 ) ;
00317     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b2 ) ;
00318
00319 #elif PPRE1 == AHBby8
00320
00321     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b0 ) ;
00322     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b1 ) ;
00323     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b2 ) ;
00324
00325 #elif PPRE1 == AHBby16
00326
00327     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b0 ) ;
00328     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b1 ) ;
00329     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE1_b2 ) ;
00330
00331 #endif
00332
00333 // APB2 prescalers:
00334 #if PPRE2 == NoDivision
00335
00336     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00337     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00338     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00339
00340 #elif PPRE2 == AHBby2
00341
00342     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00343     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00344     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00345
00346 #elif PPRE2 == AHBby4
00347
00348     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00349     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00350     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00351
00352 #elif PPRE2 == AHBby8
00353
00354     CLR_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00355     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00356     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00357
00358 #elif PPRE2 == AHBby16
00359
00360     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b0 ) ;
00361     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b1 ) ;
00362     SET_BIT( RCC->CFGR, RCC_CFGR_PPRE2_b2 ) ;
00363
00364 #endif
00365
00366
00367 // PLL configurations:
00368
00369
00370
00371 // Enable the selected CLK (HSI ON/HSE ON/PLL ON): //
00372 // HSE (ON/OFF).
00373 #if HSE_EN == ENABLE
00374
00375     SET_BIT( RCC->CR, RCC_CR_HSEON ) ;
00376
00377 #elif HSE_EN == DISABLE
00378
00379     CLR_BIT( RCC->CR, RCC_CR_HSEON ) ;
00380
00381 #endif
00382
00383
00384 // HSI (ON/OFF).
00385 #if HSI_EN == ENABLE
00386
00387     SET_BIT( RCC->CR, RCC_CR_HSION ) ;
00388
00389 #elif HSI_EN == DISABLE
00390
00391     CLR_BIT( RCC->CR, RCC_CR_HSION ) ;
00392
00393 #endif
00394

```

```
00395 }
00396
00397 /*****
00398 ****/
00399
00400 void MRCC_vEnablePeriphralCLK( u32_t A_u32BusID, u32_t A_u32PeriphralID )
00401 {
00402
00403     switch( A_u32BusID )
00404     {
00405
00406         case RCC_AHB1 :
00407
00408             SET_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00409
00410             break ;
00411
00412
00413         case RCC_AHB2 :
00414
00415             SET_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00416
00417             break ;
00418
00419
00420
00421         case RCC_APB1 :
00422
00423             SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00424
00425             break ;
00426
00427
00428         case RCC_APB2 :
00429
00430             SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00431
00432             break ;
00433
00434         case RCC_AHB1LPENR:
00435
00436             SET_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00437
00438             break ;
00439
00440         default:
00441
00442             // Error wrong Bus ID
00443
00444             break ;
00445
00446     }
00447
00448 }
00449
00450 /*****
00451 ****/
00452
00453 void MRCC_vDisablePeriphralCLK( u32_t A_u32BusID, u32_t A_u32PeriphralID )
00454 {
00455
00456     switch( A_u32BusID )
00457     {
00458
00459         case RCC_AHB1 :
00460
00461             CLR_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00462
00463             break ;
00464
00465
00466         case RCC_AHB2 :
00467
00468             CLR_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00469
00470
00471         case RCC_APB1 :
00472
00473             CLR_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00474
00475
00476         case RCC_APB2 :
```

```

00482     CLR_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00483
00484     break ;
00485
00486
00487
00488     case RCC_AHB1LPENR:
00489
00490     CLR_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00491
00492     break ;
00493
00494
00495     default:
00496
00497     // Error wrong Bus ID
00498
00499     break ;
00500
00501 }
00502
00503 }
00504
00505 /*****
00506 *****/
00507
00508
00509
00510
00511
00512
00513
00514
00515

```

7.118 SPI_config.h

```

00001 /* FILENAME: SPI_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 10/12/2022
00005 */
00006 #ifndef _SPI_config_H
00007 #define _SPI_config_H
00008
00009
00010 /*****
00011             // SPI_CRL configurations //
00012 *****/
00013
00014 /*options:
00015 *FIRST_CLK_Captured
00016 *SECOND_CLK_Captured
00017 */
00018 #define CPHA_MODE SECOND_CLK_Captured
00019
00020 /*****
00021
00022 /*options:
00023 *CLK_IdleAt0
00024 *CLK_IdleAt1
00025 */
00026 #define CPOL_MODE CLK_IdleAt1
00027
00028 /*****
00029
00030 /*options:
00031 *SLAVE
00032 *MASTER
00033 */
00034 #define MSTR_MODE MASTER
00035
00036 /*****
00037
00038 /*options:
00039 *CLK_By2
00040 *CLK_By4
00041 *CLK_By8
00042 *CLK_By16
00043 *CLK_By32
00044 *CLK_By64
00045 *CLK_By128
00046 *CLK_By256

```

```
00047 */
00048 #define BR_MODE CLK_By2
00049
00050 /***** */
00051
00052 /*options:
00053 *ENABLE
00054 *DISABLE
00055 */
00056 #define SPE_MODE ENABLE
00057
00058 /***** */
00059
00060 /*options:
00061 *MSB_First
00062 *LSB_First
00063 */
00064 #define LSBFIRST_MODE MSB_First
00065
00066 /***** */
00067
00068 /*options:
00069 *ENABLE
00070 *DISABLE
00071 */
00072 #define SSM_MODE DISABLE
00073
00074 /***** */
00075
00076 /*options:
00077 *FULL_DUPLEX
00078 *RX_ONLY
00079 */
00080 #define RXONLY_MODE FULL_DUPLEX
00081
00082 /***** */
00083
00084 /*options:
00085 *EIGHT_BITS
00086 *SIXTEEN_BITS
00087 */
00088 #define DFF_MODE EIGHT_BITS
00089
00090 /***** */
00091
00092 /*options:
00093 *CRC_PHASE
00094 *NO_CRC_PHASE
00095 */
00096 #define CRCNEXT_MODE NO_CRC_PHASE
00097
00098 /***** */
00099
00100 /*options:
00101 *ENABLE
00102 *DISABLE
00103 */
00104 #define CRCEN_MODE DISABLE
00105
00106 /***** */
00107
00108 /*options:
00109 *ENABLE
00110 *DISABLE
00111 */
00112 #define BIDIOE_MODE DISABLE
00113
00114 /***** */
00115
00116 /*options:
00117 *OneLine
00118 *TwoLines
00119 */
00120 #define BIDMODE_MODE OneLine
00121
00122 /***** */
00123
00124 /*options:
00125 *HALF_DUPLEX_TX
00126 *HALF_DUPLEX_RX
00127 */
00128 #define HALF_DUPLEX_MODE HALF_DUPLEX_TX
00129
00130 /***** */
00131
00132 /*options:
00133 *SIMPLEX_TX
```

```
00134 /*SIMPLEX_RX
00135 */
00136 #define SIMPLEX_MODE SIMPLEX_TX
00137
00138 /***** // SPI_CR2 configurations *****/
00139
00140 /***** // SPI_SR configurations *****/
00141
00142 /*options:
00143 *ENABLE
00144 *DISABLE
00145 */
00146 #define RXDMAEN_MODE DISABLE
00147
00148 /***** // SPI_SR configurations *****/
00149
00150 /*options:
00151 *ENABLE
00152 *DISABLE
00153 */
00154 #define TXDMAEN_MODE DISABLE
00155
00156 /***** // SPI_SR configurations *****/
00157
00158 /*options:
00159 *ENABLE
00160 *DISABLE
00161 */
00162 #define MULTI_MSTR_MODE DISABLE
00163
00164 /***** // SPI_SR configurations *****/
00165
00166 /*options:
00167 *MOTOROLA
00168 *TI
00169 */
00170 #define FRF_MODE
00171
00172 /***** // SPI_SR configurations *****/
00173
00174 /*options:
00175 *ENABLE
00176 *DISABLE
00177 */
00178 #define ERRIE_MODE DISABLE
00179
00180 /***** // SPI_SR configurations *****/
00181
00182 /*options:
00183 *ENABLE
00184 *DISABLE
00185 */
00186 #define RXNEIE_MODE DISABLE
00187
00188 /***** // SPI_SR configurations *****/
00189
00190 /*options:
00191 *ENABLE
00192 *DISABLE
00193 */
00194 #define TXNEIE_MODE DISABLE
00195
00196
00197 /***** // SPI_SR configurations *****/
00198
00199 /***** // SPI_SR configurations *****/
00200
00201 /*options:
00202 *ERROR
00203 *NO_ERROR
00204 */
00205 #define FRE_MODE DISABLE
00206
00207 /***** // SPI_SR configurations *****/
00208
00209 /*options:
00210 *BUSY
00211 *NOT_BUSY
00212 */
00213 #define BSY_MODE DISABLE
00214
00215 /***** // SPI_SR configurations *****/
00216
00217 /*options:
00218 *OCCURRED
00219 *NOT_OCCURRED
00220 */
```

```

00221 #define OVR_MODE DISABLE
00222
00223 /******
00224 ****
00225 /*options:
00226 *FAULT_OCCURRED
00227 *NO_FAULT_OCCURRED
00228 */
00229 #define MODF_MODE DISABLE
00230
00231 /*****
00232 ****
00233 /*options:
00234 *OCCURRED
00235 *NOT_OCCURRED
00236 */
00237 #define UDR_MODE
00238
00239 /*****
00240 ****
00241 /*options:
00242 *MATCHED
00243 *NOT_MATCHED
00244 */
00245 #define CRCERR_MODE DISABLE
00246
00247 /*****
00248 ****
00249 /*options:
00250 *ENABLE
00251 *DISABLE
00252 */
00253 #define RXNEIE_MODE DISABLE
00254
00255 /*****
00256 ****
00257 /*options:
00258 *ENABLE
00259 *DISABLE
00260 */
00261 #define TXNEIE_MODE DISABLE
00262
00263 /*****
00264 ****
00265 ****
00266 ****
00267 ****
00268 ****
00269 ****
00270 ****
00271 ****
00272 ****
00273 ****
00274 ****
00275 ****
00276 #endif // _SPI_config_H

```

7.119 SPI_interface.h

```

00001 /* FILENAME: SPI_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 10/12/2022
00005 */
00006 #ifndef _SPI_interface_H
00007 #define _SPI_interface_H
00008
00009 /*****
00010 * Peripherals declaration
00011 */
00012
00027 #define SPI1_BASE_ADDRESS 0x40013000
00028
00033 #define SPI2_BASE_ADDRESS 0x40003800
00034
00039 #define SPI3_BASE_ADDRESS 0x40003C00
00040
00045 #define SPI4_BASE_ADDRESS 0x40013400
00046
00058 #define SPI1 ((SPI_MemoryMapType *)SPI1_BASE_ADDRESS)
00059
00064 #define SPI2 ((SPI_MemoryMapType *)SPI2_BASE_ADDRESS)
00065

```

```

00070 #define SPI3 ((SPI_MemoryMapType *)SPI3_BASE_ADDRESS)
00071
00080 typedef struct
00081 {
00085     volatile u32_t CR1;
00089     volatile u32_t CR2;
00093     volatile u32_t SR;
00097     volatile u32_t DR;
00101     volatile u32_t CRCPR;
00105     volatile u32_t RXCRCR;
00109     volatile u32_t TXCRCR;
00113     volatile u32_t I2SCFGR;
00117     volatile u32_t I2SPR;
00118 } SPI_MemoryMapType;
00119
00120 /***** Functions prototypes ****/
00121 /*                                         */
00122 /*****                                         *****/
00123
00131 void MSPI_vInit(SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection);
00132
00140 u16_t MSPI_u16Transcieve(SPI_MemoryMapType *A_SPIx, u16_t A_u16Data);
00141
00148 void MSPI_vMasterTransmit(SPI_MemoryMapType *A_SPIx, u16_t A_u16Data);
00149
00156 u16_t MSPI_u16MasterRecieve(SPI_MemoryMapType *A_SPIx);
00157
00164 void MSPI_vSlaveTransmit(SPI_MemoryMapType *A_SPIx, u16_t A_u16Data);
00165
00172 u16_t MSPI_u16SlaveRecieve(SPI_MemoryMapType *A_SPIx);
00173
00181 void MSPI_vDISABLE(SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection);
00182
00188 void MSPI1_vSetCallBack(void (*Fptr)(void));
00189
00195 void MSPI2_vSetCallBack(void (*Fptr)(void));
00196
00202 void MSPI3_vSetCallBack(void (*Fptr)(void));
00203
00204 /*****                                         ****/
00205 /*                                         */
00206 /*****                                         *****/
00207
00217 #define SPIx_MSTR 1
00218
00223 #define SPIx_SLAVE 2
00224
00229 #define SPIx_SIMPLEX 1
00230
00235 #define SPIx_HALF_DUPLEX 2
00236
00241 #define SPIx_FULL_DUPLEX 3
00242
00245 #endif // _SPI_interface_H

```

7.120 SPI_private.h

```

00001 /* FILENAME: SPI_private
00002 * Author:    Ali El Bana
00003 * Version:   V1.0
00004 * DATE:     Wed 10/12/2022
00005 */
00006 #ifndef _SPI_private_H
00007 #define _SPI_private_H
00008
00009
00010 // CR1 BITS
00011 #define CPHA      0
00012 #define CPOL      1
00013 #define MSTR      2
00014 #define SPE       6
00015 #define LSBFIRST  7
00016 #define SSI       8
00017 #define SSM       9
00018 #define RXONLY    10
00019 #define DFF       11
00020 #define CRCNEXT   12
00021 #define CRCEN     13
00022 #define BIDIOE    14
00023 #define BIDIMODE   15
00024
00025 // CR2 BITS
00026 #define RXDMAEN  0

```

```

00027 #define TXDMAEN 1
00028 #define SSOE 2
00029 #define FRF 4
00030 #define ERRIE 5
00031 #define RXNEIE 6
00032 #define TXEIE 7
00033
00034 // SR BITS
00035 #define RXNE 0
00036 #define TXE 1
00037 #define CHSIDE 2
00038 #define UDR 3
00039 #define CRCERR 4
00040 #define MODF 5
00041 #define OVR 6
00042 #define BSY 7
00043 #define FRE 8
00044
00045
00046
00047
00048 // Configuration options:
00049 #define FIRST_CLK_Captured 1
00050 #define SECOND_CLK_Captured 0
00051
00052 #define CLK_IdleAt0 0
00053 #define CLK_IdleAt1 1
00054
00055 #define SLAVE 0
00056 #define MASTER 1
00057
00058 #define CLK_By2 0b000U
00059 #define CLK_By4 0b001U
00060 #define CLK_By8 0b010U
00061 #define CLK_By16 0b011U
00062 #define CLK_By32 0b100U
00063 #define CLK_By64 0b101U
00064 #define CLK_By128 0b110U
00065 #define CLK_By256 0b111U
00066
00067 #define ENABLE 1
00068 #define DISABLE 0
00069
00070 #define MSB_First 0
00071 #define LSB_First 1
00072
00073 #define FULL_DUPLEX 0
00074 #define RX_ONLY 1
00075
00076 #define EIGHT_BITS 0
00077 #define SIXTEEN_BITS 1
00078
00079 #define CRC_PHASE 1
00080 #define NO_CRC_PHASE 0
00081
00082 #define OneLine 1
00083 #define TwoLines 0
00084
00085 #define MOTOROLA 0
00086 #define TI 1
00087
00088 #define EMPTY 0
00089 #define NOTEMPTY 1
00090
00091 #define LEFT 0
00092 #define RIGHT 1
00093
00094 #define OCCURRED 1
00095 #define NOT_OCCURRED 0
00096
00097 #define MATCHED 0
00098 #define NOT_MATCHED 1
00099
00100 #define FAULT_OCCURRED 1
00101 #define NO_FAULT_OCCURRED 0
00102
00103 #define BUSY 1
00104 #define NOT_BUSY 0
00105
00106 #define ERROR 1
00107 #define NO_ERROR 0
00108
00109 // HALF DUPLEX TX/RX:
00110 #define HALF_DUPLEX_TX 1
00111 #define HALF_DUPLEX_RX 2
00112
00113 // SIMPLEX TX/RX:

```

```

00114 #define SIMPLEX_TX 1
00115 #define SIMPLEX_RX 2
00116
00117 // Reset Flags
00118 #define SR_RESET 0x0002
00119
00120
00121
00122
00123
00124
00125
00126
00127 #endif // _SPI_private_H

```

7.121 SPI_program.c

```

00001 /* FILENAME: SPI_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 10/12/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*          */
00009 /***** */
00010 #include "../../LIB/LSTD_TYPES.h"
00011 #include "../../LIB/LSTD_COMPILER.h"
00012 #include "../../LIB/LSTD_VALUES.h"
00013 #include "../../LIB/LSTD_BITMATH.h"
00014
00015 #include "../RCC/MRCC_interface.h"
00016 #include "../GPIO/GPIO_interface.h"
00017
00018 #include "SPI_interface.h"
00019 #include "SPI_private.h"
00020 #include "SPI_config.h"
00021
00022 /***** */
00023 /***** */
00024
00025 void (*MSPI1_CallBack) (void);
00026 void (*MSPI2_CallBack) (void);
00027 void (*MSPI3_CallBack) (void);
00028
00029 /***** Functions implementations *****/
00030 /*          */
00031 /***** */
00032
00033 void MSPI_vInit( SPI_MemoryMapType * A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection )
00034 {
00035
00036     if( A_SPIx == SPI1 )
00037     {
00038
00039         MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_SPI1EN ) ;
00040
00041         // PINS configurations.
00042         MGPIox_ConfigType MOSI1 =
00043         {
00044             .Port      = GPIO_PORTA      , .Pin       = GPIOx_PIN7
00045             .Mode     = GPIOx_MODE_AF    , .OutputType = GPIOx_PUSH_PULL
00046             .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull
00047             .AF_Type   = GPIOx_AF5
00048         } ;
00049
00050         MGPIox_ConfigType MISO1 =
00051         {
00052             .Port      = GPIO_PORTA      , .Pin       = GPIOx_PIN6
00053             .Mode     = GPIOx_MODE_AF    , .OutputType = GPIOx_PUSH_PULL
00054             .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull
00055             .AF_Type   = GPIOx_AF5
00056         } ;
00057
00058         MGPIox_ConfigType SCK1 =
00059         {
00060             .Port      = GPIO_PORTA      , .Pin       = GPIOx_PIN5
00061             .Mode     = GPIOx_MODE_AF    , .OutputType = GPIOx_PUSH_PULL
00062             .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull
00063             .AF_Type   = GPIOx_AF5
00064         } ;
00065
00066         MGPIox_vInit( &MOSI1 ) ;

```

```

00067     MGPIOx_vInit( &MISO1 ) ;
00068     MGPIOx_vInit( &SCK1 ) ;
00069
00070 }
00071 else if( A_SPIx == SPI2 )
00072 {
00073
00074     MRCC_vEnablePeriphralCLK( RCC_APB1, APB1ENR_SPI2EN ) ;
00075
00076 // PINs configurations.
00077 MGPIOx_ConfigType MOSI2 =
00078 {
00079     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN15      ,
00080     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00081     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00082     .AF_Type    = GPIOx_AF5      ,
00083 } ;
00084
00085 MGPIOx_ConfigType MISO2 =
00086 {
00087     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN14      ,
00088     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00089     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00090     .AF_Type    = GPIOx_AF5      ,
00091 } ;
00092
00093 MGPIOx_ConfigType SCK2 =
00094 {
00095     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN13      ,
00096     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00097     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00098     .AF_Type    = GPIOx_AF5      ,
00099 } ;
00100
00101 MGPIOx_vInit( &MOSI2 ) ;
00102 MGPIOx_vInit( &MISO2 ) ;
00103 MGPIOx_vInit( &SCK2 ) ;
00104
00105 }
00106 else if( A_SPIx == SPI3 )
00107 {
00108
00109     MRCC_vEnablePeriphralCLK( RCC_APB1, APB1ENR_SPI3EN ) ;
00110
00111 // PINs configurations.
00112 MGPIOx_ConfigType MOSI3 =
00113 {
00114     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN5       ,
00115     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00116     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00117     .AF_Type    = GPIOx_AF5      ,
00118 } ;
00119
00120 MGPIOx_ConfigType MISO3 =
00121 {
00122     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN4       ,
00123     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00124     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00125     .AF_Type    = GPIOx_AF5      ,
00126 } ;
00127
00128 MGPIOx_ConfigType SCK3 =
00129 {
00130     .Port      = GPIO_PORTB      , .Pin      = GPIOx_PIN3       ,
00131     .Mode       = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00132     .OutputSpeed = GPIOx_LowSpeed , .InputType  = GPIOx_NoPull   ,
00133     .AF_Type    = GPIOx_AF5      ,
00134 } ;
00135
00136 MGPIOx_vInit( &MOSI3 ) ;
00137 MGPIOx_vInit( &MISO3 ) ;
00138 MGPIOx_vInit( &SCK3 ) ;
00139
00140 }
00141
00142 if( A_u8RelationShip == SPIx_MSTR )
00143 {
00144     // BUAD RATE.
00145     SPI1->CR1 = (A_SPIx->CR1 & ( ~ (BR_MODE) << 3 )) | ( BR_MODE << 3 ) ;
00146 }
00147
00148 // CPHA, CPOL.
00149 #if CPHA_MODE == FIRST_CLK_Captured
00150
00151     CLR_BIT( A_SPIx->CR1, CPHA ) ;
00152
00153 #elif CPHA_MODE == SECOND_CLK_Captured

```

```
00154
00155     SET_BIT( A_SPIx->CR1, CPHA ) ;
00156
00157 #endif
00158
00159 #if CPOL_MODE == CLK_IdleAt0
00160     SET_BIT( A_SPIx->CR1, CPOL ) ;
00162
00163 #elif CPOL_MODE == CLK_IdleAt1
00164     SET_BIT( A_SPIx->CR1, CPOL ) ;
00165
00166 #endif
00168
00169 // Data format.
00170 #if DFF_MODE == EIGHT_BITS
00171     CLR_BIT( A_SPIx->CR1, DFF ) ;
00173
00174 #elif DFF_MODE == SIXTEEN_BITS
00175     SET_BIT( A_SPIx->CR1, DFF ) ;
00177
00178 #endif
00179
00180 // MSB/LSB first.
00181 #if LSBFIRST_MODE == MSB_First
00182     CLR_BIT( A_SPIx->CR1, LSBFIRST ) ;
00183
00185 #elif LSBFIRST_MODE == LSB_First
00186     SET_BIT( A_SPIx->CR1, LSBFIRST ) ;
00188
00189 #endif
00190
00191 // Select SW slave management.
00192 switch( A_u8Relationship )
00193 {
00194     case SPIx_MSTR:
00195
00196         SET_BIT( A_SPIx->CR1, SSM ) ;
00197         SET_BIT( A_SPIx->CR1, SSI ) ;
00198
00199         break ;
00200
00201     case SPIx_SLAVE:
00202
00203         SET_BIT( A_SPIx->CR1, SSM ) ;
00204         CLR_BIT( A_SPIx->CR1, SSI ) ;
00205
00206         break ;
00207
00208 }
00209
00210 // Select Motorola or TI Mode:
00211 CLR_BIT( A_SPIx->CR2, FRF ) ; // Motorola
00212
00213 // MASTER/SLAVE Mode.
00214 switch( A_u8Relationship )
00215 {
00216     case SPIx_MSTR:    SET_BIT( A_SPIx->CR1, MSTR ) ; break ;
00217
00218     case SPIx_SLAVE:  CLR_BIT( A_SPIx->CR1, MSTR ) ; break ;
00219
00220 }
00221
00222 // Data direction:
00223 switch( A_u8Relationship )
00224 {
00225     case SPIx_MSTR:
00226
00227         switch( A_u8DataDirection )
00228     {
00229
00230             case SPIx_FULL_DUPLEX:
00231
00232                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00233                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00234
00235                 break ;
00236
00237             case SPIx_SIMPLEX      :
00238
00239                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00240
00241 }
```

```

00241 #if SIMPLEX_MODE == SIMPLEX_TX
00242             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00244
00245 #elif SIMPLEX_MODE == SIMPLEX_RX
00246             SET_BIT( A_SPIx->CR1, RXONLY ) ;
00248
00249 #endif
00250         break ;
00252
00253     case SPIx_HALF_DUPLEX:
00254
00255 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00256             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00258             SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00259
00260 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00261             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00263             CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00264
00265 #endif
00266         break ;
00268
00269     }
00270
00271     break;
00272
00273     case SPIx_SLAVE:
00274
00275         switch( A_u8DataDirection )
00276     {
00277
00278         case SPIx_FULL_DUPLEX:
00279
00280             CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00281             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00282
00283         break ;
00284
00285         case SPIx_SIMPLEX      :
00286
00287             CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00288
00289 #if SIMPLEX_MODE == SIMPLEX_TX
00290             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00292
00293 #elif SIMPLEX_MODE == SIMPLEX_RX
00294             SET_BIT( A_SPIx->CR1, RXONLY ) ;
00296
00297 #endif
00298         break ;
00299
00300     case SPIx_HALF_DUPLEX:
00302
00303 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00304
00305             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00306             SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00307
00308 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00309
00310             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00311             CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00312
00313 #endif
00314
00315         break ;
00316
00317     }
00318
00319     break;
00320
00321 }
00322
00323 // Multi-master mode:
00324 #if MULTI_MSTR_MODE == ENABLE
00325
00326     CLR_BIT( A_SPIx->CR2, SSOE ) ;
00327

```

```
00328 #elif MULTI_MSTR_MODE == DISABLE
00329     SET_BIT( A_SPIx->CR2, SSOE ) ;
00331 #endif
00333 // EN/DIS IRQ:
00335 #if TXNEIE_MODE == ENABLE
00336     SET_BIT( A_SPIx->CR2, TXEIE ) ;
00338
00339 #elif TXNEIE_MODE == DISABLE
00340     CLR_BIT( A_SPIx->CR2, TXEIE ) ;
00342
00343 #endif
00344
00345 #if RXNEIE_MODE == ENABLE
00346     SET_BIT( A_SPIx->CR2, RXNEIE ) ;
00348
00349 #elif RXNEIE_MODE == DISABLE
00350     CLR_BIT( A_SPIx->CR2, RXNEIE ) ;
00352
00353 #endif
00354
00355 // EN/DIS SPI.
00356     SET_BIT( A_SPIx->CR1, SPE ) ;
00357
00358 }
00359
00360 /*****
00361 ****
00362
00363 u16_t MSPI_u16Transcieve( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00364 {
00365
00366     SPI1->DR = A_u16Data ;
00367
00368     // Wait for transmission complete.
00369     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00370     {
00371         asm("NOP") ;
00372     }
00373
00374     // Wait for the last transmission bit.
00375     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00376     {
00377         asm("NOP") ;
00378     }
00379
00380     // Wait for reception complete.
00381     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00382     {
00383         asm("NOP") ;
00384     }
00385
00386
00387     return A_SPIx->DR ;
00388
00389 }
00390
00391 ****
00392 ****
00393
00394 void MSPI_vMasterTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00395 {
00396
00397     // Start transmission.
00398     SPI1->DR = A_u16Data ;
00399
00400     // Wait for transmission complete.
00401     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00402     {
00403         asm("NOP") ;
00404     }
00405
00406     // Wait for the last transmission bit.
00407     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00408     {
00409         asm("NOP") ;
00410     }
00411
00412 }
00413
00414 ****/
```

```

00415 /*****
00416
00417 u16_t MSPI_u16MasterRecieve( SPI_MemoryMapType * A_SPIx )
00418 {
00419     // Wait for reception complete.
00420     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00421     {
00422         asm("NOP") ;
00423     }
00424
00425     // Return data register.
00426     return A_SPIx->DR ;
00427
00428 }
00429 */
00430
00431 *****/
00432 *****/
00433
00434 void MSPI_vSlaveTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00435 {
00436
00437     // Start transmission.
00438     SPI1->DR = A_u16Data ;
00439
00440     // Wait for transmission complete.
00441     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00442     {
00443         asm("NOP") ;
00444     }
00445
00446     // Wait for the last transmission bit.
00447     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00448     {
00449         asm("NOP") ;
00450     }
00451
00452 }
00453
00454 *****/
00455 *****/
00456
00457 u16_t MSPI_u16SlaveRecieve( SPI_MemoryMapType * A_SPIx )
00458 {
00459
00460     // Wait for reception complete.
00461     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00462     {
00463         asm("NOP") ;
00464     }
00465
00466     // Return data register.
00467     return A_SPIx->DR ;
00468
00469 }
00470
00471 *****/
00472 *****/
00473
00474 void MSPI_vDISABLE( SPI_MemoryMapType * A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection )
00475 {
00476
00477     switch( A_u8DataDirection )
00478     {
00479
00480         case SPIx_FULL_DUPLEX:
00481
00482             // Wait for reception complete.
00483             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00484
00485             // Wait for transmission complete.
00486             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00487
00488             // Wait for the last transmission bit.
00489             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00490
00491             // Disabling the SPIx.
00492             CLR_BIT( A_SPIx->CR1, SPE ) ;
00493
00494         break ;
00495
00496
00497         case SPIx_SIMPLEX      :
00498
00499 #if SIMPLEX_MODE == SIMPLEX_TX
00500
00501             // Wait for transmission complete.

```

```

00502     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00503
00504     // Wait for the last transmission bit.
00505     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00506
00507     // Disabling the SPIx.
00508     CLR_BIT( A_SPIx->CRL, SPE ) ;
00509
00510 #elif SIMPLEX_MODE == SIMPLEX_RX
00511
00512     switch( A_u8Relationship )
00513     {
00514
00515         case SPIx_MSTR:
00516
00517             // Wait for reception complete.
00518             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00519
00520             // Disabling the SPIx.
00521             CLR_BIT( A_SPIx->CRL, SPE ) ;
00522
00523         break ;
00524
00525         case SPIx_SLAVE:
00526
00527             // Disabling the SPIx.
00528             CLR_BIT( A_SPIx->CRL, SPE ) ;
00529
00530             // Wait for the last transmission bit.
00531             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00532
00533         break ;
00534
00535     }
00536
00537 #endif
00538
00539     break ;
00540
00541
00542     case SPIx_HALF_DUPLEX:
00543
00544 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00545
00546         // Wait for transmission complete.
00547         while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00548
00549         // Wait for the last transmission bit.
00550         while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00551
00552         // Disabling the SPIx.
00553         CLR_BIT( A_SPIx->CRL, SPE ) ;
00554
00555 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00556
00557     switch( A_u8Relationship )
00558     {
00559
00560         case SPIx_MSTR:
00561
00562             // Wait for reception complete.
00563             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00564
00565             // Disabling the SPIx.
00566             CLR_BIT( A_SPIx->CRL, SPE ) ;
00567
00568         break ;
00569
00570         case SPIx_SLAVE:
00571
00572             // Disabling the SPIx.
00573             CLR_BIT( A_SPIx->CRL, SPE ) ;
00574
00575             // Wait for the last transmission bit.
00576             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00577
00578         break ;
00579
00580     }
00581
00582 #endif
00583
00584     break ;
00585
00586 }
00587
00588 if( A_SPIx == SPI1 )

```

```

00589     {
00590         MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_SPI1EN ) ;
00591     }
00592     else if( A_SPIx == SPI2 )
00593     {
00594         MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI2EN ) ;
00595     }
00596     else if( A_SPIx == SPI3 )
00597     {
00598         MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI3EN ) ;
00599     }
00600 }
00601 }
00602
00603 /*****
00604 *****/
00605
00606 void MSP11_vSetCallBack( void (*Fptr) (void) )
00607 {
00608     MSP11_CallBack = Fptr ;
00609 }
00610
00611 /*****
00612 *****/
00613
00614 void MSP12_vSetCallBack( void (*Fptr) (void) )
00615 {
00616     MSP12_CallBack = Fptr ;
00617 }
00618
00619 /*****
00620 *****/
00621
00622 void MSP13_vSetCallBack( void (*Fptr) (void) )
00623 {
00624     MSP13_CallBack = Fptr ;
00625 }
00626
00627 /*****
00628 *****/
00629
00630 void SPI1_IRQHandler(void)
00631 {
00632
00633     // CLEAR FLAGS:
00634     SPI1->SR = SR_RESET ;
00635
00636     MSP11_CallBack() ;
00637
00638 }
00639
00640 /*****
00641 *****/
00642
00643 void SPI2_IRQHandler(void)
00644 {
00645
00646     // CLEAR FLAGS:
00647     SPI2->SR = SR_RESET ;
00648
00649     MSP12_CallBack() ;
00650
00651 }
00652
00653 /*****
00654 *****/
00655
00656 void SPI3_IRQHandler(void)
00657 {
00658
00659     // CLEAR FLAGS:
00660     SPI3->SR = SR_RESET ;
00661
00662     MSP13_CallBack() ;
00663
00664 }
00665
00666 /*****
00667 *****/
00668
00669
00670
00671
00672
00673
00674
00675

```

7.122 COTS/MCAL/SysTick/SysTick_config.h File Reference

Configuration file for SysTick driver.

Macros

- #define CLK_SOURCE AHB_DividedBy8
- #define Exception_Request Dont AssertRequest
- #define SINGLE_INTERVAL_MODE (1)
- #define PERIODIC_INTERVAL_MODE (2)
- #define MAX_TICKS (16777216)

7.122.1 Detailed Description

Configuration file for SysTick driver.

This file contains configurations for SysTick driver

Date

9/4/2021

Version

1.0

Author

Ali El Bana

Definition in file [SysTick_config.h](#).

7.123 SysTick_config.h

[Go to the documentation of this file.](#)

```
00001
00010 #ifndef _SysTick_config_H
00011 #define _SysTick_config_H
00012
00024 #define CLK_SOURCE AHB_DividedBy8
00025
00026 /***** 
00027
00033 #define Exception_Request Dont AssertRequest
00034
00035 ****/
00036
00041 #define SINGLE_INTERVAL_MODE (1)
00042
00047 #define PERIODIC_INTERVAL_MODE (2)
00048
00053 #define MAX_TICKS (16777216)
00054
00057 #endif // _SysTick_config_H
```

7.124 COTS/MCAL/SysTick/SysTick_interface.h File Reference

This file contains the interfacing information for the Systick module.

Macros

- `#define BUSY_TICK_TIME (1000)`
Set the time you want to count when using the busy wait function.
- `#define SINGLE_INTERVAL_TICK_TIME (1000)`
Set the time you want to count when using the single interval function.
- `#define PERIODIC_INTERVAL_TICK_TIME (1000)`
Set the time you want to count when using the periodic interval function.
- `#define MILLI_SEC (1)`
Delay in milli seconds.
- `#define MICRO_SEC (2)`
Delay in micro seconds.
- `#define SEC (3)`
Delay in seconds.

Functions

- `void MSysTick_vInit (void)`
Initialize the Systick module.
- `void MSysTick_vSetBusyWait (u32_t A_u32Ticks)`
Synchronous delay function.
- `void MSysTick_vDelay (u32_t A_u32Ticks, u32_t A_u32TickType)`
Synchronous delay function.
- `void MSysTick_vDelayMicroSec (u32_t A_u32Ticks)`
Synchronous delay function.
- `void MSysTick_vDelayMilliSec (u32_t A_u32Ticks)`
Synchronous delay function.
- `void MSysTick_vDelaySec (u32_t A_u32Ticks)`
Synchronous delay function.
- `void MSysTick_vSetSingleInterval (u32_t A_u32Ticks, void(*A_Fptr)(void))`
Asynchronous 1-cycle delay function.
- `void MSysTick_vSetPeriodicInterval (u32_t A_u32Ticks, void(*A_Fptr)(void))`
Asynchronous repetitive delay function.
- `void MSysTick_vStopInterval (void)`
Stop the interval delay function.
- `u32_t MSysTick_u32GetElapsedTime (void)`
Return the elapsed time since the start of the countdown.
- `u32_t MSysTick_u32GetRemainingTime (void)`
Return the remaining time in the systick register.
- `void MSysTick_vEnable (void)`
Enable the Systick timer.
- `void MSysTick_vDisable (void)`
Disable the Systick timer.
- `void MSysTick_vEnableException (void)`
Enable the Systick Exception.
- `void MSysTick_vDisableException (void)`
Disable the Systick Exception.

7.124.1 Detailed Description

This file contains the interfacing information for the Systick module.

Author

Ali El Bana

Version

1.0

Date

09/04/2022

Definition in file [SysTick_interface.h](#).

7.124.2 Macro Definition Documentation

7.124.2.1 BUSY_TICK_TIME

```
#define BUSY_TICK_TIME (1000)
```

Set the time you want to count when using the busy wait function.

Definition at line [118](#) of file [SysTick_interface.h](#).

7.124.2.2 SINGLE_INTERVAL_TICK_TIME

```
#define SINGLE_INTERVAL_TICK_TIME (1000)
```

Set the time you want to count when using the single interval function.

Definition at line [124](#) of file [SysTick_interface.h](#).

7.124.2.3 PERIODIC_INTERVAL_TICK_TIME

```
#define PERIODIC_INTERVAL_TICK_TIME (1000)
```

Set the time you want to count when using the periodic interval function.

Definition at line [130](#) of file [SysTick_interface.h](#).

7.124.3 Function Documentation

7.124.3.1 MSysTick_vInit()

```
void MSysTick_vInit (
    void )
```

Initialize the Systick module.

Definition at line 27 of file [SysTick_program.c](#).

```
00028 {
00029
00030     // Reset timer value.
00031     SysTick->VAL = INITIAL_ZERO;
00032
00033 // Choose the input CLK source.
00034 #if CLK_SOURCE == AHB_DividedBy8
00035     CLR_BIT(SysTick->CTRL, CLKSOURCE);
00036
00037 #elif CLK_SOURCE == AHB
00038     SET_BIT(SysTick->CTRL, CLKSOURCE);
00039 #endif
00040
00041 // SysTick exception request enable.
00042 #if Exception_Request == Dont AssertRequest
00043     CLR_BIT(SysTick->CTRL, TICKINT);
00044
00045 #elif Exception_Request == AssertRequest
00046     SET_BIT(SysTick->CTRL, TICKINT);
00047 #endif
00048 }
```

References [CLKSOURCE](#), [CLR_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.124.3.2 MSysTick_vSetBusyWait()

```
void MSysTick_vSetBusyWait (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
----	-------------------	------------------------------

Definition at line 53 of file [SysTick_program.c](#).

```
00054 {
00055
00056     // Load Ticks to the load register.
00057     SysTick->LOAD = A_u32Ticks;
00058
00059     // Reset timer value.
00060     SysTick->VAL = INITIAL_ZERO;
00061
00062     // Start Timer.
00063     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00064
00065     // Wait till the flag is raised (= 1).
00066     while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00067         ;
```

```

00068 // Stop the timer.
00069 CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00070
00071 // Clear the LOAD and VAL registers
00072 SysTick->LOAD = INITIAL_ZERO;
00073 SysTick->VAL = INITIAL_ZERO;
00074
00075 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), and [SysTick](#).

7.124.3.3 MSysTick_vDelay()

```

void MSysTick_vDelay (
    u32_t A_u32Ticks,
    u32_t A_u32TickType )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_u32TickType</i>	Type of the delay

Definition at line 79 of file [SysTick_program.c](#).

```

00080 {
00081
00082     u32_t l_u32TickNum = INITIAL_ZERO;
00083
00084     if (A_u32TickType == MILLI_SEC)
00085     {
00086         l_u32TickNum = (A_u32Ticks * 1000) - 1;
00087     }
00088
00089     else if (A_u32TickType == MICRO_SEC)
00090     {
00091         l_u32TickNum = A_u32Ticks - 1;
00092     }
00093
00094     else if (A_u32TickType == SEC)
00095     {
00096         l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00097     }
00098
00099     else
00100     {
00101         // error
00102     }
00103
00104     if (l_u32TickNum < MAX_TICKS)
00105     {
00106
00107         // Load Ticks to the load register.
00108         SysTick->LOAD = l_u32TickNum;
00109
00110         // Reset timer value.
00111         SysTick->VAL = INITIAL_ZERO;
00112
00113         // Start Timer.
00114         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00115
00116         // Wait till the flag is raised (= 1).
00117         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00118             ;
00119
00120         // Stop the timer.
00121         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00122
00123         // Clear the LOAD and VAL registers
```

```

00124     SysTick->LOAD = INITIAL_ZERO;
00125     SysTick->VAL = INITIAL_ZERO;
00126 }
00127 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [MICRO_SEC](#), [MILLI_SEC](#), [SEC](#), [SET_BIT](#), and [SysTick](#).

7.124.3.4 MSysTick_vDelayMicroSec()

```
void MSysTick_vDelayMicroSec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	A_u32Ticks	Number of ticks to delay in micro seconds for
----	------------	---

Definition at line 132 of file [SysTick_program.c](#).

```

00133 {
00134
00135     u32_t
00136     l_u32TickNum = INITIAL_ZERO;
00137
00138     l_u32TickNum = (A_u32Ticks)-1;
00139
00140     if (l_u32TickNum < MAX_TICKS)
00141     {
00142
00143         // Load Ticks to the load register.
00144         SysTick->LOAD = l_u32TickNum;
00145
00146         // Reset timer value.
00147         SysTick->VAL = INITIAL_ZERO;
00148
00149         // Start Timer.
00150         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00151
00152         // Wait till the flag is raised (= 1).
00153         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00154             ;
00155
00156         // Stop the timer.
00157         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00158
00159         // Clear the LOAD and VAL registers
00160         SysTick->LOAD = INITIAL_ZERO;
00161         SysTick->VAL = INITIAL_ZERO;
00162     }
00163 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), and [SysTick](#).

Referenced by [HULTSNC_vTrigger\(\)](#).

7.124.3.5 MSysTick_vDelayMilliSec()

```
void MSysTick_vDelayMilliSec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay in milli seconds for
----	-------------------	---

Definition at line 168 of file [SysTick_program.c](#).

```

00169 {
00170
00171     u32_t
00172     l_u32TickNum = INITIAL_ZERO;
00173
00174     l_u32TickNum = (A_u32Ticks * 1000) - 1;
00175
00176     if (l_u32TickNum < MAX_TICKS)
00177     {
00178
00179         // Load Ticks to the load register.
00180         SysTick->LOAD = l_u32TickNum;
00181
00182         // Reset timer value.
00183         SysTick->VAL = INITIAL_ZERO;
00184
00185         // Start Timer.
00186         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00187
00188         // Wait till the flag is raised (= 1).
00189         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00190             ;
00191
00192         // Stop the timer.
00193         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00194
00195         // Clear the LOAD and VAL registers
00196         SysTick->LOAD = INITIAL_ZERO;
00197         SysTick->VAL = INITIAL_ZERO;
00198     }
00199 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), and [SysTick](#).

7.124.3.6 MSysTick_vDelaySec()

```
void MSysTick_vDelaySec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay in seconds for
----	-------------------	---

Definition at line 204 of file [SysTick_program.c](#).

```

00205 {
00206
00207     u32_t
00208     l_u32TickNum = INITIAL_ZERO;
00209
00210     l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00211
00212     if (l_u32TickNum < MAX_TICKS)
00213     {
00214
00215         // Load Ticks to the load register.
00216         SysTick->LOAD = l_u32TickNum;
00217
00218         // Reset timer value.
00219         SysTick->VAL = INITIAL_ZERO;
```

```

00220
00221     // Start Timer.
00222     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00223
00224     // Wait till the flag is raised (= 1).
00225     while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00226         ;
00227
00228     // Stop the timer.
00229     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00230
00231     // Clear the LOAD and VAL registers
00232     SysTick->LOAD = INITIAL_ZERO;
00233     SysTick->VAL = INITIAL_ZERO;
00234 }
00235 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), and [SysTick](#).

7.124.3.7 MSysTick_vSetSingleInterval()

```
void MSysTick_vSetSingleInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Asynchronous 1-cycle delay function.

This function counts down the ticks number, and when it reaches 0, it stops, unlike [MSysTick_vSetPeriodicInterval](#)

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_Fptr</i>	Callback function to call when the countdown has finished

Definition at line 240 of file [SysTick_program.c](#).

```

00241 {
00242
00243     // Save the callback.
00244     GS_vCallbackFunc = A_Fptr;
00245
00246     // Reset timer value.
00247     SysTick->VAL = INITIAL_ZERO;
00248
00249     // Load Ticks to the load register.
00250     SysTick->LOAD = A_u32Ticks;
00251
00252     // Start Timer.
00253     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00254
00255     // Set interval mode to single.
00256     GS_u8MyIntervalMode = SINGLE_INTERVAL_MODE;
00257
00258     // Enable the IRQ.
00259     SET_BIT(SysTick->CTRL, TICKINT);
00260 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [SET_BIT](#), [SINGLE_INTERVAL_MODE](#), [SysTick](#), and [TICKINT](#).

7.124.3.8 MSysTick_vSetPeriodicInterval()

```
void MSysTick_vSetPeriodicInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Asynchronous repetitive delay function.

This function counts down the ticks number, and when it reaches 0, it starts the countdown again

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_Fptr</i>	Callback function to call when the countdown has finished

See also

- [MSysTick_vSetSingleInterval](#) for a 1-cycle delay function
- [MSysTick_vSetBusyWait](#) for a synchronous delay function
- [MSysTick_vStopInterval](#) to stop the interval delay function

Definition at line 265 of file [SysTick_program.c](#).

```
00266 {
00267
00268     // Save the callback.
00269     GS_vCallbackFunc = A_Fptr;
00270
00271     // Reset timer value.
00272     SysTick->VAL = INITIAL_ZERO;
00273
00274     // Load Ticks to the load register.
00275     SysTick->LOAD = A_u32Ticks;
00276
00277     // Start Timer.
00278     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00279
00280     // Set interval mode to single.
00281     GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00282
00283     // Enable the IRQ.
00284     SET_BIT(SysTick->CTRL, TICKINT);
00285 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [PERIODIC_INTERVAL_MODE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.124.3.9 MSysTick_vStopInterval()

```
void MSysTick_vStopInterval (
    void )
```

Stop the interval delay function.

See also

[MSysTick_vSetPeriodicInterval](#) for a interval delay function

Definition at line 290 of file [SysTick_program.c](#).

```
00291 {
00292
00293     // Disable the IRQ.
00294     SET_BIT(SysTick->CTRL, TICKINT);
00295
00296     // Stop the timer.
00297     SET_BIT(SysTick->CTRL, TICKINT);
00298
00299     // Clear the LOAD and VAL registers
00300     SysTick->LOAD = INITIAL_ZERO;
00301     SysTick->VAL = INITIAL_ZERO;
00302 }
```

References [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.124.3.10 MSysTick_u32GetElapsedTime()

```
u32_t MSysTick_u32GetElapsedTime (
    void )
```

Return the elapsed time since the start of the countdown.

Returns

Return the elapsed time since the start of the countdown

Definition at line 308 of file [SysTick_program.c](#).

```
00309 {
00310     u32_t
00311     L_u32ElapsedTime = INITIAL_ZERO;
00312
00313     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);
00314
00315     return L_u32ElapsedTime;
00316 }
```

References [INITIAL_ZERO](#), and [SysTick](#).

7.124.3.11 MSysTick_u32GetRemainingTime()

```
u32_t MSysTick_u32GetRemainingTime (
    void )
```

Return the remaining time in the systick register.

Returns

The remaining time in the Systick register

Definition at line 322 of file [SysTick_program.c](#).

```
00323 {
00324     u32_t
00325     L_u32RemainingTime = INITIAL_ZERO;
00326
00327     L_u32RemainingTime = SysTick->VAL;
00328
00329     return L_u32RemainingTime;
00330 }
```

References [INITIAL_ZERO](#), and [SysTick](#).

7.124.3.12 MSysTick_vEnable()

```
void MSysTick_vEnable (
    void )
```

Enable the Systick timer.

Definition at line 335 of file [SysTick_program.c](#).

```
00336 {
00337
00338     // Start Timer.
00339     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00340 }
```

References [COUNTER_ENABLE](#), [SET_BIT](#), and [SysTick](#).

7.124.3.13 MSysTick_vDisable()

```
void MSysTick_vDisable (
    void )
```

Disable the Systick timer.

Definition at line 345 of file [SysTick_program.c](#).

```
00346 {
00347
00348     // Disable the peripheral interrupt.
00349     SET_BIT(SysTick->CTRL, TICKINT);
00350
00351     // Stop the timer.
00352     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00353 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.124.3.14 MSysTick_vEnableException()

```
void MSysTick_vEnableException (
    void )
```

Enable the Systick Exception.

Definition at line 366 of file [SysTick_program.c](#).

```
00367 {
00368     SET_BIT(SysTick->CTRL, TICKINT);
00369 }
```

References [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.124.3.15 MSysTick_vDisableException()

```
void MSysTick_vDisableException (
    void )
```

Disable the Systick Exception.

Definition at line 358 of file [SysTick_program.c](#).

```
00359 {
00360     CLR_BIT(SysTick->CTRL, TICKINT);
00361 }
```

References [CLR_BIT](#), [SysTick](#), and [TICKINT](#).

7.125 SysTick_interface.h

[Go to the documentation of this file.](#)

```
00001 /* Header file guard */
00009 #ifndef _SysTick_interface_H
00011 #define _SysTick_interface_H
00012
00013 /***** Functions prototypes *****/
00014 /*                                         */
00015 /*****                                         *****/
00016
00020 void MSysTick_vInit(void);
00021
00026 void MSysTick_vSetBusyWait(u32_t A_u32Ticks);
00027
00033 void MSysTick_vDelay(u32_t A_u32Ticks, u32_t A_u32TickType);
00034
00039 void MSysTick_vDelayMicroSec(u32_t A_u32Ticks);
00040
00045 void MSysTick_vDelayMilliSec(u32_t A_u32Ticks);
00046
00051 void MSysTick_vDelaySec(u32_t A_u32Ticks);
00052
00059 void MSysTick_vSetSingleInterval(u32_t A_u32Ticks, void (*A_Fptr)(void));
00060
00070 void MSysTick_vSetPeriodicInterval(u32_t A_u32Ticks, void (*A_Fptr)(void));
00071
00076 void MSysTick_vStopInterval(void);
00077
00082 u32_t MSysTick_u32GetElapsedTime(void);
00083
00088 u32_t MSysTick_u32GetRemainingTime(void);
00089
00093 void MSysTick_vEnable(void);
00094
00098 void MSysTick_vDisable(void);
00099
00103 void MSysTick_vEnableException(void) ;
00104
00108 void MSysTick_vDisableException(void) ;
00109
00110 /*****                                         */
00111 /*                                         */
00112 /*****                                         */
00113
00118 #define BUSY_TICK_TIME (1000)
00119
00124 #define SINGLE_INTERVAL_TICK_TIME (1000)
00125
00130 #define PERIODIC_INTERVAL_TICK_TIME (1000)
00131
00142 #define MILLI_SEC (1)
00143
00148 #define MICRO_SEC (2)
00149
00154 #define SEC (3)
00157 #endif // _SysTick_interface_H
```

7.126 COTS/MCAL/SysTick/SysTick_private.h File Reference

This file contains the registers information and addresses for the Systick module.

Data Structures

- struct [SysTick_Type](#)

Systick memory map structure declaration for the Systick's registers.

Macros

- #define [SysTick_BASE_ADDRESS](#) (0xE000E010)
- #define [SysTick](#) ((volatile P2VAR([SysTick_Type](#)))([SysTick_BASE_ADDRESS](#)))
- #define [COUNTFLAG](#) (16)
- #define [CLKSOURCE](#) (2)
- #define [TICKINT](#) (1)
- #define [COUNTER_ENABLE](#) (0)
- #define [AHB_DividedBy8](#) (1)
- #define [AHB](#) (2)
- #define [Dont_AssertRequest](#) (1)
- #define [AssertRequest](#) (2)

7.126.1 Detailed Description

This file contains the registers information and addresses for the Systick module.

Author

Ali El Bana

Version

1.0

Date

09/04/2022

Definition in file [SysTick_private.h](#).

7.127 SysTick_private.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
00010 #ifndef _SysTick_private_H
00011 #define _SysTick_private_H
00012
00018 typedef struct
00019 {
00024     u32_t CTRL;
00029     u32_t LOAD;
00034     u32_t VAL;
00039     u32_t CALIB;
00040
00041 } SysTick_Type;
00042
00055 #define SysTick_BASE_ADDRESS (0xE000E010)
00056
00070 #define SysTick ((volatile P2VAR(SysTick_Type))(SysTick_BASE_ADDRESS))
00071
00085 #define COUNTFLAG (16)
00086
00091 #define CLKSOURCE (2)
00092
00097 #define TICKINT (1)
00098
00108 #define COUNTER_ENABLE (0)
00120 #define AHB_DividedBy8 (1)
00121
00126 #define AHB (2)
00138 #define Dont_AssertRequest (1)
00139
00144 #define AssertRequest (2)
00147 #endif // _SysTick_private_H

```

7.128 COTS/MCAL/SysTick/SysTick_program.c File Reference

This file contains the source code of the interfacing for the Systick modules.

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "SysTick_interface.h"
#include "SysTick_private.h"
#include "SysTick_config.h"

```

Functions

- void **MSysTick_vInit** (void)
Initialize the Systick module.
- void **MSysTick_vSetBusyWait** (u32_t A_u32Ticks)
Synchronous delay function.
- void **MSysTick_vDelay** (u32_t A_u32Ticks, u32_t A_u32TickType)
Synchronous delay function.
- void **MSysTick_vDelayMicroSec** (u32_t A_u32Ticks)
Synchronous delay function.
- void **MSysTick_vDelayMilliSec** (u32_t A_u32Ticks)
Synchronous delay function.
- void **MSysTick_vDelaySec** (u32_t A_u32Ticks)

- void **MSysTick_vSetSingleInterval** (**u32_t** A_u32Ticks, void(*A_Fptr)(void))
Asynchronous 1-cycle delay function.
- void **MSysTick_vSetPeriodicInterval** (**u32_t** A_u32Ticks, void(*A_Fptr)(void))
Asynchronous repetitive delay function.
- void **MSysTick_vStopInterval** (void)
Stop the interval delay function.
- **u32_t MSysTick_u32GetElapsedTime** (void)
Return the elapsed time since the start of the countdown.
- **u32_t MSysTick_u32GetRemainingTime** (void)
Return the remaining time in the systick register.
- void **MSysTick_vEnable** (void)
Enable the Systick timer.
- void **MSysTick_vDisable** (void)
Disable the Systick timer.
- void **MSysTick_vDisableException** (void)
Disable the Systick Exception.
- void **MSysTick_vEnableException** (void)
Enable the Systick Exception.
- void **SysTick_Handler** (void)
This function is responsible for handling the Systick interrupt.

7.128.1 Detailed Description

This file contains the source code of the interfacing for the Systick modules.

Author

Ali El Bana

Version

1.0

Date

09/04/2022

Definition in file [SysTick_program.c](#).

7.128.2 Function Documentation

7.128.2.1 MSysTick_vInit()

```
void MSysTick_vInit (
    void )
```

Initialize the Systick module.

Definition at line 27 of file [SysTick_program.c](#).

```
00028 {
00029
00030     // Reset timer value.
00031     SysTick->VAL = INITIAL_ZERO;
00032
00033 // Choose the input CLK source.
00034 #if CLK_SOURCE == AHB_DividedBy8
00035     CLR_BIT(SysTick->CTRL, CLKSOURCE);
00036
00037 #elif CLK_SOURCE == AHB
00038     SET_BIT(SysTick->CTRL, CLKSOURCE);
00039 #endif
00040
00041 // SysTick exception request enable.
00042 #if Exception_Request == Dont AssertRequest
00043     CLR_BIT(SysTick->CTRL, TICKINT);
00044
00045 #elif Exception_Request == AssertRequest
00046     SET_BIT(SysTick->CTRL, TICKINT);
00047 #endif
00048 }
```

References [CLKSOURCE](#), [CLR_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.2 MSysTick_vSetBusyWait()

```
void MSysTick_vSetBusyWait (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
----	-------------------	------------------------------

Definition at line 53 of file [SysTick_program.c](#).

```
00054 {
00055
00056     // Load Ticks to the load register.
00057     SysTick->LOAD = A_u32Ticks;
00058
00059     // Reset timer value.
00060     SysTick->VAL = INITIAL_ZERO;
00061
00062     // Start Timer.
00063     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00064
00065     // Wait till the flag is raised (= 1).
00066     while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00067         ;
00068
00069     // Stop the timer.
00070     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00071
00072     // Clear the LOAD and VAL registers
00073     SysTick->LOAD = INITIAL_ZERO;
00074     SysTick->VAL = INITIAL_ZERO;
00075 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), and [SysTick](#).

7.128.2.3 MSysTick_vDelay()

```
void MSysTick_vDelay (
    u32_t A_u32Ticks,
    u32_t A_u32TickType )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_u32TickType</i>	Type of the delay

Definition at line 79 of file [SysTick_program.c](#).

```
00080 {
00081
00082     u32_t l_u32TickNum = INITIAL_ZERO;
00083
00084     if (A_u32TickType == MILLI_SEC)
00085     {
00086         l_u32TickNum = (A_u32Ticks * 1000) - 1;
00087     }
00088
00089     else if (A_u32TickType == MICRO_SEC)
00090     {
00091         l_u32TickNum = A_u32Ticks - 1;
00092     }
00093
00094     else if (A_u32TickType == SEC)
00095     {
00096         l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00097     }
00098
00099     else
00100     {
00101         // error
00102     }
00103
00104     if (l_u32TickNum < MAX_TICKS)
00105     {
00106
00107         // Load Ticks to the load register.
00108         SysTick->LOAD = l_u32TickNum;
00109
00110         // Reset timer value.
00111         SysTick->VAL = INITIAL_ZERO;
00112
00113         // Start Timer.
00114         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00115
00116         // Wait till the flag is raised (= 1).
00117         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00118             ;
00119
00120         // Stop the timer.
00121         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00122
00123         // Clear the LOAD and VAL registers
00124         SysTick->LOAD = INITIAL_ZERO;
00125         SysTick->VAL = INITIAL_ZERO;
00126     }
00127 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [MICRO_SEC](#), [MILLI_SEC](#), [SEC](#), [SET_BIT](#), and [SysTick](#).

7.128.2.4 MSysTick_vDelayMicroSec()

```
void MSysTick_vDelayMicroSec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	A_u32Ticks	Number of ticks to delay in micro seconds for
----	------------	---

Definition at line 132 of file [SysTick_program.c](#).

```
00133 {
00134
00135     u32_t
00136     l_u32TickNum = INITIAL_ZERO;
00137
00138     l_u32TickNum = (A_u32Ticks)-1;
00139
00140     if (l_u32TickNum < MAX_TICKS)
00141     {
00142
00143         // Load Ticks to the load register.
00144         SysTick->LOAD = l_u32TickNum;
00145
00146         // Reset timer value.
00147         SysTick->VAL = INITIAL_ZERO;
00148
00149         // Start Timer.
00150         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00151
00152         // Wait till the flag is raised (= 1).
00153         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00154             ;
00155
00156         // Stop the timer.
00157         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00158
00159         // Clear the LOAD and VAL registers
00160         SysTick->LOAD = INITIAL_ZERO;
00161         SysTick->VAL = INITIAL_ZERO;
00162     }
00163 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), and [SysTick](#).

Referenced by [HULTSNC_vTrigger\(\)](#).

7.128.2.5 MSysTick_vDelayMilliSec()

```
void MSysTick_vDelayMilliSec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	A_u32Ticks	Number of ticks to delay in milli seconds for
----	------------	---

Definition at line 168 of file [SysTick_program.c](#).

```

00169 {
00170     u32_t
00171     l_u32TickNum = INITIAL_ZERO;
00172
00173     l_u32TickNum = (A_u32Ticks * 1000) - 1;
00174
00175     if (l_u32TickNum < MAX_TICKS)
00176     {
00177
00178         // Load Ticks to the load register.
00179         SysTick->LOAD = l_u32TickNum;
00180
00181         // Reset timer value.
00182         SysTick->VAL = INITIAL_ZERO;
00183
00184         // Start Timer.
00185         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00186
00187         // Wait till the flag is raised (= 1).
00188         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00189             ;
00190
00191         // Stop the timer.
00192         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00193
00194         // Clear the LOAD and VAL registers
00195         SysTick->LOAD = INITIAL_ZERO;
00196         SysTick->VAL = INITIAL_ZERO;
00197     }
00198 }
00199 }
```

References `CLR_BIT`, `COUNTER_ENABLE`, `COUNTFLAG`, `FLAG_CLEARED`, `GET_BIT`, `INITIAL_ZERO`, `MAX_TICKS`, `SET_BIT`, and `SysTick`.

7.128.2.6 MSysTick_vDelaySec()

```
void MSysTick_vDelaySec (
    u32_t A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<code>A_u32Ticks</code>	Number of ticks to delay in seconds for
----	-------------------------	---

Definition at line 204 of file `SysTick_program.c`.

```

00205 {
00206     u32_t
00207     l_u32TickNum = INITIAL_ZERO;
00208
00209     l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00210
00211     if (l_u32TickNum < MAX_TICKS)
00212     {
00213
00214         // Load Ticks to the load register.
00215         SysTick->LOAD = l_u32TickNum;
00216
00217         // Reset timer value.
00218         SysTick->VAL = INITIAL_ZERO;
00219
00220         // Start Timer.
00221         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00222
00223         // Wait till the flag is raised (= 1).
00224         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00225             ;
00226
00227         // Stop the timer.
```

```

00229     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00230
00231     // Clear the LOAD and VAL registers
00232     SysTick->LOAD = INITIAL_ZERO;
00233     SysTick->VAL = INITIAL_ZERO;
00234 }
00235 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), and [SysTick](#).

7.128.2.7 MSysTick_vSetSingleInterval()

```

void MSysTick_vSetSingleInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Asynchronous 1-cycle delay function.

This function counts down the ticks number, and when it reaches 0, it stops, unlike [MSysTick_vSetPeriodicInterval](#)

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_Fptr</i>	Callback function to call when the countdown has finished

Definition at line 240 of file [SysTick_program.c](#).

```

00241 {
00242
00243     // Save the callback.
00244     GS_vCallbackFunc = A_Fptr;
00245
00246     // Reset timer value.
00247     SysTick->VAL = INITIAL_ZERO;
00248
00249     // Load Ticks to the load register.
00250     SysTick->LOAD = A_u32Ticks;
00251
00252     // Start Timer.
00253     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00254
00255     // Set interval mode to single.
00256     GS_u8MyIntervalMode = SINGLE_INTERVAL_MODE;
00257
00258     // Enable the IRQ.
00259     SET_BIT(SysTick->CTRL, TICKINT);
00260 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [SET_BIT](#), [SINGLE_INTERVAL_MODE](#), [SysTick](#), and [TICKINT](#).

7.128.2.8 MSysTick_vSetPeriodicInterval()

```

void MSysTick_vSetPeriodicInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Asynchronous repetitive delay function.

This function counts down the ticks number, and when it reaches 0, it starts the countdown again

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
in	<i>A_Fptr</i>	Callback function to call when the countdown has finished

See also

[MSysTick_vSetSingleInterval](#) for a 1-cycle delay function
[MSysTick_vSetBusyWait](#) for a synchronous delay function
[MSysTick_vStopInterval](#) to stop the interval delay function

Definition at line 265 of file [SysTick_program.c](#).

```
00266 {
00267
00268     // Save the callback.
00269     GS_vCallbackFunc = A_Fptr;
00270
00271     // Reset timer value.
00272     SysTick->VAL = INITIAL_ZERO;
00273
00274     // Load Ticks to the load register.
00275     SysTick->LOAD = A_u32Ticks;
00276
00277     // Start Timer.
00278     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00279
00280     // Set interval mode to single.
00281     GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00282
00283     // Enable the IRQ.
00284     SET_BIT(SysTick->CTRL, TICKINT);
00285 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [PERIODIC_INTERVAL_MODE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.9 MSysTick_vStopInterval()

```
void MSysTick_vStopInterval (
    void )
```

Stop the interval delay function.

See also

[MSysTick_vSetPeriodicInterval](#) for a interval delay function

Definition at line 290 of file [SysTick_program.c](#).

```
00291 {
00292
00293     // Disable the IRQ.
00294     SET_BIT(SysTick->CTRL, TICKINT);
00295
00296     // Stop the timer.
00297     SET_BIT(SysTick->CTRL, TICKINT);
00298
00299     // Clear the LOAD and VAL registers
00300     SysTick->LOAD = INITIAL_ZERO;
00301     SysTick->VAL = INITIAL_ZERO;
00302 }
```

References [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.10 MSysTick_u32GetElapsedTime()

```
u32_t MSysTick_u32GetElapsedTime (
    void )
```

Return the elapsed time since the start of the countdown.

Returns

Return the elapsed time since the start of the countdown

Definition at line 308 of file [SysTick_program.c](#).

```
00309 {
00310     u32_t
00311     L_u32ElapsedTime = INITIAL_ZERO;
00312
00313     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);
00314
00315     return L_u32ElapsedTime;
00316 }
```

References [INITIAL_ZERO](#), and [SysTick](#).

7.128.2.11 MSysTick_u32GetRemainingTime()

```
u32_t MSysTick_u32GetRemainingTime (
    void )
```

Return the remaining time in the systick register.

Returns

The remaining time in the Systick register

Definition at line 322 of file [SysTick_program.c](#).

```
00323 {
00324     u32_t
00325     L_u32RemainingTime = INITIAL_ZERO;
00326
00327     L_u32RemainingTime = SysTick->VAL;
00328
00329     return L_u32RemainingTime;
00330 }
```

References [INITIAL_ZERO](#), and [SysTick](#).

7.128.2.12 MSysTick_vEnable()

```
void MSysTick_vEnable (
    void )
```

Enable the Systick timer.

Definition at line 335 of file [SysTick_program.c](#).

```
00336 {
00337
00338     // Start Timer.
00339     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00340 }
```

References [COUNTER_ENABLE](#), [SET_BIT](#), and [SysTick](#).

7.128.2.13 MSysTick_vDisable()

```
void MSysTick_vDisable (
    void )
```

Disable the Systick timer.

Definition at line 345 of file [SysTick_program.c](#).

```
00346 {
00347
00348     // Disable the peripheral interrupt.
00349     SET_BIT(SysTick->CTRL, TICKINT);
00350
00351     // Stop the timer.
00352     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00353 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.14 MSysTick_vDisableException()

```
void MSysTick_vDisableException (
    void )
```

Disable the Systick Exception.

Definition at line 358 of file [SysTick_program.c](#).

```
00359 {
00360     CLR_BIT(SysTick->CTRL, TICKINT);
00361 }
```

References [CLR_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.15 MSysTick_vEnableException()

```
void MSysTick_vEnableException (
    void )
```

Enable the Systick Exception.

Definition at line 366 of file [SysTick_program.c](#).

```
00367 {
00368     SET_BIT(SysTick->CTRL, TICKINT);
00369 }
```

References [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.128.2.16 SysTick_Handler()

```
void SysTick_Handler (
    void )
```

This function is responsible for handling the Systick interrupt.

This function is called by the NVIC when the Systick interrupt is raised

Definition at line 378 of file [SysTick_program.c](#).

```
00379 {
00380     volatile u8_t L_u8ReadFlag = INITIAL_ZERO;
00382
00383     if (GS_vCallbackFunc != NULL)
00384     {
00385         // Callback notification.
00386         GS_vCallbackFunc();
00387     }
00388
00389     if (GS_u8MyIntervalMode == SINGLE_INTERVAL_MODE)
00390     {
00391         // Disable the IRQ.
00392         CLR_BIT(SysTick->CTRL, TICKINT);
00393
00394         // Stop the timer.
00395         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00396
00397         // Clear the LOAD and VAL registers
00398         SysTick->LOAD = INITIAL_ZERO;
00399         SysTick->VAL = INITIAL_ZERO;
00400
00401         GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00402     }
00403
00404     // Clear IRQ flag.
00405     L_u8ReadFlag = GET_BIT(SysTick->CTRL, COUNTFLAG);
00406 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [GET_BIT](#), [INITIAL_ZERO](#), [NULL](#), [PERIODIC_INTERVAL_MODE](#), [SINGLE_INTERVAL_MODE](#), [SysTick](#), and [TICKINT](#).

7.129 SysTick_program.c

[Go to the documentation of this file.](#)

```
00001
00009 /*****
00010 /*           Include headers
00011 *****/
00012 #include "../../../LIB/LSTD_TYPES.h"
00013 #include "../../../LIB/LSTD_COMPILER.h"
00014 #include "../../../LIB/LSTD_VALUES.h"
00015 #include "../../../LIB/LSTD_BITMATH.h"
00016 #include "SysTick_interface.h"
00017 #include "SysTick_private.h"
00018 #include "SysTick_config.h"
00019
00020 STATIC P2FUNC(VAR(void), GS_vCallbackFunc) (void) = NULL;
00021
00022 STATIC VAR(u8_t) GS_u8MyIntervalMode = INITIAL_ZERO;
00023
00024 /*****
00025 *****/
00026
00027 void MSysTick_vInit(void)
00028 {
00029
00030     // Reset timer value.
00031     SysTick->VAL = INITIAL_ZERO;
00032
00033 // Choose the input CLK source.
00034 #if CLK_SOURCE == AHB_DividedBy8
00035     CLR_BIT(SysTick->CTRL, CLKSOURCE);
```

```

00036
00037 #elif CLK_SOURCE == AHB
00038     SET_BIT(SysTick->CTRL, CLKSOURCE);
00039 #endif
00040
00041 // SysTick exception request enable.
00042 #if Exception_Request == Dont_AassertRequest
00043     CLR_BIT(SysTick->CTRL, TICKINT);
00044
00045 #elif Exception_Request == AssertRequest
00046     SET_BIT(SysTick->CTRL, TICKINT);
00047 #endif
00048 }
00049
00050
00051 /*****
00052 *****/
00053 void MSysTick_vSetBusyWait(u32_t A_u32Ticks)
00054 {
00055
00056     // Load Ticks to the load register.
00057     SysTick->LOAD = A_u32Ticks;
00058
00059     // Reset timer value.
00060     SysTick->VAL = INITIAL_ZERO;
00061
00062     // Start Timer.
00063     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00064
00065     // Wait till the flag is raised (= 1).
00066     while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00067         ;
00068
00069     // Stop the timer.
00070     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00071
00072     // Clear the LOAD and VAL registers
00073     SysTick->LOAD = INITIAL_ZERO;
00074     SysTick->VAL = INITIAL_ZERO;
00075 }
00076
00077
00078 /*****
00079 *****/
00079 void MSysTick_vDelay(u32_t A_u32Ticks, u32_t A_u32TickType)
00080 {
00081
00082     u32_t l_u32TickNum = INITIAL_ZERO;
00083
00084     if (A_u32TickType == MILLI_SEC)
00085     {
00086         l_u32TickNum = (A_u32Ticks * 1000) - 1;
00087     }
00088
00089     else if (A_u32TickType == MICRO_SEC)
00090     {
00091         l_u32TickNum = A_u32Ticks - 1;
00092     }
00093
00094     else if (A_u32TickType == SEC)
00095     {
00096         l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00097     }
00098
00099     else
00100     {
00101         // error
00102     }
00103
00104     if (l_u32TickNum < MAX_TICKS)
00105     {
00106
00107         // Load Ticks to the load register.
00108         SysTick->LOAD = l_u32TickNum;
00109
00110         // Reset timer value.
00111         SysTick->VAL = INITIAL_ZERO;
00112
00113         // Start Timer.
00114         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00115
00116         // Wait till the flag is raised (= 1).
00117         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00118             ;

```

```

00119
00120      // Stop the timer.
00121      CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00122
00123      // Clear the LOAD and VAL registers
00124      SysTick->LOAD = INITIAL_ZERO;
00125      SysTick->VAL = INITIAL_ZERO;
00126  }
00127 }
00128
00129
00130 /*****
00131 *****/
00132 void MSysTick_vDelayMicroSec(u32_t A_u32Ticks)
00133 {
00134
00135     u32_t
00136     l_u32TickNum = INITIAL_ZERO;
00137
00138     l_u32TickNum = (A_u32Ticks)-1;
00139
00140     if (l_u32TickNum < MAX_TICKS)
00141     {
00142
00143         // Load Ticks to the load register.
00144         SysTick->LOAD = l_u32TickNum;
00145
00146         // Reset timer value.
00147         SysTick->VAL = INITIAL_ZERO;
00148
00149         // Start Timer.
00150         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00151
00152         // Wait till the flag is raised (= 1).
00153         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00154             ;
00155
00156         // Stop the timer.
00157         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00158
00159         // Clear the LOAD and VAL registers
00160         SysTick->LOAD = INITIAL_ZERO;
00161         SysTick->VAL = INITIAL_ZERO;
00162     }
00163 }
00164
00165
00166 /*****
00167 *****/
00168 void MSysTick_vDelayMilliSec(u32_t A_u32Ticks)
00169 {
00170
00171     u32_t
00172     l_u32TickNum = INITIAL_ZERO;
00173
00174     l_u32TickNum = (A_u32Ticks * 1000) - 1;
00175
00176     if (l_u32TickNum < MAX_TICKS)
00177     {
00178
00179         // Load Ticks to the load register.
00180         SysTick->LOAD = l_u32TickNum;
00181
00182         // Reset timer value.
00183         SysTick->VAL = INITIAL_ZERO;
00184
00185         // Start Timer.
00186         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00187
00188         // Wait till the flag is raised (= 1).
00189         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00190             ;
00191
00192         // Stop the timer.
00193         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00194
00195         // Clear the LOAD and VAL registers
00196         SysTick->LOAD = INITIAL_ZERO;
00197         SysTick->VAL = INITIAL_ZERO;
00198     }
00199 }
00200
00201

```

```

00202 /*****
00203 ****/
00204 void MSysTick_vDelaySec(u32_t A_u32Ticks)
00205 {
00206
00207     u32_t
00208     l_u32TickNum = INITIAL_ZERO;
00209
00210     l_u32TickNum = (A_u32Ticks * 1000000) - 1;
00211
00212     if (l_u32TickNum < MAX_TICKS)
00213     {
00214
00215         // Load Ticks to the load register.
00216         SysTick->LOAD = l_u32TickNum;
00217
00218         // Reset timer value.
00219         SysTick->VAL = INITIAL_ZERO;
00220
00221         // Start Timer.
00222         SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00223
00224         // Wait till the flag is raised (= 1).
00225         while (GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED)
00226             ;
00227
00228         // Stop the timer.
00229         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00230
00231         // Clear the LOAD and VAL registers
00232         SysTick->LOAD = INITIAL_ZERO;
00233         SysTick->VAL = INITIAL_ZERO;
00234     }
00235 }
00236
00237 /*****
00238 ****/
00239
00240 void MSysTick_vSetSingleInterval(u32_t A_u32Ticks, void (*A_Fptr)(void))
00241 {
00242
00243     // Save the callback.
00244     GS_vCallbackFunc = A_Fptr;
00245
00246     // Reset timer value.
00247     SysTick->VAL = INITIAL_ZERO;
00248
00249     // Load Ticks to the load register.
00250     SysTick->LOAD = A_u32Ticks;
00251
00252     // Start Timer.
00253     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00254
00255     // Set interval mode to single.
00256     GS_u8MyIntervalMode = SINGLE_INTERVAL_MODE;
00257
00258     // Enable the IRQ.
00259     SET_BIT(SysTick->CTRL, TICKINT);
00260 }
00261
00262 /*****
00263 ****/
00264
00265 void MSysTick_vSetPeriodicInterval(u32_t A_u32Ticks, void (*A_Fptr)(void))
00266 {
00267
00268     // Save the callback.
00269     GS_vCallbackFunc = A_Fptr;
00270
00271     // Reset timer value.
00272     SysTick->VAL = INITIAL_ZERO;
00273
00274     // Load Ticks to the load register.
00275     SysTick->LOAD = A_u32Ticks;
00276
00277     // Start Timer.
00278     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00279
00280     // Set interval mode to single.
00281     GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00282

```

```
00283     // Enable the IRQ.  
00284     SET_BIT(SysTick->CTRL, TICKINT);  
00285 }  
00286  
00287 /*****  
00288 /*****  
00289  
00290 void MSysTick_vStopInterval(void)  
00291 {  
00292     // Disable the IRQ.  
00293     SET_BIT(SysTick->CTRL, TICKINT);  
00295  
00296     // Stop the timer.  
00297     SET_BIT(SysTick->CTRL, TICKINT);  
00298  
00299     // Clear the LOAD and VAL registers  
00300     SysTick->LOAD = INITIAL_ZERO;  
00301     SysTick->VAL = INITIAL_ZERO;  
00302 }  
00303  
00304 /*****  
00305 /*****  
00306  
00307 u32_t  
00308 MSysTick_u32GetElapsedTime(void)  
00309 {  
00310     u32_t  
00311     L_u32ElapsedTime = INITIAL_ZERO;  
00312  
00313     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);  
00314  
00315     return L_u32ElapsedTime;  
00316 }  
00317  
00318 /*****  
00319 /*****  
00320  
00321 u32_t  
00322 MSysTick_u32GetRemainingTime(void)  
00323 {  
00324     u32_t  
00325     L_u32RemainingTime = INITIAL_ZERO;  
00326  
00327     L_u32RemainingTime = SysTick->VAL;  
00328  
00329     return L_u32RemainingTime;  
00330 }  
00331  
00332 /*****  
00333 /*****  
00334  
00335 void MSysTick_vEnable(void)  
00336 {  
00337  
00338     // Start Timer.  
00339     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);  
00340 }  
00341  
00342 /*****  
00343 /*****  
00344  
00345 void MSysTick_vDisable(void)  
00346 {  
00347  
00348     // Disable the peripheral interrupt.  
00349     SET_BIT(SysTick->CTRL, TICKINT);  
00350  
00351     // Stop the timer.  
00352     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);  
00353 }  
00354  
00355 /*****  
00356 /*****  
00357 
```

```

00358 void MSysTick_vDisableException(void)
00359 {
00360     CLR_BIT(SysTick->CTRL, TICKINT);
00361 }
00362
00363
00364 /*****
00365 ****
00366 void MSysTick_vEnableException(void)
00367 {
00368     SET_BIT(SysTick->CTRL, TICKINT);
00369 }
00370
00371
00372 /*****
00373 ****
00374 void SysTick_Handler(void)
00375 {
00376
00377     volatile u8_t L_u8ReadFlag = INITIAL_ZERO;
00378
00379     if (GS_vCallbackFunc != NULL)
00380     {
00381         // Callback notification.
00382         GS_vCallbackFunc();
00383     }
00384
00385     if (GS_u8MyIntervalMode == SINGLE_INTERVAL_MODE)
00386     {
00387         // Disable the IRQ.
00388         CLR_BIT(SysTick->CTRL, TICKINT);
00389
00390         // Stop the timer.
00391         CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00392
00393         // Clear the LOAD and VAL registers
00394         SysTick->LOAD = INITIAL_ZERO;
00395         SysTick->VAL = INITIAL_ZERO;
00396
00397         GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00398     }
00399
00400     // Clear IRQ flag.
00401     L_u8ReadFlag = GET_BIT(SysTick->CTRL, COUNTFLAG);
00402 }
00403
00404
00405 /*****
00406 ****
00407 ****
00408 ****
00409 ****
00410 ****
00411 ****
00412 ****
00413 ****
00414 ****
00415 ****
00416 ****
00417 #define UPDATE_CNTRL_ENABLE
00418
00419 ****
00420 ****
00421 ****
00422 ****
00423 ****
00424 ****
00425 #define UPDATE_SRC ANY_EVENT
00426

```

7.130 TIM1_config.h

```

00001 /* FILENAME: TIM1_config
00002 * Author:    Ali El Bana
00003 * Version:   V1.0
00004 * DATE:     Tue 01/24/2023
00005 */
00006 #ifndef _TIM1_config_H
00007 #define _TIM1_config_H
00008
00009 ****
00010 /*
00011     TIM1_CRL configurations
00012 */
00013 /*options:
00014 *ENABLE
00015 *DISABLE
00016 */
00017 #define UPDATE_CNTRL_ENABLE
00018
00019 ****
00020
00021 /*options:
00022 *ANY_EVENT
00023 *OVF_UDF
00024 */
00025 #define UPDATE_SRC ANY_EVENT
00026

```

```
00027 /*****  
00028  
00029 /*options:  
00030 *ENABLE  
00031 *DISABLE  
00032 */  
00033 #define OP_MODE DISABLE  
00034  
00035 /*****  
00036  
00037 /*options:  
00038 *UP  
00039 *DOWN  
00040 */  
00041 #define CNT_DIR UP  
00042  
00043 /*****  
00044  
00045 /*options:  
00046 *EDGE  
00047 *CENTER1  
00048 *CENTER2  
00049 *CENTER3  
00050 */  
00051 #define CMS CENTER3  
00052  
00053 /*****  
00054  
00055 /*options:  
00056 *NOT_BUFFERED  
00057 *BUFFERED  
00058 */  
00059 #define ARP_EN BUFFERED  
00060  
00061 /*****  
00062  
00063 /*options:  
00064 *ONE_TCK  
00065 *TWO_TCK  
00066 *FOUR_TCK  
00067 */  
00068 #define CKD ONE_TCK  
00069  
00070 /*****  
00071 /* TIM1_CR2 configurations */  
00072 /*****  
00073  
00074 /*options:  
00075 *NOT_PRELOADED  
00076 *PRELOADED  
00077 */  
00078 #define CCP_CTRL NOT_PRELOADED  
00079  
00080 /*****  
00081  
00082 /*options:  
00083 *COMG_ONLY  
00084 *ANY_EVENT  
00085 */  
00086 #define CCU_SLCT COMG_ONLY  
00087  
00088 /*****  
00089  
00090 /*options:  
00091 *CCx_EVENT  
00092 *UPDATE_EVENT  
00093 */  
00094 #define CCD_SLCT CCx_EVENT  
00095  
00096 /*****  
00097  
00098 /*options:  
00099 *RST  
00100 *ENABLE  
00101 *UPDATE  
00102 *COMP_PULSE  
00103 *COMP1  
00104 *COMP2  
00105 *COMP3  
00106 *COMP4  
00107 */  
00108 #define MMS COMP1  
00109  
00110 /*****  
00111  
00112 /*options:  
00113 *CH1
```

```
00114 *CH1_2_3
00115 */
00116 #define TI1_SLCT CH1
00117
00118 /***** */
00119
00120 /*options:
00121 *LOW
00122 *HIGH
00123 */
00124 #define OC1_IDLE_STATE LOW
00125
00126 /***** */
00127
00128 /*options:
00129 *LOW
00130 *HIGH
00131 */
00132 #define OC1N_IDLE_STATE LOW
00133
00134 /***** */
00135
00136 /*options:
00137 *LOW
00138 *HIGH
00139 */
00140 #define OC2_IDLE_STATE LOW
00141
00142 /***** */
00143
00144 /*options:
00145 *LOW
00146 *HIGH
00147 */
00148 #define OC2N_IDLE_STATE LOW
00149
00150 /***** */
00151
00152 /*options:
00153 *LOW
00154 *HIGH
00155 */
00156 #define OC3_IDLE_STATE LOW
00157
00158 /***** */
00159
00160 /*options:
00161 *LOW
00162 *HIGH
00163 */
00164 #define OC3N_IDLE_STATE LOW
00165
00166 /***** */
00167
00168 /*options:
00169 *LOW
00170 *HIGH
00171 */
00172 #define OC4_IDLE_STATE LOW
00173
00174 /***** */
00175 /*          TIM1_SMCR configurations */
00176 /***** */
00177
00178 /*options:
00179 *DISABLE
00180 *ENCODER1
00181 *ENCODER2
00182 *ENCODER3
00183 *RST
00184 *GATED
00185 *TRIGGER
00186 *EXT_CLK
00187 */
00188 #define SMS_DISABLE
00189
00190 /***** */
00191
00192 /*options:
00193 *ITR0
00194 *ITR1
00195 *ITR2
00196 *ITR3
00197 *TI1F_ED
00198 *TI1FP1
00199 *TI1FP2
00200 *ETRF
```

```
00201 */
00202 #define TS_ITR0
00203
00204 /*****
00205
00206 /*options:
00207 *NO_ACTION
00208 *TRGI
00209 */
00210 #define MSTR_SLAVE_MODE TRGI
00211
00212 *****/
00213
00214 /*options:
00215 *NO_FLTR
00216 *FCK_N2
00217 *FCK_N4
00218 *FCK_N8
00219 *FDTS2_N6
00220 *FDTS2_N8
00221 *FDTS4_N6
00222 *FDTS4_N8
00223 *FDTS8_N6
00224 *FDTS8_N8
00225 *FDTS16_N5
00226 *FDTS16_N6
00227 *FDTS16_N8
00228 *FDTS32_N5
00229 *FDTS32_N6
00230 *FDTS32_N8
00231 */
00232 #define EXT_FLTR NO_FLTR
00233
00234 *****/
00235
00236 /*options:
00237 *OFF
00238 *BY2
00239 *BY4
00240 *BY8
00241 */
00242 #define ETPS OFF
00243
00244 *****/
00245
00246 /*options:
00247 *ENABLE
00248 *DISABLE
00249 */
00250 #define EXT_CLK_EN DISABLE
00251
00252 *****/
00253
00254 /*options:
00255 *NON_INVERTED
00256 *INVERTED
00257 */
00258 #define EXTP NON_INVERTED
00259
00260 *****/
00261 /* TIM1_EGR configurations */
00262 *****/
00263
00264 /*options:
00265 *NO_ACTION
00266 *RE_INIT
00267 */
00268 #define UP_GEN RE_INIT
00269
00270 *****/
00271
00272 /*options:
00273 *NO_ACTION
00274 *CC_GENERATED
00275 */
00276 #define CC1_GEN NO_ACTION
00277
00278 *****/
00279
00280 /*options:
00281 *NO_ACTION
00282 *CC_GENERATED
00283 */
00284 #define CC2_GEN NO_ACTION
00285
00286 *****/
00287
```

```
00288 /*options:  
00289 *NO_ACTION  
00290 *CC_GENERATED  
00291 */  
00292 #define CC3_GEN NO_ACTION  
00293  
00294 /*****  
00295  
00296 /*options:  
00297 *NO_ACTION  
00298 *CC_GENERATED  
00299 */  
00300 #define CC4_GEN NO_ACTION  
00301  
00302 /*****  
00303  
00304 /*options:  
00305 *NO_ACTION  
00306 *CC_UPDATED  
00307 */  
00308 #define COM_GEN NO_ACTION  
00309  
00310 /*****  
00311  
00312 /*options:  
00313 *NO_ACTION  
00314 *TRG_ENABLED  
00315 */  
00316 #define TRG_GEN NO_ACTION  
00317  
00318 /*****  
00319  
00320 /*options:  
00321 *NO_ACTION  
00322 *BRK_GENERATED  
00323 */  
00324 #define BRK_GEN NO_ACTION  
00325  
00326 /*****  
00327 /* TIM1_CCMR1 configurations */  
00328 /*****  
00329  
00330 /*options:  
00331 *CC1_OUTPUT  
00332 *CC1_INPUT_TI1  
00333 *CC1_INPUT_TI2  
00334 *CC1_INPUT_TRC  
00335 */  
00336 #define CC1S CC1_OUTPUT  
00337  
00338 /*****  
00339  
00340 /*options:  
00341 *NO_PRESCALER  
00342 *EVERY_2_EVENTS  
00343 *EVERY_4_EVENTS  
00344 *EVERY_8_EVENTS  
00345 */  
00346 #define IC1PSC NO_PRESCALER  
00347  
00348 /*****  
00349  
00350 /*options:  
00351 *CC2_OUTPUT  
00352 *CC2_INPUT_TI1  
00353 *CC2_INPUT_TI2  
00354 *CC2_INPUT_TRC  
00355 */  
00356 #define CC2S CC2_OUTPUT  
00357  
00358 /*****  
00359  
00360 /*options:  
00361 *NO_PRESCALER  
00362 *EVERY_2_EVENTS  
00363 *EVERY_4_EVENTS  
00364 *EVERY_8_EVENTS  
00365 */  
00366 #define IC2PSC NO_PRESCALER  
00367  
00368 /*****  
00369  
00370 /*options:  
00371 *NORMALLY  
00372 *INPUT_TRIGGER  
00373 */  
00374 #define OC1F_EN NORMALLY
```

```
00375
00376 /****** */
00377
00378 /*options:
00379 *ENABLE
00380 *DISABLE
00381 */
00382 #define OC1P_EN ENABLE
00383
00384 /****** */
00385
00386 /*options:
00387 *FROZEN
00388 *OC1REF_HIGH
00389 *OC1REF_LOW
00390 *OC1REF_TOGGLE
00391 *OC1REF_FORCED_LOW
00392 *OC1REF_FORCED_HIGH
00393 *PWM1
00394 *PWM2
00395 */
00396 #define OC1M PWM1
00397
00398 /****** */
00399
00400 /*options:
00401 *OC1Ref_NOTEFFECTED
00402 *OC1Ref_CLEARED
00403 */
00404 #define OC1C_EN OC1Ref_NOTEFFECTED
00405
00406 /****** */
00407
00408 /*options:
00409 *NORMALLY
00410 *INPUT_TRIGGER
00411 */
00412 #define OC2F_EN NO_ACTION
00413
00414 /****** */
00415
00416 /*options:
00417 *ENABLE
00418 *DISABLE
00419 */
00420 #define OC2P_EN DISABLE
00421
00422 /****** */
00423
00424 /*options:
00425 *FROZEN
00426 *OC2REF_HIGH
00427 *OC2REF_LOW
00428 *OC2REF_TOGGLE
00429 *OC2REF_FORCED_LOW
00430 *OC2REF_FORCED_HIGH
00431 *PWM1
00432 *PWM2
00433 */
00434 #define OC2M FROZEN
00435
00436 /****** */
00437
00438 /*options:
00439 *OC2Ref_NOTEFFECTED
00440 *OC2Ref_CLEARED
00441 */
00442 #define OC2C_EN OC2Ref_NOTEFFECTED
00443
00444 /****** */
00445 /* TIM1_CCMR2 configurations */
00446 /****** */
00447
00448 /*options:
00449 *CC3_OUTPUT
00450 *CC3_INPUT_TI3
00451 *CC3_INPUT_TI4
00452 *CC3_INPUT_TRC
00453 */
00454 #define CC3S CC3_OUTPUT
00455
00456 /****** */
00457
00458 /*options:
00459 *NO_PRESCALER
00460 *EVERY_2_EVENTS
00461 *EVERY_4_EVENTS
```

```
00462 /*EVERY_8_EVENTS
00463 */
00464 #define IC3PSC NO_PRESCALER
00465
00466 /*****
00467
00468 /*options:
00469 *CC4_OUTPUT
00470 *CC4_INPUT_TI4
00471 *CC4_INPUT_TI3
00472 *CC4_INPUT_TRC
00473 */
00474 #define CC4S CC4_OUTPUT
00475
00476 /*****
00477
00478 /*options:
00479 *NO_PRESCALER
00480 *EVERY_2_EVENTS
00481 *EVERY_4_EVENTS
00482 *EVERY_8_EVENTS
00483 */
00484 #define IC4PSC NO_PRESCALER
00485
00486 /*****
00487
00488 /*options:
00489 *NORMALLY
00490 *INPUT_TRIGGER
00491 */
00492 #define OC3F_EN NORMALLY
00493
00494 /*****
00495
00496 /*options:
00497 *ENABLE
00498 *DISABLE
00499 */
00500 #define OC3P_EN DISABLE
00501
00502 /*****
00503
00504 /*options:
00505 *FROZEN
00506 *OC3REF_HIGH
00507 *OC3REF_LOW
00508 *OC3REF_TOGGLE
00509 *OC3REF_FORCED_LOW
00510 *OC3REF_FORCED_HIGH
00511 *PWM1
00512 *PWM2
00513 */
00514 #define OC3M FROZEN
00515
00516 /*****
00517
00518 /*options:
00519 *OC3Ref_NOTEFFECTED
00520 *OC3Ref_CLEARED
00521 */
00522 #define OC3C_EN OC3Ref_NOTEFFECTED
00523
00524 /*****
00525
00526 /*options:
00527 *NORMALLY
00528 *INPUT_TRIGGER
00529 */
00530 #define OC4F_EN NORMALLY
00531
00532 /*****
00533
00534 /*options:
00535 *ENABLE
00536 *DISABLE
00537 */
00538 #define OC4P_EN DISABLE
00539
00540 /*****
00541
00542 /*options:
00543 *FROZEN
00544 *OC4REF_HIGH
00545 *OC4REF_LOW
00546 *OC4REF_TOGGLE
00547 *OC4REF_FORCED_LOW
00548 *OC4REF_FORCED_HIGH
```

```
00549 *PWM1
00550 *PWM2
00551 */
00552 #define OC4M FROZEN
00553
00554 /***** */
00555
00556 /*options:
00557 *OC4Ref_NOTEFFECTED
00558 *OC4Ref_CLEARED
00559 */
00560 #define OC4C_EN OC4Ref_NOTEFFECTED
00561
00562 /***** */
00563 /*          TIM1_CCER configurations */
00564 /***** */
00565
00566 /*options:
00567 *ENABLE
00568 *DISABLE
00569 */
00570 #define CC1_EN ENABLE
00571
00572 /***** */
00573
00574 /*options:
00575 *ACTIVE_HIGH
00576 *ACTIVE_LOW
00577 */
00578 #define CC1_POLARITY ACTIVE_HIGH
00579
00580 /***** */
00581
00582 /*options:
00583 *ENABLE
00584 *DISABLE
00585 */
00586 #define CC1N_EN ENABLE
00587
00588 /***** */
00589
00590 /*options:
00591 *ACTIVE_HIGH
00592 *ACTIVE_LOW
00593 */
00594 #define CC1N_POLARITY ACTIVE_HIGH
00595
00596 /***** */
00597
00598 /*options:
00599 *ENABLE
00600 *DISABLE
00601 */
00602 #define CC2_EN DISABLE
00603
00604 /***** */
00605
00606 /*options:
00607 *ACTIVE_HIGH
00608 *ACTIVE_LOW
00609 */
00610 #define CC2_POLARITY ACTIVE_HIGH
00611
00612 /***** */
00613
00614 /*options:
00615 *ENABLE
00616 *DISABLE
00617 */
00618 #define CC2N_EN DISABLE
00619
00620 /***** */
00621
00622 /*options:
00623 *ACTIVE_HIGH
00624 *ACTIVE_LOW
00625 */
00626 #define CC2N_POLARITY ACTIVE_HIGH
00627
00628 /***** */
00629
00630 /*options:
00631 *ENABLE
00632 *DISABLE
00633 */
00634 #define CC3_EN DISABLE
00635
```

```
00636 /*****  
00637  
00638 /*options:  
00639 *ACTIVE_HIGH  
00640 *ACTIVE_LOW  
00641 */  
00642 #define CC3_POLARITY ACTIVE_HIGH  
00643  
00644 /*****  
00645  
00646 /*options:  
00647 *ENABLE  
00648 *DISABLE  
00649 */  
00650 #define CC3N_EN DISABLE  
00651  
00652 /*****  
00653  
00654 /*options:  
00655 *ACTIVE_HIGH  
00656 *ACTIVE_LOW  
00657 */  
00658 #define CC3N_POLARITY ACTIVE_HIGH  
00659  
00660 /*****  
00661  
00662 /*options:  
00663 *ENABLE  
00664 *DISABLE  
00665 */  
00666 #define CC4_EN DISABLE  
00667  
00668 /*****  
00669  
00670 /*options:  
00671 *ACTIVE_HIGH  
00672 *ACTIVE_LOW  
00673 */  
00674 #define CC4_POLARITY ACTIVE_HIGH  
00675  
00676 /*****  
00677 /* TIM1_BDTR configurations */  
00678 /*****  
00679  
00680 /*options:  
00681 *OC_DISABLE  
00682 *OC_ENABLE  
00683 */  
00684 #define OSSI_MODE OC_ENABLE  
00685  
00686 /*****  
00687  
00688 /*options:  
00689 *OC_DISABLE  
00690 *OC_ENABLE  
00691 */  
00692 #define OSSR_MODE OC_ENABLE  
00693  
00694 /*****  
00695  
00696 /*options:  
00697 *SW  
00698 *AUTO  
00699 */  
00700 #define AO_EN SW  
00701  
00702 /*****  
00703  
00704 /*options:  
00705 *OC_DISABLE  
00706 *OC_ENABLE  
00707 */  
00708 #define MO_EN OC_ENABLE  
00709  
00710 /*****  
00711  
00712  
00713  
00714  
00715  
00716  
00717  
00718  
00719  
00720  
00721  
00722
```

```

00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783 #endif // _TIM1_config_H

```

7.131 TIM1_interface.h

```

00001 /* FILENAME: TIM1_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 01/24/2023
00005 */
00006 #ifndef _TIM1_interface_H
00007 #define _TIM1_interface_H
00008
00009 /*****+
00010 /* Functions prototypes */
00011 *****/
00012
00013 void MTIM1_vInit( void ) ;
00014
00015 void MTIM1_vSetCounterValue( u16_t A_u16CNT_Value ) ;
00016
00017 void MTIM1_vSetPrescalerValue( u16_t A_u16PSC_Value ) ;
00018
00019 void MTIM1_vSetAutoReloadValue( u16_t A_u16ARR_Value ) ;

```

```

00020
00021 void MTIM1_vSetRepetitionValue( u8_t A_u8RCR_Value ) ;
00022
00023 void MTIM1_vSetCompareReg1Value( u16_t A_u16CCR1_Value ) ;
00024
00025 u16_t MTIM1_u16GetCaptureReg1Value( void ) ;
00026
00027 void MTIM1_vSetCompareReg2Value( u16_t A_u16CCR2_Value ) ;
00028
00029 u16_t MTIM1_u16GetCaptureReg2Value( void ) ;
00030
00031 void MTIM1_vSetCompareReg3Value( u16_t A_u16CCR3_Value ) ;
00032
00033 u16_t MTIM1_u16GetCaptureReg3Value( void ) ;
00034
00035 void MTIM1_vSetCompareReg4Value( u16_t A_u16CCR4_Value ) ;
00036
00037 u16_t MTIM1_u16GetCaptureReg4Value( void ) ;
00038
00039 void MTIM1_vGeneratePWM( u8_t A_u8T1CHx, u8_t A_u8PWM_Mode, u8_t A_u8CenterMode,
00040           u16_t A_u16PSC_Value, u16_t A_u16ARR_Value, u16_t A_u16CCRx_Value ) ;
00041
00042 void MTIM1_vReadPWM( void ) ;
00043
00044 void MTIM1_vEnableCounter( void ) ;
00045
00046 void MTIM1_vDisableCounter( void ) ;
00047
00048 /***** Interfacing macros *****/
00049 /* ***** */
00050 /***** */
00051
00052 #define T1CHx_PORT          GPIO_PORTA
00053 #define T1CH1_PIN           GPIOx_PIN8
00054 #define T1CH2_PIN           GPIOx_PIN9
00055 #define T1CH3_PIN           GPIOx_PIN10
00056 #define T1CH4_PIN           GPIOx_PIN11
00057
00058 #define PSC_VALUE           7
00059 #define ARR_VALUE           4000
00060 #define CR_VALUE            0
00061
00062 #define TIM1_CH1 1
00063 #define TIM1_CH2 2
00064 #define TIM1_CH3 3
00065 #define TIM1_CH4 4
00066
00067 #define FROZEN               1
00068 #define OC1REF_HIGH          2
00069 #define OC1REF_LOW           3
00070 #define OC1REF_TOGGLE         4
00071 #define OC1REF_FORCED_LOW    5
00072 #define OC1REF_FORCED_HIGH   6
00073 #define PWM1                 7
00074 #define PWM2                 8
00075
00076 #define EDGE                1
00077 #define CENTER1              2
00078 #define CENTER2              3
00079 #define CENTER3              4
00080
00081
00082 #endif // _TIM1_interface_H

```

7.132 TIM1_private.h

```

00001 /* FILENAME: TIM1_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Tue 01/24/2023
00005 */
00006 #ifndef _TIM1_private_H
00007 #define _TIM1_private_H
00008
00009 /***** Peripherals declaration *****/
00010 /* ***** */
00011 /***** */
00012
00013 typedef struct
00014 {
00020     volatile u32_t CRL           ;
00022

```

```

00023     volatile u32_t CR2          ;
00024
00025     volatile u32_t SMCR      ;
00026
00027     volatile u32_t DIER      ;
00028
00029     volatile u32_t SR        ;
00030
00031     volatile u32_t EGR       ;
00032
00033     volatile u32_t CCMR1    ;
00034
00035     volatile u32_t CCMR2    ;
00036
00037     volatile u32_t CCER      ;
00038
00039     volatile u32_t CNT       ;
00040
00041     volatile u32_t PSC       ;
00042
00043     volatile u32_t ARR       ;
00044
00045     volatile u32_t RCR       ;
00046
00047     volatile u32_t CCR1      ;
00048
00049     volatile u32_t CCR2      ;
00050
00051     volatile u32_t CCR3      ;
00052
00053     volatile u32_t CCR4      ;
00054
00055     volatile u32_t BDTR      ;
00056
00057     volatile u32_t DCR       ;
00058
00059     volatile u32_t DMAR      ;
00060
00061 } TIM1_MemoryMapType ;
00062
00063
00064 #define TIM1_BASE_ADDRESS 0x40010000
00065
00066 #define TIM1 ( ( volatile TIM1_MemoryMapType* ) (TIM1_BASE_ADDRESS) )
00067
00068 /***** TIM1_CRL register bits ****/
00069 /*          */
00070 /***** */
00071
00072 #define CKD1    9
00073 #define CKD0    8
00074 #define ARPE    7
00075 #define CMS1    6
00076 #define CMS0    5
00077 #define DIR     4
00078 #define OPM     3
00079 #define URS     2
00080 #define UDIS    1
00081 #define CEN     0
00082
00083 /***** TIM1_CR2 register bits ****/
00084 /*          */
00085 /***** */
00086
00087 #define OIS4    14
00088 #define OIS3N   13
00089 #define OIS3    12
00090 #define OIS2N   11
00091 #define OIS2    10
00092 #define OIS1N   9
00093 #define OIS1    8
00094 #define TI1S    7
00095 #define MMS2    6
00096 #define MMS1    5
00097 #define MMS0    4
00098 #define CCDS   3
00099 #define CCUS   2
00100 #define CCPC   0
00101
00102 /***** TIM1_SMCR register bits ****/
00103 /*          */
00104 /***** */
00105
00106 #define ETP     15
00107 #define ECE     14
00108 #define ETPS1   13
00109 #define ETPS0   12

```

```
00110 #define ETF3      11
00111 #define ETF2      10
00112 #define ETF1       9
00113 #define ETF0       8
00114 #define MSM        7
00115 #define TS2        6
00116 #define TS1        5
00117 #define TSO        4
00118 #define SMS2       2
00119 #define SMS1       1
00120 #define SMS0       0
00121
00122 /****** */
00123 /*          TIM1_DIER register bits           */
00124 /****** */
00125
00126 #define TDE        14
00127 #define COMDE     13
00128 #define CC4DE     12
00129 #define CC3DE     11
00130 #define CC2DE     10
00131 #define CC1DE     9
00132 #define UDE        8
00133 #define BIE        7
00134 #define TIE        6
00135 #define COMIE      5
00136 #define CC4IE      4
00137 #define CC3IE      3
00138 #define CC2IE      2
00139 #define CC1IE      1
00140 #define UIE        0
00141
00142 /****** */
00143 /*          TIM1_SR register bits           */
00144 /****** */
00145
00146 #define CC4OF      12
00147 #define CC3OF      11
00148 #define CC2OF      10
00149 #define CC1OF      9
00150 #define BIF         7
00151 #define TIF         6
00152 #define COMIF      5
00153 #define CC4IF      4
00154 #define CC3IF      3
00155 #define CC2IF      2
00156 #define CC1IF      1
00157 #define UIF         0
00158
00159 /****** */
00160 /*          TIM1_EGR register bits           */
00161 /****** */
00162
00163 #define BG         7
00164 #define TG         6
00165 #define COMG      5
00166 #define CC4G      4
00167 #define CC3G      3
00168 #define CC2G      2
00169 #define CC1G      1
00170 #define UG         0
00171
00172 /****** */
00173 /*          TIM1_CCMR1 register bits           */
00174 /****** */
00175
00176 // General:
00177 #define CC2S1      9
00178 #define CC2S0      8
00179 #define CC1S1      1
00180 #define CC1S0      0
00181
00182 // Output Capture mode:
00183 #define OC2CE      15
00184 #define OC2M2      14
00185 #define OC2M1      13
00186 #define OC2M0      12
00187 #define OC2PE      11
00188 #define OC2FE      10
00189 #define OC1CE      7
00190 #define OC1M2      6
00191 #define OC1M1      5
00192 #define OC1M0      4
00193 #define OC1PE      3
00194 #define OC1FE      2
00195
00196 // Input Capture mode:
```

```
00197 #define IC2F3    15
00198 #define IC2F2    14
00199 #define IC2F1    13
00200 #define IC2FO    12
00201 #define IC2PSC1   11
00202 #define IC2PSC0   10
00203 #define IC1F3     7
00204 #define IC1F2     6
00205 #define IC1F1     5
00206 #define IC1FO     4
00207 #define IC1PSC1   3
00208 #define IC1PSC0   2
00209
00210 /***** TIM1_CCMR2 register bits *****/
00211 /*          */
00212 /***** */
00213
00214 // General:
00215 #define CC4S1    9
00216 #define CC4S0    8
00217 #define CC3S1    1
00218 #define CC3S0    0
00219
00220 // For Output compare mode:
00221 #define OC4CE    15
00222 #define OC4M2    14
00223 #define OC4M1    13
00224 #define OC4M0    12
00225 #define OC4PE    11
00226 #define OC4FE    10
00227 #define OC3CE    7
00228 #define OC3M2    6
00229 #define OC3M1    5
00230 #define OC3M0    4
00231 #define OC3PE    3
00232 #define OC3FE    2
00233
00234 // For Input capture mode:
00235 #define IC4F3    15
00236 #define IC4F2    14
00237 #define IC4F1    13
00238 #define IC4FO    12
00239 #define IC4PSC1   11
00240 #define IC4PSC0   10
00241 #define IC3F3    7
00242 #define IC3F2    6
00243 #define IC3F1    5
00244 #define IC3FO    4
00245 #define IC3PSC1   3
00246 #define IC3PSC0   2
00247
00248 /***** TIM1_CCER2 register bits *****/
00249 /*          */
00250 /***** */
00251
00252 #define CC4P     13
00253 #define CC4E     12
00254 #define CC3NP    11
00255 #define CC3NE    10
00256 #define CC3P     9
00257 #define CC3E     8
00258 #define CC2NP    7
00259 #define CC2NE    6
00260 #define CC2P     5
00261 #define CC2E     4
00262 #define CC1NP    3
00263 #define CC1NE    2
00264 #define CC1P     1
00265 #define CC1E     0
00266
00267 /***** TIM1_BDTR register bits *****/
00268 /*          */
00269 /***** */
00270
00271 #define MOE      15
00272 #define AOE      14
00273 #define BKP      13
00274 #define BKE      12
00275 #define OSSR     11
00276 #define OSSI     10
00277 #define LOCK1    9
00278 #define LOCK0    8
00279 #define DTG7     7
00280 #define DTG6     6
00281 #define DTG5     5
00282 #define DTG4     4
00283 #define DTG3     3
```

```
00284 #define DTG2      2
00285 #define DTG1      1
00286 #define DTG0      0
00287
00288 /***** TIM1_DCR register bits *****/
00289 /*          Config.h Macros           */
00290 /***** */
00291
00292 #define DBL4      12
00293 #define DBL3      11
00294 #define DBL2      10
00295 #define DBL1      9
00296 #define DBL0      8
00297 #define DBA4      4
00298 #define DBA3      3
00299 #define DBA2      2
00300 #define DBA1      1
00301 #define DBA0      0
00302
00303 /***** */
00304 /*          Config.h Macros           */
00305 /***** */
00306
00307 #define ENABLE      2
00308 #define DISABLE     1
00309
00310 #define ANY_EVENT    1
00311 #define OVF_UDF     2
00312
00313 #define UP          1
00314 #define DOWN        2
00315
00316 #define EDGE         1
00317 #define CENTER1      2
00318 #define CENTER2      3
00319 #define CENTER3      4
00320
00321
00322 #define NOT_BUFFERED 1
00323 #define BUFFERED     2
00324
00325 #define ONE_TCK      1
00326 #define TWO_TCK      2
00327 #define FOUR_TCK     3
00328
00329 #define NOT_PRELOADED 1
00330 #define PRELOADED    2
00331
00332 #define COMG_ONLY     2
00333
00334 #define CCx_EVENT    1
00335 #define UPDATE_EVENT 2
00336
00337 #define RST          1
00338 #define UPDATE       3
00339 #define COMP_PULSE    4
00340 #define COMP1         5
00341 #define COMP2         6
00342 #define COMP3         7
00343 #define COMP4         8
00344
00345 #define CH1          1
00346 #define CH1_2_3      2
00347
00348 #define LOW          1
00349 #define HIGH         2
00350
00351 #define ENCODER1     3
00352 #define ENCODER2     4
00353 #define ENCODER3     5
00354 #define GATED         6
00355 #define TRIGGER       7
00356 #define EXT_CLK      8
00357
00358 #define ITR0         1
00359 #define ITR1         2
00360 #define ITR2         3
00361 #define ITR3         4
00362 #define TI1F_ED      5
00363 #define TI1FP1       6
00364 #define TI1FP2       7
00365 #define ETRF         8
00366
00367 #define NO_ACTION    1
00368 #define TRGI         2
00369
00370 #define NO_FLTR      1
```

```

00371 #define FCK_N2          2
00372 #define FCK_N4          3
00373 #define FCK_N8          4
00374 #define FDTS2_N6        5
00375 #define FDTS2_N8        6
00376 #define FDTS4_N6        7
00377 #define FDTS4_N8        8
00378 #define FDTS8_N6        9
00379 #define FDTS8_N8        10
00380 #define FDTS16_N5       11
00381 #define FDTS16_N6       12
00382 #define FDTS16_N8       13
00383 #define FDTS32_N5       14
00384 #define FDTS32_N6       15
00385 #define FDTS32_N8       16
00386
00387 #define OFF            1
00388 #define BY2            2
00389 #define BY4            3
00390 #define BY8            4
00391
00392 #define NON_INVERTED   1
00393 #define INVERTED        2
00394
00395 #define RE_INIT         2
00396
00397 #define CC_GENERATED    2
00398
00399 #define CC_UPDATED       2
00400
00401 #define TRG_ENABLED      2
00402
00403 #define BRK_GENERATED    2
00404
00405 #define CC1_OUTPUT        1
00406 #define CC1_INPUT_TI1     2
00407 #define CC1_INPUT_TI2     3
00408 #define CC1_INPUT_TRC     4
00409
00410 #define NO_PRESCALER     1
00411 #define EVERY_2_EVENTS    2
00412 #define EVERY_4_EVENTS    3
00413 #define EVERY_8_EVENTS    4
00414
00415 #define CC2_OUTPUT        1
00416 #define CC2_INPUT_TI1     2
00417 #define CC2_INPUT_TI2     3
00418 #define CC2_INPUT_TRC     4
00419
00420 #define NORMALLY          1
00421 #define INPUT_TRIGGER     2
00422
00423 #define FROZEN            1
00424 #define OC1REF_HIGH        2
00425 #define OC1REF_LOW         3
00426 #define OC1REF_TOGGLE       4
00427 #define OC1REF_FORCED_LOW   5
00428 #define OC1REF_FORCED_HIGH  6
00429 #define PWM1              7
00430 #define PWM2              8
00431
00432 #define OC1Ref_NOTEFFECTED 1
00433 #define OC1Ref_CLEARED     2
00434
00435 #define OC2Ref_NOTEFFECTED 1
00436 #define OC2Ref_CLEARED     2
00437
00438 #define OC2REF_HIGH        2
00439 #define OC2REF_LOW         3
00440 #define OC2REF_TOGGLE       4
00441 #define OC2REF_FORCED_LOW   5
00442 #define OC2REF_FORCED_HIGH  6
00443
00444 #define CC3_OUTPUT         1
00445 #define CC3_INPUT_TI3     2
00446 #define CC3_INPUT_TI4     3
00447 #define CC3_INPUT_TRC     4
00448
00449 #define CC4_OUTPUT         1
00450 #define CC4_INPUT_TI4     2
00451 #define CC4_INPUT_TI3     3
00452 #define CC4_INPUT_TRC     4
00453
00454 #define OC3REF_HIGH        2
00455 #define OC3REF_LOW         3
00456 #define OC3REF_TOGGLE       4
00457 #define OC3REF_FORCED_LOW   5

```

```

00458 #define OC3REF_FORCED_HIGH      6
00459
00460 #define OC4REF_HIGH           2
00461 #define OC4REF_LOW            3
00462 #define OC4REF_TOGGLE          4
00463 #define OC4REF_FORCED_LOW       5
00464 #define OC4REF_FORCED_HIGH      6
00465
00466 #define OC3Ref_NOTEFFECTED     1
00467 #define OC3Ref_CLEARED         2
00468
00469 #define OC4Ref_NOTEFFECTED     1
00470 #define OC4Ref_CLEARED         2
00471
00472 #define ACTIVE_HIGH           1
00473 #define ACTIVE_LOW            2
00474
00475 #define OC_DISABLE           1
00476 #define OC_ENABLE             2
00477
00478 #define SW                  1
00479 #define AUTO                2
00480
00481
00482 #endif // _TIM1_private_H

```

7.133 TIM1_program.c

```

00001 /* FILENAME: TIM1_program
00002 * Author:   Ali El Bana
00003 * Version:  V1.0
00004 * DATE:    Tue 01/24/2023
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010 #include "../../LIB/LSTD_TYPES.h"
00011 #include "../../LIB/LSTD_COMPILER.h"
00012 #include "../../LIB/LSTD_VALUES.h"
00013 #include "../../LIB/LSTD_BITMATH.h"
00014
00015 #include "../RCC/MRCC_interface.h"
00016 #include "../GPIO/GPIO_interface.h"
00017
00018 #include "TIM1_interface.h"
00019 #include "TIM1_private.h"
00020 #include "TIM1_config.h"
00021
00022 /***** Functions implementations *****/
00023 /*           Functions implementations           */
00024 /***** Functions implementations *****/
00025
00026 void MTIM1_vInit( void )
00027 {
00028
00029     // EN TIM1 CLK:
00030     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_TIM1EN ) ;
00031
00032     /*_____ CR1 _____*/
00033
00034     // EN/DIS update:
00035 #if UPDATE_CNTRL == ENABLE
00036
00037     CLR_BIT( TIM1->CR1, UDIS ) ;
00038
00039 #elif UPDATE_CNTRL == DISABLE
00040
00041     SET_BIT( TIM1->CR1, UDIS ) ;
00042
00043 #endif
00044
00045     // Update request source:
00046 #if UPDATE_SRC == OVF_UDF
00047
00048     SET_BIT( TIM1->CR1, URS ) ;
00049
00050 #elif UPDATE_SRC == ANY_EVENT
00051
00052     CLR_BIT( TIM1->CR1, URS ) ;
00053
00054 #endif
00055

```

```

00056      // One pulse mode:
00057 #if OP_MODE == ENABLE
00058
00059     SET_BIT( TIM1->CR1, OPM ) ;
00060
00061 #elif OP_MODE == DISABLE
00062
00063     CLR_BIT( TIM1->CR1, OPM ) ;
00064
00065 #endif
00066
00067      // Counting Direction:
00068 #if CNT_DIR == UP
00069
00070     CLR_BIT( TIM1->CR1, DIR ) ;
00071
00072 #elif CNT_DIR == DOWN
00073
00074     SET_BIT( TIM1->CR1, DIR ) ;
00075
00076 #endif
00077
00078      // Center-aligned mode selection:
00079 #if CMS == EDGE
00080
00081     CLR_BIT( TIM1->CR1, CMS0 ) ;
00082     CLR_BIT( TIM1->CR1, CMS1 ) ;
00083
00084 #elif CMS == CENTER1
00085
00086     SET_BIT( TIM1->CR1, CMS0 ) ;
00087     CLR_BIT( TIM1->CR1, CMS1 ) ;
00088
00089 #elif CMS == CENTER2
00090
00091     CLR_BIT( TIM1->CR1, CMS0 ) ;
00092     SET_BIT( TIM1->CR1, CMS1 ) ;
00093
00094 #elif CMS == CENTER3
00095
00096     SET_BIT( TIM1->CR1, CMS0 ) ;
00097     SET_BIT( TIM1->CR1, CMS1 ) ;
00098
00099 #endif
00100
00101      // Auto-reload preload enable:
00102 #if ARP_EN == NOT_BUFFERED
00103
00104     CLR_BIT( TIM1->CR1, ARPE ) ;
00105
00106 #elif ARP_EN == BUFFERED
00107
00108     SET_BIT( TIM1->CR1, ARPE ) ;
00109
00110 #endif
00111
00112      // Clock division:
00113 #if CKD == ONE_TCK
00114
00115     CLR_BIT( TIM1->CR1, CKD0 ) ;
00116     CLR_BIT( TIM1->CR1, CKD1 ) ;
00117
00118 #elif CKD == TWO_TCK
00119
00120     SET_BIT( TIM1->CR1, CKD0 ) ;
00121     CLR_BIT( TIM1->CR1, CKD1 ) ;
00122
00123 #elif CKD == FOUR_TCK
00124
00125     CLR_BIT( TIM1->CR1, CKD0 ) ;
00126     SET_BIT( TIM1->CR1, CKD1 ) ;
00127
00128 #endif
00129
00130     /*_____CR2_____*/
00131
00132      // Capture/compare preloaded control:
00133 #if CCP_CTRL == NOT_PRELOADED
00134
00135     CLR_BIT( TIM1->CR2, CCPC ) ;
00136
00137 #elif CCP_CTRL == PRELOADED
00138
00139     SET_BIT( TIM1->CR2, CCPC ) ;
00140
00141 #endif
00142

```

```
00143     // Capture/compare control update selection:  
00144 #if CCU_SLCT == COMG_ONLY  
00145  
00146     CLR_BIT( TIM1->CR2, CCUS ) ;  
00147  
00148 #elif CCU_SLCT == ANY_EVENT  
00149  
00150     SET_BIT( TIM1->CR2, CCUS ) ;  
00151  
00152 #endif  
00153  
00154     // Capture/compare DMA selection:  
00155 #if CCD_SLCT == CCX_EVENT  
00156  
00157     CLR_BIT( TIM1->CR2, CCDS ) ;  
00158  
00159 #elif CCD_SLCT == UPDATE_EVENT  
00160  
00161     SET_BIT( TIM1->CR2, CCDS ) ;  
00162  
00163 #endif  
00164  
00165     // Master mode selection:  
00166 #if MMS == RST  
00167  
00168     CLR_BIT( TIM1->CR2, MMS0 ) ;  
00169     CLR_BIT( TIM1->CR2, MMS1 ) ;  
00170     CLR_BIT( TIM1->CR2, MMS2 ) ;  
00171  
00172 #elif MMS == ENABLE  
00173  
00174     SET_BIT( TIM1->CR2, MMS0 ) ;  
00175     CLR_BIT( TIM1->CR2, MMS1 ) ;  
00176     CLR_BIT( TIM1->CR2, MMS2 ) ;  
00177  
00178 #elif MMS == UPDATE  
00179  
00180     CLR_BIT( TIM1->CR2, MMS0 ) ;  
00181     SET_BIT( TIM1->CR2, MMS1 ) ;  
00182     CLR_BIT( TIM1->CR2, MMS2 ) ;  
00183  
00184 #elif MMS == COMP_PULSE  
00185  
00186     SET_BIT( TIM1->CR2, MMS0 ) ;  
00187     SET_BIT( TIM1->CR2, MMS1 ) ;  
00188     CLR_BIT( TIM1->CR2, MMS2 ) ;  
00189  
00190 #elif MMS == COMP1  
00191  
00192     CLR_BIT( TIM1->CR2, MMS0 ) ;  
00193     CLR_BIT( TIM1->CR2, MMS1 ) ;  
00194     SET_BIT( TIM1->CR2, MMS2 ) ;  
00195  
00196 #elif MMS == COMP2  
00197  
00198     SET_BIT( TIM1->CR2, MMS0 ) ;  
00199     CLR_BIT( TIM1->CR2, MMS1 ) ;  
00200     SET_BIT( TIM1->CR2, MMS2 ) ;  
00201  
00202 #elif MMS == COMP3  
00203  
00204     CLR_BIT( TIM1->CR2, MMS0 ) ;  
00205     SET_BIT( TIM1->CR2, MMS1 ) ;  
00206     SET_BIT( TIM1->CR2, MMS2 ) ;  
00207  
00208 #elif MMS == COMP4  
00209  
00210     SET_BIT( TIM1->CR2, MMS0 ) ;  
00211     SET_BIT( TIM1->CR2, MMS1 ) ;  
00212     SET_BIT( TIM1->CR2, MMS2 ) ;  
00213  
00214 #endif  
00215  
00216     // TI1 selection:  
00217 #if TI1_SLCT == CH1  
00218  
00219     CLR_BIT( TIM1->CR2, TI1S ) ;  
00220  
00221 #elif TI1_SLCT == CH1_2_3  
00222  
00223     SET_BIT( TIM1->CR2, TI1S ) ;  
00224  
00225 #endif  
00226  
00227     // Output Idle state 1 (OC1 output):  
00228 #if OC1_IDLE_STATE == LOW  
00229
```

```

00230     CLR_BIT( TIM1->CR2, OIS1 ) ;
00231
00232 #elif OC1_IDLE_STATE == HIGH
00233
00234     SET_BIT( TIM1->CR2, OIS1 ) ;
00235
00236 #endif
00237
00238 // Output Idle state 1 (OC1N output):
00239 #if OC1N_IDLE_STATE == LOW
00240
00241     CLR_BIT( TIM1->CR2, OIS1N ) ;
00242
00243 #elif OC1N_IDLE_STATE == HIGH
00244
00245     SET_BIT( TIM1->CR2, OIS1N ) ;
00246
00247 #endif
00248
00249 // Output Idle state 2 (OC2 output):
00250 #if OC2_IDLE_STATE == LOW
00251
00252     CLR_BIT( TIM1->CR2, OIS2 ) ;
00253
00254 #elif OC2_IDLE_STATE == HIGH
00255
00256     SET_BIT( TIM1->CR2, OIS2 ) ;
00257
00258 #endif
00259
00260 // Output Idle state 2 (OC2N output):
00261 #if OC2N_IDLE_STATE == LOW
00262
00263     CLR_BIT( TIM1->CR2, OIS2N ) ;
00264
00265 #elif OC2N_IDLE_STATE == HIGH
00266
00267     SET_BIT( TIM1->CR2, OIS2N ) ;
00268
00269 #endif
00270
00271 // Output Idle state 3 (OC3 output):
00272 #if OC3_IDLE_STATE == LOW
00273
00274     CLR_BIT( TIM1->CR2, OIS3 ) ;
00275
00276 #elif OC3_IDLE_STATE == HIGH
00277
00278     SET_BIT( TIM1->CR2, OIS3 ) ;
00279
00280 #endif
00281
00282 // Output Idle state 3 (OC3N output):
00283 #if OC3N_IDLE_STATE == LOW
00284
00285     CLR_BIT( TIM1->CR2, OIS3N ) ;
00286
00287 #elif OC3N_IDLE_STATE == HIGH
00288
00289     SET_BIT( TIM1->CR2, OIS3N ) ;
00290
00291 #endif
00292
00293 // Output Idle state 4 (OC4 output):
00294 #if OC3_IDLE_STATE == LOW
00295
00296     CLR_BIT( TIM1->CR2, OIS4 ) ;
00297
00298 #elif OC3_IDLE_STATE == HIGH
00299
00300     SET_BIT( TIM1->CR2, OIS4 ) ;
00301
00302 #endif
00303
00304 /* _____SMCR_____ */
00305
00306 // Slave mode selection:
00307 #if SMS == DISABLE
00308
00309     CLR_BIT( TIM1->SMCR, SMS0 ) ;
00310     CLR_BIT( TIM1->SMCR, SMS1 ) ;
00311     CLR_BIT( TIM1->SMCR, SMS2 ) ;
00312
00313 #elif SMS == ENCODER1
00314
00315     SET_BIT( TIM1->SMCR, SMS0 ) ;
00316     CLR_BIT( TIM1->SMCR, SMS1 ) ;

```

```
00317     CLR_BIT( TIM1->SMCR, SMS2 ) ;
00318
00319 #elif SMS == ENCODER2
00320
00321     CLR_BIT( TIM1->SMCR, SMS0 ) ;
00322     SET_BIT( TIM1->SMCR, SMS1 ) ;
00323     CLR_BIT( TIM1->SMCR, SMS2 ) ;
00324
00325 #elif SMS == ENCODER3
00326
00327     SET_BIT( TIM1->SMCR, SMS0 ) ;
00328     SET_BIT( TIM1->SMCR, SMS1 ) ;
00329     CLR_BIT( TIM1->SMCR, SMS2 ) ;
00330
00331 #elif SMS == RST
00332
00333     CLR_BIT( TIM1->SMCR, SMS0 ) ;
00334     CLR_BIT( TIM1->SMCR, SMS1 ) ;
00335     SET_BIT( TIM1->SMCR, SMS2 ) ;
00336
00337 #elif SMS == GATED
00338
00339     SET_BIT( TIM1->SMCR, SMS0 ) ;
00340     CLR_BIT( TIM1->SMCR, SMS1 ) ;
00341     SET_BIT( TIM1->SMCR, SMS2 ) ;
00342
00343 #elif SMS == TRIGGER
00344
00345     CLR_BIT( TIM1->SMCR, SMS0 ) ;
00346     SET_BIT( TIM1->SMCR, SMS1 ) ;
00347     SET_BIT( TIM1->SMCR, SMS2 ) ;
00348
00349 #elif SMS == EXT_CLK
00350
00351     SET_BIT( TIM1->SMCR, SMS0 ) ;
00352     SET_BIT( TIM1->SMCR, SMS1 ) ;
00353     SET_BIT( TIM1->SMCR, SMS2 ) ;
00354
00355 #endif
00356
00357 // Trigger selection:
00358 #if TS == ITR0
00359
00360     CLR_BIT( TIM1->SMCR, TS0 ) ;
00361     CLR_BIT( TIM1->SMCR, TS1 ) ;
00362     CLR_BIT( TIM1->SMCR, TS2 ) ;
00363
00364 #elif TS == ITR1
00365
00366     SET_BIT( TIM1->SMCR, TS0 ) ;
00367     CLR_BIT( TIM1->SMCR, TS1 ) ;
00368     CLR_BIT( TIM1->SMCR, TS2 ) ;
00369
00370 #elif TS == ITR2
00371
00372     CLR_BIT( TIM1->SMCR, TS0 ) ;
00373     SET_BIT( TIM1->SMCR, TS1 ) ;
00374     CLR_BIT( TIM1->SMCR, TS2 ) ;
00375
00376 #elif TS == ITR3
00377
00378     SET_BIT( TIM1->SMCR, TS0 ) ;
00379     SET_BIT( TIM1->SMCR, TS1 ) ;
00380     CLR_BIT( TIM1->SMCR, TS2 ) ;
00381
00382 #elif TS == TI1F_ED
00383
00384     CLR_BIT( TIM1->SMCR, TS0 ) ;
00385     CLR_BIT( TIM1->SMCR, TS1 ) ;
00386     SET_BIT( TIM1->SMCR, TS2 ) ;
00387
00388 #elif TS == TI1FP1
00389
00390     SET_BIT( TIM1->SMCR, TS0 ) ;
00391     CLR_BIT( TIM1->SMCR, TS1 ) ;
00392     SET_BIT( TIM1->SMCR, TS2 ) ;
00393
00394 #elif TS == TI1FP2
00395
00396     CLR_BIT( TIM1->SMCR, TS0 ) ;
00397     SET_BIT( TIM1->SMCR, TS1 ) ;
00398     SET_BIT( TIM1->SMCR, TS2 ) ;
00399
00400 #elif TS == ETRF
00401
00402     SET_BIT( TIM1->SMCR, TS0 ) ;
00403     SET_BIT( TIM1->SMCR, TS1 ) ;
```

```

00404     SET_BIT( TIM1->SMCR, TS2 ) ;
00405
00406 #endif
00407
00408 // Master/slave mode:
00409 #if MSTR_SLAVE_MODE == NO_ACTION
00410
00411     CLR_BIT( TIM1->SMCR, MSM ) ;
00412
00413 #elif MSTR_SLAVE_MODE == TRGI
00414
00415     SET_BIT( TIM1->SMCR, MSM ) ;
00416
00417 #endif
00418
00419 // External trigger prescaler:
00420 #if ETPS == OFF
00421
00422     CLR_BIT( TIM1->SMCR, ETPS0 ) ;
00423     CLR_BIT( TIM1->SMCR, ETPS1 ) ;
00424
00425 #elif ETPS == BY2
00426
00427     SET_BIT( TIM1->SMCR, ETPS0 ) ;
00428     CLR_BIT( TIM1->SMCR, ETPS1 ) ;
00429
00430 #elif ETPS == BY4
00431
00432     CLR_BIT( TIM1->SMCR, ETPS0 ) ;
00433     SET_BIT( TIM1->SMCR, ETPS1 ) ;
00434
00435 #elif ETPS == BY8
00436
00437     SET_BIT( TIM1->SMCR, ETPS0 ) ;
00438     SET_BIT( TIM1->SMCR, ETPS1 ) ;
00439
00440 #endif
00441
00442 // External clock enable:
00443 #if EXT_CLK_EN == ENABLE
00444
00445     SET_BIT( TIM1->SMCR, ECE ) ;
00446
00447 #elif EXT_CLK_EN == DISABLE
00448
00449     CLR_BIT( TIM1->SMCR, ECE ) ;
00450
00451 #endif
00452
00453 // External trigger polarity:
00454 #if EXTP == INVERTED
00455
00456     SET_BIT( TIM1->SMCR, ETP ) ;
00457
00458 #elif EXTP == NON_INVERTED
00459
00460     CLR_BIT( TIM1->SMCR, ETP ) ;
00461
00462 #endif
00463
00464 /*_____EGR_____*/
00465
00466 // Update generation:
00467 #if UP_GEN == RE_INIT
00468
00469     SET_BIT( TIM1->EGR, UG ) ;
00470
00471 #elif UP_GEN == NO_ACTION
00472
00473     CLR_BIT( TIM1->EGR, UG ) ;
00474
00475 #endif
00476
00477 // Capture/Compare 1 generation:
00478 #if CC1_GEN == CC_GENERATED
00479
00480     SET_BIT( TIM1->EGR, CC1G ) ;
00481
00482 #elif CC1_GEN == NO_ACTION
00483
00484     CLR_BIT( TIM1->EGR, CC1G ) ;
00485
00486 #endif
00487
00488 // Capture/Compare 2 generation:
00489 #if CC2_GEN == CC_GENERATED
00490

```

```
00491     SET_BIT( TIM1->EGR, CC2G ) ;
00492
00493 #elif CC2_GEN == NO_ACTION
00494
00495     CLR_BIT( TIM1->EGR, CC2G ) ;
00496
00497 #endif
00498
00499 // Capture/Compare 3 generation:
00500 #if CC3_GEN == CC_GENERATED
00501
00502     SET_BIT( TIM1->EGR, CC3G ) ;
00503
00504 #elif CC3_GEN == NO_ACTION
00505
00506     CLR_BIT( TIM1->EGR, CC3G ) ;
00507
00508 #endif
00509
00510 // Capture/Compare 4 generation:
00511 #if CC4_GEN == CC_GENERATED
00512
00513     SET_BIT( TIM1->EGR, CC4G ) ;
00514
00515 #elif CC4_GEN == NO_ACTION
00516
00517     CLR_BIT( TIM1->EGR, CC4G ) ;
00518
00519 #endif
00520
00521 // Capture/Compare control update generation:
00522 #if COM_GEN == CC_UPDATED
00523
00524     SET_BIT( TIM1->EGR, COMG ) ;
00525
00526 #elif COM_GEN == NO_ACTION
00527
00528     CLR_BIT( TIM1->EGR, COMG ) ;
00529
00530 #endif
00531
00532 // Trigger generation:
00533 #if TRG_GEN == TRG_ENABLED
00534
00535     SET_BIT( TIM1->EGR, TG ) ;
00536
00537 #elif TRG_GEN == NO_ACTION
00538
00539     CLR_BIT( TIM1->EGR, TG ) ;
00540
00541 #endif
00542
00543 // Break generation:
00544 #if BRK_GEN == BRK_GENERATED
00545
00546     SET_BIT( TIM1->EGR, BG ) ;
00547
00548 #elif BRK_GEN == NO_ACTION
00549
00550     CLR_BIT( TIM1->EGR, BG ) ;
00551
00552 #endif
00553
00554 /*_____CCMR1_____*/
00555
00556 // Capture/Compare 1 Selection:
00557 #if CC1S == CC1_OUTPUT
00558
00559     CLR_BIT( TIM1->CCMR1, CC1S0 ) ;
00560     CLR_BIT( TIM1->CCMR1, CC1S1 ) ;
00561
00562 #elif CC1S == CC1_INPUT_T1
00563
00564     SET_BIT( TIM1->CCMR1, CC1S0 ) ;
00565     CLR_BIT( TIM1->CCMR1, CC1S1 ) ;
00566
00567 #elif CC1S == CC1_INPUT_T2
00568
00569     CLR_BIT( TIM1->CCMR1, CC1S0 ) ;
00570     SET_BIT( TIM1->CCMR1, CC1S1 ) ;
00571
00572 #elif CC1S == CC1_INPUT_TRC
00573
00574     SET_BIT( TIM1->CCMR1, CC1S0 ) ;
00575     SET_BIT( TIM1->CCMR1, CC1S1 ) ;
00576
00577 #endif
```

```

00578     // Capture/Compare 2 Selection:
00579 #if CC2S == CC2_OUTPUT
00580     CLR_BIT( TIM1->CCMR1, CC2S0 ) ;
00581     CLR_BIT( TIM1->CCMR1, CC2S1 ) ;
00582
00583 #elif CC2S == CC2_INPUT_TI1
00584     SET_BIT( TIM1->CCMR1, CC2S0 ) ;
00585     CLR_BIT( TIM1->CCMR1, CC2S1 ) ;
00586
00587 #elif CC2S == CC2_INPUT_TI2
00588     CLR_BIT( TIM1->CCMR1, CC2S0 ) ;
00589     SET_BIT( TIM1->CCMR1, CC2S1 ) ;
00590
00591 #elif CC2S == CC2_INPUT_TRC
00592     SET_BIT( TIM1->CCMR1, CC2S0 ) ;
00593     SET_BIT( TIM1->CCMR1, CC2S1 ) ;
00594
00595 #endif
00596
00597 // Input capture 1 prescaler:
00598 #if IC1PSC == NO_PRESCALER
00599
00600     CLR_BIT( TIM1->CCMR1, IC1PSC0 ) ;
00601     CLR_BIT( TIM1->CCMR1, IC1PSC1 ) ;
00602
00603 #elif IC1PSC == EVERY_2_EVENTS
00604     SET_BIT( TIM1->CCMR1, IC1PSC0 ) ;
00605     CLR_BIT( TIM1->CCMR1, IC1PSC1 ) ;
00606
00607 #elif IC1PSC == EVERY_4_EVENTS
00608     CLR_BIT( TIM1->CCMR1, IC1PSC0 ) ;
00609     SET_BIT( TIM1->CCMR1, IC1PSC1 ) ;
00610
00611 #elif IC1PSC == EVERY_8_EVENTS
00612     SET_BIT( TIM1->CCMR1, IC1PSC0 ) ;
00613     SET_BIT( TIM1->CCMR1, IC1PSC1 ) ;
00614
00615 #endif
00616
00617 // Input capture 2 prescaler:
00618 #if IC2PSC == NO_PRESCALER
00619
00620     CLR_BIT( TIM1->CCMR1, IC2PSC0 ) ;
00621     CLR_BIT( TIM1->CCMR1, IC2PSC1 ) ;
00622
00623 #endif
00624
00625
00626 // Output Compare 1 fast enable:
00627 #if OC1F_EN == NORMALLY
00628
00629     CLR_BIT( TIM1->CCMR1, OC1FE ) ;
00630
00631 #elif OC1F_EN == INPUT_TRIGGER
00632     SET_BIT( TIM1->CCMR1, OC1FE ) ;
00633
00634 #endif
00635
00636 // Output Compare 1 preload enable:
00637 #if OC1P_EN == ENABLE
00638
00639     SET_BIT( TIM1->CCMR1, OC1PE ) ;
00640
00641 #endif
00642
00643 #endif
00644
00645 // Output Compare 1 fast enable:
00646 #if OC1F_EN == NORMALLY
00647
00648     CLR_BIT( TIM1->CCMR1, OC1FE ) ;
00649
00650 #elif OC1F_EN == INPUT_TRIGGER
00651     SET_BIT( TIM1->CCMR1, OC1FE ) ;
00652
00653 #endif
00654
00655 // Output Compare 1 preload enable:
00656 #if OC1P_EN == ENABLE
00657
00658 #endif
00659
00660 // Output Compare 1 preload enable:
00661 #if OC1P_EN == ENABLE
00662
00663     SET_BIT( TIM1->CCMR1, OC1PE ) ;
00664

```

```
00665 #elif OC1P_EN == DISABLE
00666     CLR_BIT( TIM1->CCMR1, OC1PE ) ;
00668 #endif
00670 // Output Compare 1 mode:
00672 #if OC1M == FROZEN
00673     CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
00675     CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
00676     CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
00677
00678 #elif OC1M == OC1REF_HIGH
00679
00680     SET_BIT( TIM1->CCMR1, OC1M0 ) ;
00681     CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
00682     CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
00683
00684 #elif OC1M == OC1REF_LOW
00685
00686     CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
00687     SET_BIT( TIM1->CCMR1, OC1M1 ) ;
00688     CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
00689
00690 #elif OC1M == OC1REF_TOGGLE
00691
00692     SET_BIT( TIM1->CCMR1, OC1M0 ) ;
00693     SET_BIT( TIM1->CCMR1, OC1M1 ) ;
00694     CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
00695
00696 #elif OC1M == OC1REF_FORCED_LOW
00697
00698     CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
00699     CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
00700     SET_BIT( TIM1->CCMR1, OC1M2 ) ;
00701
00702 #elif OC1M == OC1REF_FORCED_HIGH
00703
00704     SET_BIT( TIM1->CCMR1, OC1M0 ) ;
00705     CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
00706     SET_BIT( TIM1->CCMR1, OC1M2 ) ;
00707
00708 #elif OC1M == PWM1
00709
00710     CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
00711     SET_BIT( TIM1->CCMR1, OC1M1 ) ;
00712     SET_BIT( TIM1->CCMR1, OC1M2 ) ;
00713
00714 #elif OC1M == PWM2
00715
00716     SET_BIT( TIM1->CCMR1, OC1M0 ) ;
00717     SET_BIT( TIM1->CCMR1, OC1M1 ) ;
00718     SET_BIT( TIM1->CCMR1, OC1M2 ) ;
00719
00720 #endif
00721 // Output Compare 1 clear enable:
00723 #if OC1C_EN == OC1Ref_NOTEFFECTED
00724
00725     CLR_BIT( TIM1->CCMR1, OC1CE ) ;
00726
00727 #elif OC1C_EN == OC1Ref_CLEARED
00728
00729     SET_BIT( TIM1->CCMR1, OC1CE ) ;
00730
00731 #endif
00732
00733 // Output Compare 2 fast enable:
00734 #if OC2F_EN == NORMALLY
00735
00736     CLR_BIT( TIM1->CCMR1, OC2FE ) ;
00737
00738 #elif OC2F_EN == INPUT_TRIGGER
00739
00740     SET_BIT( TIM1->CCMR1, OC2FE ) ;
00741
00742 #endif
00743
00744 // Output Compare 2 preload enable:
00745 #if OC2P_EN == ENABLE
00746
00747     SET_BIT( TIM1->CCMR1, OC2PE ) ;
00748
00749 #elif OC2P_EN == DISABLE
00750
00751     CLR_BIT( TIM1->CCMR1, OC2PE ) ;
```

```

00752
00753 #endif
00754
00755 // Output Compare 2 mode:
00756 #if OC2M == FROZEN
00757
00758     CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
00759     CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
00760     CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
00761
00762 #elif OC2M == OC2REF_HIGH
00763
00764     SET_BIT( TIM1->CCMR1, OC2M0 ) ;
00765     CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
00766     CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
00767
00768 #elif OC2M == OC2REF_LOW
00769
00770     CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
00771     SET_BIT( TIM1->CCMR1, OC2M1 ) ;
00772     CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
00773
00774 #elif OC2M == OC2REF_TOGGLE
00775
00776     SET_BIT( TIM1->CCMR1, OC2M0 ) ;
00777     SET_BIT( TIM1->CCMR1, OC2M1 ) ;
00778     CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
00779
00780 #elif OC2M == OC2REF_FORCED_LOW
00781
00782     CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
00783     CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
00784     SET_BIT( TIM1->CCMR1, OC2M2 ) ;
00785
00786 #elif OC2M == OC2REF_FORCED_HIGH
00787
00788     SET_BIT( TIM1->CCMR1, OC2M0 ) ;
00789     CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
00790     SET_BIT( TIM1->CCMR1, OC2M2 ) ;
00791
00792 #elif OC2M == PWM1
00793
00794     CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
00795     SET_BIT( TIM1->CCMR1, OC2M1 ) ;
00796     SET_BIT( TIM1->CCMR1, OC2M2 ) ;
00797
00798 #elif OC2M == PWM2
00799
00800     SET_BIT( TIM1->CCMR1, OC2M0 ) ;
00801     SET_BIT( TIM1->CCMR1, OC2M1 ) ;
00802     SET_BIT( TIM1->CCMR1, OC2M2 ) ;
00803
00804 #endif
00805
00806 // Output Compare 2 clear enable:
00807 #if OC2C_EN == OC2Ref_NOTEFFECTED
00808
00809     CLR_BIT( TIM1->CCMR1, OC2CE ) ;
00810
00811 #elif OC2C_EN == OC2Ref_CLEARED
00812
00813     SET_BIT( TIM1->CCMR1, OC2CE ) ;
00814
00815 #endif
00816
00817 /*_____CCMR2_____*/
00818
00819 // Capture/Compare 3 Selection:
00820 #if CC3S == CC3_OUTPUT
00821
00822     CLR_BIT( TIM1->CCMR2, CC3S0 ) ;
00823     CLR_BIT( TIM1->CCMR2, CC3S1 ) ;
00824
00825 #elif CC3S == CC3_INPUT_TI1
00826
00827     SET_BIT( TIM1->CCMR2, CC3S0 ) ;
00828     CLR_BIT( TIM1->CCMR2, CC3S1 ) ;
00829
00830 #elif CC3S == CC3_INPUT_TI2
00831
00832     CLR_BIT( TIM1->CCMR2, CC3S0 ) ;
00833     SET_BIT( TIM1->CCMR2, CC3S1 ) ;
00834
00835 #elif CC3S == CC3_INPUT_TRC
00836
00837     SET_BIT( TIM1->CCMR2, CC3S0 ) ;
00838     SET_BIT( TIM1->CCMR2, CC3S1 ) ;

```

```
00839
00840 #endif
00841
00842 // Capture/Compare 4 Selection:
00843 #if CC4S == CC4_OUTPUT
00844
00845 CLR_BIT( TIM1->CCMR2, CC4S0 ) ;
00846 CLR_BIT( TIM1->CCMR2, CC4S1 ) ;
00847
00848 #elif CC4S == CC4_INPUT_T11
00849
00850 SET_BIT( TIM1->CCMR2, CC4S0 ) ;
00851 CLR_BIT( TIM1->CCMR2, CC4S1 ) ;
00852
00853 #elif CC4S == CC4_INPUT_T12
00854
00855 CLR_BIT( TIM1->CCMR2, CC4S0 ) ;
00856 SET_BIT( TIM1->CCMR2, CC4S1 ) ;
00857
00858 #elif CC4S == CC4_INPUT_TRC
00859
00860 SET_BIT( TIM1->CCMR2, CC4S0 ) ;
00861 SET_BIT( TIM1->CCMR2, CC4S1 ) ;
00862
00863 #endif
00864
00865 // Input capture 3 prescaler:
00866 #if IC3PSC == NO_PRESCALER
00867
00868 CLR_BIT( TIM1->CCMR2, IC3PSC0 ) ;
00869 CLR_BIT( TIM1->CCMR2, IC3PSC1 ) ;
00870
00871 #elif IC3PSC == EVERY_2_EVENTS
00872
00873 SET_BIT( TIM1->CCMR2, IC3PSC0 ) ;
00874 CLR_BIT( TIM1->CCMR2, IC3PSC1 ) ;
00875
00876 #elif IC3PSC == EVERY_4_EVENTS
00877
00878 CLR_BIT( TIM1->CCMR2, IC3PSC0 ) ;
00879 SET_BIT( TIM1->CCMR2, IC3PSC1 ) ;
00880
00881 #elif IC3PSC == EVERY_8_EVENTS
00882
00883 SET_BIT( TIM1->CCMR2, IC3PSC0 ) ;
00884 SET_BIT( TIM1->CCMR2, IC3PSC1 ) ;
00885
00886 #endif
00887
00888 // Input capture 4 prescaler:
00889 #if IC4PSC == NO_PRESCALER
00890
00891 CLR_BIT( TIM1->CCMR2, IC4PSC0 ) ;
00892 CLR_BIT( TIM1->CCMR2, IC4PSC1 ) ;
00893
00894 #elif IC4PSC == EVERY_2_EVENTS
00895
00896 SET_BIT( TIM1->CCMR2, IC4PSC0 ) ;
00897 CLR_BIT( TIM1->CCMR2, IC4PSC1 ) ;
00898
00899 #elif IC4PSC == EVERY_4_EVENTS
00900
00901 CLR_BIT( TIM1->CCMR2, IC4PSC0 ) ;
00902 SET_BIT( TIM1->CCMR2, IC4PSC1 ) ;
00903
00904 #elif IC4PSC == EVERY_8_EVENTS
00905
00906 SET_BIT( TIM1->CCMR2, IC4PSC0 ) ;
00907 SET_BIT( TIM1->CCMR2, IC4PSC1 ) ;
00908
00909 #endif
00910
00911 // Output Compare 3 fast enable:
00912 #if OC3F_EN == NORMALLY
00913
00914 CLR_BIT( TIM1->CCMR2, OC3FE ) ;
00915
00916 #elif OC3F_EN == INPUT_TRIGGER
00917
00918 SET_BIT( TIM1->CCMR2, OC3FE ) ;
00919
00920 #endif
00921
00922 // Output Compare 3 preload enable:
00923 #if OC1P_EN == ENABLE
00924
00925 SET_BIT( TIM1->CCMR2, OC3PE ) ;
```

```

00926
00927 #elif OC1P_EN == DISABLE
00928
00929     CLR_BIT( TIM1->CCMR2, OC3PE ) ;
00930
00931 #endif
00932
00933 // Output Compare 3 mode:
00934 #if OC3M == FROZEN
00935
00936     CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
00937     CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
00938     CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
00939
00940 #elif OC3M == OC3REF_HIGH
00941
00942     SET_BIT( TIM1->CCMR2, OC3M0 ) ;
00943     CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
00944     CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
00945
00946 #elif OC3M == OC3REF_LOW
00947
00948     CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
00949     SET_BIT( TIM1->CCMR2, OC3M1 ) ;
00950     CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
00951
00952 #elif OC3M == OC3REF_TOGGLE
00953
00954     SET_BIT( TIM1->CCMR2, OC3M0 ) ;
00955     SET_BIT( TIM1->CCMR2, OC3M1 ) ;
00956     CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
00957
00958 #elif OC3M == OC3REF_FORCED_LOW
00959
00960     CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
00961     CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
00962     SET_BIT( TIM1->CCMR2, OC3M2 ) ;
00963
00964 #elif OC3M == OC3REF_FORCED_HIGH
00965
00966     SET_BIT( TIM1->CCMR2, OC3M0 ) ;
00967     CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
00968     SET_BIT( TIM1->CCMR2, OC3M2 ) ;
00969
00970 #elif OC3M == PWM1
00971
00972     CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
00973     SET_BIT( TIM1->CCMR2, OC3M1 ) ;
00974     SET_BIT( TIM1->CCMR2, OC3M2 ) ;
00975
00976 #elif OC3M == PWM2
00977
00978     SET_BIT( TIM1->CCMR2, OC3M0 ) ;
00979     SET_BIT( TIM1->CCMR2, OC3M1 ) ;
00980     SET_BIT( TIM1->CCMR2, OC3M2 ) ;
00981
00982 #endif
00983
00984 // Output Compare 3 clear enable:
00985 #if OC3C_EN == OC3Ref_NOTEFFECTED
00986
00987     CLR_BIT( TIM1->CCMR2, OC3CE ) ;
00988
00989 #elif OC3C_EN == OC3Ref_CLEARED
00990
00991     SET_BIT( TIM1->CCMR2, OC3CE ) ;
00992
00993 #endif
00994
00995 // Output Compare 4 fast enable:
00996 #if OC4F_EN == NORMALLY
00997
00998     CLR_BIT( TIM1->CCMR2, OC4FE ) ;
00999
01000 #elif OC4F_EN == INPUT_TRIGGER
01001
01002     SET_BIT( TIM1->CCMR2, OC4FE ) ;
01003
01004 #endif
01005
01006 // Output Compare 4 preload enable:
01007 #if OC4P_EN == ENABLE
01008
01009     SET_BIT( TIM1->CCMR2, OC4PE ) ;
01010
01011 #elif OC4P_EN == DISABLE
01012

```

```
01013     CLR_BIT( TIM1->CCMR2, OC4PE ) ;
01014
01015 #endif
01016
01017 // Output Compare 4 mode:
01018 #if OC4M == FROZEN
01019
01020     CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01021     CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01022     CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01023
01024 #elif OC4M == OC4REF_HIGH
01025
01026     SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01027     CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01028     CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01029
01030 #elif OC4M == OC4REF_LOW
01031
01032     CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01033     SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01034     CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01035
01036 #elif OC4M == OC4REF_TOGGLE
01037
01038     SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01039     SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01040     CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01041
01042 #elif OC4M == OC4REF_FORCED_LOW
01043
01044     CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01045     CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01046     SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01047
01048 #elif OC4M == OC4REF_FORCED_HIGH
01049
01050     SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01051     CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01052     SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01053
01054 #elif OC4M == PWM1
01055
01056     CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01057     SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01058     SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01059
01060 #elif OC4M == PWM2
01061
01062     SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01063     SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01064     SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01065
01066 #endif
01067
01068 // Output Compare 4 clear enable:
01069 #if OC4C_EN == OC4Ref_NOTEFFECTED
01070
01071     CLR_BIT( TIM1->CCMR2, OC4CE ) ;
01072
01073 #elif OC4C_EN == OC4Ref_CLEARED
01074
01075     SET_BIT( TIM1->CCMR2, OC4CE ) ;
01076
01077 #endif
01078
01079 /*_____CCER_____*/
01080
01081 // Capture/Compare 1 output enable:
01082 #if CC1_EN == ENABLE
01083
01084     SET_BIT( TIM1->CCER, CC1E ) ;
01085
01086 #elif CC1_EN == DISABLE
01087
01088     CLR_BIT( TIM1->CCER, CC1E ) ;
01089
01090 #endif
01091
01092 // Capture/Compare 1 output polarity:
01093 #if CC1_POLARITY == ACTIVE_HIGH
01094
01095     CLR_BIT( TIM1->CCER, CC1P ) ;
01096
01097 #elif CC1_POLARITY == ACTIVE_LOW
01098
01099     SET_BIT( TIM1->CCER, CC1P ) ;
```

```
01100
01101 #endif
01102
01103 // Capture/Compare 1 complementary output enable:
01104 #if CC1N_EN == ENABLE
01105
01106     SET_BIT( TIM1->CCER, CC1NE ) ;
01107
01108 #elif CC1N_EN == DISABLE
01109
01110     CLR_BIT( TIM1->CCER, CC1NE ) ;
01111
01112 #endif
01113
01114 // Capture/Compare 1 complementary output polarity:
01115 #if CC1N_POLARITY == ACTIVE_HIGH
01116
01117     CLR_BIT( TIM1->CCER, CC1NP ) ;
01118
01119 #elif CC1N_POLARITY == ACTIVE_LOW
01120
01121     SET_BIT( TIM1->CCER, CC1NP ) ;
01122
01123 #endif
01124
01125 // Capture/Compare 2 output enable:
01126 #if CC2_EN == ENABLE
01127
01128     SET_BIT( TIM1->CCER, CC2E ) ;
01129
01130 #elif CC2_EN == DISABLE
01131
01132     CLR_BIT( TIM1->CCER, CC2E ) ;
01133
01134 #endif
01135
01136 // Capture/Compare 2 output polarity:
01137 #if CC2_POLARITY == ACTIVE_HIGH
01138
01139     CLR_BIT( TIM1->CCER, CC2P ) ;
01140
01141 #elif CC2_POLARITY == ACTIVE_LOW
01142
01143     SET_BIT( TIM1->CCER, CC2P ) ;
01144
01145 #endif
01146
01147 // Capture/Compare 2 complementary output enable:
01148 #if CC2N_EN == ENABLE
01149
01150     SET_BIT( TIM1->CCER, CC2NE ) ;
01151
01152 #elif CC2N_EN == DISABLE
01153
01154     CLR_BIT( TIM1->CCER, CC2NE ) ;
01155
01156 #endif
01157
01158 // Capture/Compare 2 complementary output polarity:
01159 #if CC2N_POLARITY == ACTIVE_HIGH
01160
01161     CLR_BIT( TIM1->CCER, CC2NP ) ;
01162
01163 #elif CC2N_POLARITY == ACTIVE_LOW
01164
01165     SET_BIT( TIM1->CCER, CC2NP ) ;
01166
01167 #endif
01168
01169 // Capture/Compare 3 output enable:
01170 #if CC3_EN == ENABLE
01171
01172     SET_BIT( TIM1->CCER, CC3E ) ;
01173
01174 #elif CC3_EN == DISABLE
01175
01176     CLR_BIT( TIM1->CCER, CC3E ) ;
01177
01178 #endif
01179
01180 // Capture/Compare 3 output polarity:
01181 #if CC1_POLARITY == ACTIVE_HIGH
01182
01183     CLR_BIT( TIM1->CCER, CC3P ) ;
01184
01185 #elif CC1_POLARITY == ACTIVE_LOW
01186
```

```
01187     SET_BIT( TIM1->CCER, CC3P ) ;
01188
01189 #endif
01190
01191 // Capture/Compare 3 complementary output enable:
01192 #if CC3N_EN == ENABLE
01193
01194     SET_BIT( TIM1->CCER, CC3NE ) ;
01195
01196 #elif CC3N_EN == DISABLE
01197
01198     CLR_BIT( TIM1->CCER, CC3NE ) ;
01199
01200 #endif
01201
01202 // Capture/Compare 3 complementary output polarity:
01203 #if CC3N_POLARITY == ACTIVE_HIGH
01204
01205     CLR_BIT( TIM1->CCER, CC3NP ) ;
01206
01207 #elif CC3N_POLARITY == ACTIVE_LOW
01208
01209     SET_BIT( TIM1->CCER, CC3NP ) ;
01210
01211 #endif
01212
01213 // Capture/Compare 4 output enable:
01214 #if CC4_EN == ENABLE
01215
01216     SET_BIT( TIM1->CCER, CC4E ) ;
01217
01218 #elif CC4_EN == DISABLE
01219
01220     CLR_BIT( TIM1->CCER, CC4E ) ;
01221
01222 #endif
01223
01224 // Capture/Compare 4 output polarity:
01225 #if CC4_POLARITY == ACTIVE_HIGH
01226
01227     CLR_BIT( TIM1->CCER, CC4P ) ;
01228
01229 #elif CC4_POLARITY == ACTIVE_LOW
01230
01231     SET_BIT( TIM1->CCER, CC4P ) ;
01232
01233 #endif
01234
01235 /*_____BDTR_____*/
01236
01237 // Off-state selection for Idle mode:
01238 #if OSSI_MODE == OC_ENABLE
01239
01240     SET_BIT( TIM1->BDTR, OSSI ) ;
01241
01242 #elif OSSI_MODE == OC_DISABLE
01243
01244     CLR_BIT( TIM1->BDTR, OSSI ) ;
01245
01246 #endif
01247
01248 // Off-state selection for Run mode:
01249 #if OSSR_MODE == OC_ENABLE
01250
01251     SET_BIT( TIM1->BDTR, OSSR ) ;
01252
01253 #elif OSSR_MODE == OC_DISABLE
01254
01255     CLR_BIT( TIM1->BDTR, OSSR ) ;
01256
01257 #endif
01258
01259 // Automatic output enable:
01260 #if AO_EN == AUTO
01261
01262     SET_BIT( TIM1->BDTR, AOE ) ;
01263
01264 #elif AO_EN == SW
01265
01266     CLR_BIT( TIM1->BDTR, AOE ) ;
01267
01268 #endif
01269
01270 // Main output enable:
01271 #if MO_EN == OC_ENABLE
01272
01273     SET_BIT( TIM1->BDTR, MOE ) ;
```

```
01274
01275 #elif MO_EN == OC_DISABLE
01276
01277     CLR_BIT( TIM1->BDTR, MOE ) ;
01278
01279 #endif
01280
01281 }
01282
01283
01284 //*****
01285
01286 void MTIM1_vSetCounterValue( u16_t A_u16CNT_Value )
01287 {
01288
01289     TIM1->CNT = A_u16CNT_Value ;
01290
01291 }
01292
01293
01294 //*****
01295
01296 void MTIM1_vSetPrescalerValue( u16_t A_u16PSC_Value )
01297 {
01298
01299     TIM1->PSC = A_u16PSC_Value ;
01300
01301 }
01302
01303
01304
01305
01306 void MTIM1_vSetAutoReloadValue( u16_t A_u16ARR_Value )
01307 {
01308
01309     TIM1->ARR = A_u16ARR_Value ;
01310
01311 }
01312
01313
01314
01315
01316 void MTIM1_vSetRepetitionValue( u8_t A_u8RCR_Value )
01317 {
01318
01319     TIM1->RCR = A_u8RCR_Value ;
01320
01321 }
01322
01323
01324
01325
01326 void MTIM1_vSetCompareReg1Value( u16_t A_u16CCR1_Value )
01327 {
01328
01329     TIM1->CCR1 = A_u16CCR1_Value ;
01330
01331 }
01332
01333
01334
01335
01336 u16_t MTIM1_u16GetCaptureReg1Value( void )
01337 {
01338
01339     return TIM1->CCR1 ;
01340
01341 }
01342
01343
01344
01345
01346 void MTIM1_vSetCompareReg2Value( u16_t A_u16CCR2_Value )
```

```

01347 {
01348     TIM1->CCR2 = A_u16CCR2_Value ;
01350
01351 }
01352
01353
01354 //*****
01355 //*****
01356 u16_t MTIM1_u16GetCaptureReg2Value( void )
01357 {
01358     return TIM1->CCR2 ;
01359 }
01360
01361 }
01362
01363
01364 //*****
01365 //*****
01366 void MTIM1_vSetCompareReg3Value( u16_t A_u16CCR3_Value )
01367 {
01368
01369     TIM1->CCR3 = A_u16CCR3_Value ;
01370
01371 }
01372
01373
01374 //*****
01375 //*****
01376 u16_t MTIM1_u16GetCaptureReg3Value( void )
01377 {
01378
01379     return TIM1->CCR3 ;
01380
01381 }
01382
01383
01384 //*****
01385 //*****
01386 void MTIM1_vSetCompareReg4Value( u16_t A_u16CCR4_Value )
01387 {
01388
01389     TIM1->CCR4 = A_u16CCR4_Value ;
01390
01391 }
01392
01393
01394 //*****
01395 //*****
01396 u16_t MTIM1_u16GetCaptureReg4Value( void )
01397 {
01398
01399     return TIM1->CCR4 ;
01400
01401 }
01402
01403
01404 //*****
01405 //*****
01406 void MTIM1_vGeneratePWM( u8_t A_u8T1CHx, u8_t A_u8PWM_Mode, u8_t A_u8CenterMode,
01407     u16_t A_u16PSC_Value, u16_t A_u16ARR_Value, u16_t A_u16CCRx_Value )
01408 {
01409
01410     // Auto-reload preload enable:
01411     SET_BIT( TIM1->CR1, ARPE ) ;
01412
01413     // Set counter direction as up-counter:
01414     CLR_BIT( TIM1->CR1, DIR ) ;
01415
01416     // Compare - OC1REF signal is used as trigger output (TRGO):
01417     CLR_BIT( TIM1->CR2, MMS0 ) ;
01418     CLR_BIT( TIM1->CR2, MMS1 ) ;
01419     SET_BIT( TIM1->CR2, MMS2 ) ;
01420
01421     // Select master mode Compare:

```

```

01422 SET_BIT( TIM1->SMCR, MSM ) ;
01423
01424 // Set timer Prescaler, bus clock = 8 MHz, fCK_PSC / (PSC[15:0] + 1)
01425 // CK_CNT = F / (PSC+1) -> FREQ Hz -> time base = 1/FREQ
01426 TIM1->PSC = A_u16PSC_Value ;
01427
01428 // Set timer auto reload value:
01429 TIM1->ARR = A_u16ARR_Value ;
01430
01431 // Set timer compare value:
01432 switch( A_u8T1CHx )
01433 {
01434     case TIM1_CH1: TIM1->CCR1 = A_u16CCRx_Value ; break ;
01435
01436     case TIM1_CH2: TIM1->CCR2 = A_u16CCRx_Value ; break ;
01437
01438     case TIM1_CH3: TIM1->CCR3 = A_u16CCRx_Value ; break ;
01439
01440     case TIM1_CH4: TIM1->CCR4 = A_u16CCRx_Value ; break ;
01441 }
01442
01443 // Set Capture/Compare CHx as output & preload enable:
01444 switch( A_u8T1CHx )
01445 {
01446     case TIM1_CH1:
01447     {
01448         CLR_BIT( TIM1->CCMR1, CC1S0 ) ;
01449         CLR_BIT( TIM1->CCMR1, CC1S1 ) ;
01450
01451         SET_BIT( TIM1->CCMR1, OC1PE ) ;
01452     }
01453     break ;
01454
01455     case TIM1_CH2:
01456     {
01457         CLR_BIT( TIM1->CCMR1, CC2S0 ) ;
01458         CLR_BIT( TIM1->CCMR1, CC2S1 ) ;
01459
01460         SET_BIT( TIM1->CCMR1, OC2PE ) ;
01461     }
01462     break ;
01463
01464     case TIM1_CH3:
01465     {
01466         CLR_BIT( TIM1->CCMR2, CC3S0 ) ;
01467         CLR_BIT( TIM1->CCMR2, CC3S1 ) ;
01468
01469         SET_BIT( TIM1->CCMR2, OC3PE ) ;
01470     }
01471     break ;
01472
01473     case TIM1_CH4:
01474     {
01475         CLR_BIT( TIM1->CCMR2, CC4S0 ) ;
01476         CLR_BIT( TIM1->CCMR2, CC4S1 ) ;
01477
01478         SET_BIT( TIM1->CCMR2, OC4PE ) ;
01479     }
01480     break ;
01481
01482 }
01483
01484 // Select Output Compare PWM mode:
01485 switch( A_u8T1CHx )
01486 {
01487     case TIM1_CH1:
01488     {
01489         switch( A_u8PWM_Mode )
01490         {
01491
01492             case FROZEN:
01493             {
01494                 CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
01495                 CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
01496                 CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
01497             }
01498             break ;
01499
01500             case OC1REF_HIGH:
01501             {
01502                 SET_BIT( TIM1->CCMR1, OC1M0 ) ;
01503                 CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
01504                 CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
01505             }
01506             break ;
01507
01508             case OC1REF_LOW:
01509
01510         }
01511     }
01512 }

```

```

01509         {
01510             CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
01511             SET_BIT( TIM1->CCMR1, OC1M1 ) ;
01512             CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
01513         }
01514         break ;
01515
01516     case OC1REF_TOGGLE:
01517     {
01518         SET_BIT( TIM1->CCMR1, OC1M0 ) ;
01519         SET_BIT( TIM1->CCMR1, OC1M1 ) ;
01520         CLR_BIT( TIM1->CCMR1, OC1M2 ) ;
01521     }
01522     break ;
01523
01524     case OC1REF_FORCED_LOW:
01525     {
01526         CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
01527         CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
01528         SET_BIT( TIM1->CCMR1, OC1M2 ) ;
01529     }
01530     break ;
01531
01532     case OC1REF_FORCED_HIGH:
01533     {
01534         SET_BIT( TIM1->CCMR1, OC1M0 ) ;
01535         CLR_BIT( TIM1->CCMR1, OC1M1 ) ;
01536         SET_BIT( TIM1->CCMR1, OC1M2 ) ;
01537     }
01538     break ;
01539
01540     case PWM1:
01541     {
01542         CLR_BIT( TIM1->CCMR1, OC1M0 ) ;
01543         SET_BIT( TIM1->CCMR1, OC1M1 ) ;
01544         SET_BIT( TIM1->CCMR1, OC1M2 ) ;
01545     }
01546     break ;
01547
01548     case PWM2:
01549     {
01550         SET_BIT( TIM1->CCMR1, OC1M0 ) ;
01551         SET_BIT( TIM1->CCMR1, OC1M1 ) ;
01552         SET_BIT( TIM1->CCMR1, OC1M2 ) ;
01553     }
01554     break ;
01555 }
01556 }
01557 break ;
01558
01559 case TIM1_CH2:
01560 {
01561     switch( A_u8PWM_Mode )
01562     {
01563         case FROZEN:
01564     {
01565         CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
01566         CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
01567         CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
01568     }
01569     break ;
01570
01571         case OC1REF_HIGH:
01572     {
01573         SET_BIT( TIM1->CCMR1, OC2M0 ) ;
01574         CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
01575         CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
01576     }
01577     break ;
01578
01579         case OC1REF_LOW:
01580     {
01581         CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
01582         SET_BIT( TIM1->CCMR1, OC2M1 ) ;
01583         CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
01584     }
01585     break ;
01586
01587         case OC1REF_TOGGLE:
01588     {
01589         SET_BIT( TIM1->CCMR1, OC2M0 ) ;
01590         SET_BIT( TIM1->CCMR1, OC2M1 ) ;
01591         CLR_BIT( TIM1->CCMR1, OC2M2 ) ;
01592     }
01593     break ;
01594
01595 }
```

```

01596     case OC1REF_FORCED_LOW:
01597     {
01598         CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
01599         CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
01600         SET_BIT( TIM1->CCMR1, OC2M2 ) ;
01601     }
01602     break ;
01603
01604     case OC1REF_FORCED_HIGH:
01605     {
01606         SET_BIT( TIM1->CCMR1, OC2M0 ) ;
01607         CLR_BIT( TIM1->CCMR1, OC2M1 ) ;
01608         SET_BIT( TIM1->CCMR1, OC2M2 ) ;
01609     }
01610     break ;
01611
01612     case PWM1:
01613     {
01614         CLR_BIT( TIM1->CCMR1, OC2M0 ) ;
01615         SET_BIT( TIM1->CCMR1, OC2M1 ) ;
01616         SET_BIT( TIM1->CCMR1, OC2M2 ) ;
01617     }
01618     break ;
01619
01620     case PWM2:
01621     {
01622         SET_BIT( TIM1->CCMR1, OC2M0 ) ;
01623         SET_BIT( TIM1->CCMR1, OC2M1 ) ;
01624         SET_BIT( TIM1->CCMR1, OC2M2 ) ;
01625     }
01626     break ;
01627 }
01628 }
01629 break ;
01630
01631 case TIM1_CH3:
01632 {
01633     switch( A_u8PWM_Mode )
01634     {
01635
01636         case FROZEN:
01637         {
01638             CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
01639             CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
01640             CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
01641         }
01642     break ;
01643
01644         case OC1REF_HIGH:
01645         {
01646             SET_BIT( TIM1->CCMR2, OC3M0 ) ;
01647             CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
01648             CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
01649         }
01650     break ;
01651
01652         case OC1REF_LOW:
01653         {
01654             CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
01655             SET_BIT( TIM1->CCMR2, OC3M1 ) ;
01656             CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
01657         }
01658     break ;
01659
01660         case OC1REF_TOGGLE:
01661         {
01662             SET_BIT( TIM1->CCMR2, OC3M0 ) ;
01663             SET_BIT( TIM1->CCMR2, OC3M1 ) ;
01664             CLR_BIT( TIM1->CCMR2, OC3M2 ) ;
01665         }
01666     break ;
01667
01668         case OC1REF_FORCED_LOW:
01669         {
01670             CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
01671             CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
01672             SET_BIT( TIM1->CCMR2, OC3M2 ) ;
01673         }
01674     break ;
01675
01676         case OC1REF_FORCED_HIGH:
01677         {
01678             SET_BIT( TIM1->CCMR2, OC3M0 ) ;
01679             CLR_BIT( TIM1->CCMR2, OC3M1 ) ;
01680             SET_BIT( TIM1->CCMR2, OC3M2 ) ;
01681         }
01682     break ;

```

```

01683
01684     case PWM1:
01685     {
01686         CLR_BIT( TIM1->CCMR2, OC3M0 ) ;
01687         SET_BIT( TIM1->CCMR2, OC3M1 ) ;
01688         SET_BIT( TIM1->CCMR2, OC3M2 ) ;
01689     }
01690     break ;
01691
01692     case PWM2:
01693     {
01694         SET_BIT( TIM1->CCMR2, OC3M0 ) ;
01695         SET_BIT( TIM1->CCMR2, OC3M1 ) ;
01696         SET_BIT( TIM1->CCMR2, OC3M2 ) ;
01697     }
01698     break ;
01699 }
01700 }
01701 break ;
01702
01703 case TIM1_CH4:
01704 {
01705     switch( A_u8PWM_Mode )
01706     {
01707
01708         case FROZEN:
01709     {
01710         CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01711         CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01712         CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01713     }
01714     break ;
01715
01716         case OC1REF_HIGH:
01717     {
01718         SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01719         CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01720         CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01721     }
01722     break ;
01723
01724         case OC1REF_LOW:
01725     {
01726         CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01727         SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01728         CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01729     }
01730     break ;
01731
01732         case OC1REF_TOGGLE:
01733     {
01734         SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01735         SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01736         CLR_BIT( TIM1->CCMR2, OC4M2 ) ;
01737     }
01738     break ;
01739
01740         case OC1REF_FORCED_LOW:
01741     {
01742         CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01743         CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01744         SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01745     }
01746     break ;
01747
01748         case OC1REF_FORCED_HIGH:
01749     {
01750         SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01751         CLR_BIT( TIM1->CCMR2, OC4M1 ) ;
01752         SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01753     }
01754     break ;
01755
01756         case PWM1:
01757     {
01758         CLR_BIT( TIM1->CCMR2, OC4M0 ) ;
01759         SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01760         SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01761     }
01762     break ;
01763
01764         case PWM2:
01765     {
01766         SET_BIT( TIM1->CCMR2, OC4M0 ) ;
01767         SET_BIT( TIM1->CCMR2, OC4M1 ) ;
01768         SET_BIT( TIM1->CCMR2, OC4M2 ) ;
01769     }

```

```

01770             break ;
01771         }
01772     }
01773     break ;
01774 }
01775
01776 // Center-aligned mode selection:
01777 switch( A_u8CenterMode )
01778 {
01779     case EDGE:
01780     {
01781         CLR_BIT( TIM1->CR1, CMS0 ) ;
01782         CLR_BIT( TIM1->CR1, CMS1 ) ;
01783     }
01784     break ;
01785
01786     case CENTER1:
01787     {
01788         SET_BIT( TIM1->CR1, CMS0 ) ;
01789         CLR_BIT( TIM1->CR1, CMS1 ) ;
01790     }
01791     break ;
01792
01793     case CENTER2:
01794     {
01795         CLR_BIT( TIM1->CR1, CMS0 ) ;
01796         SET_BIT( TIM1->CR1, CMS1 ) ;
01797     }
01798     break ;
01799
01800     case CENTER3:
01801     {
01802         SET_BIT( TIM1->CR1, CMS0 ) ;
01803         SET_BIT( TIM1->CR1, CMS1 ) ;
01804     }
01805     break ;
01806 }
01807
01808 // Initialize all the registers:
01809 SET_BIT( TIM1->EGR, UG ) ;
01810
01811 // Enable Capture/Compare output:
01812 if( A_u8T1CHx == TIM1_CH1 )
01813 {
01814     CLR_BIT( TIM1->CCER, CC1P ) ;
01815
01816     SET_BIT( TIM1->CCER, CC1E ) ;
01817
01818     MGPIox_ConfigType T1_CH1 =
01819     {
01820         .Port          = T1CHx_PORT           , .Pin          = T1CH1_PIN           ,
01821         .Mode          = GPIOx_MODE_AF        , .OutputType   = GPIOx_PUSH_PULL   ,
01822         .OutputSpeed   = GPIOx_LowSpeed      , .InputType    = GPIOx_NoPull      ,
01823         .AF_Type       = GPIOx_AF1          ,           .
01824     } ;
01825
01826     MGPIox_vInit( &T1_CH1 ) ;
01827 }
01828 else
01829 {
01830     CLR_BIT( TIM1->CCER, CC1E ) ;
01831 }
01832
01833 if( A_u8T1CHx == TIM1_CH2 )
01834 {
01835     CLR_BIT( TIM1->CCER, CC2P ) ;
01836
01837     SET_BIT( TIM1->CCER, CC2E ) ;
01838
01839     MGPIox_ConfigType T1_CH2 =
01840     {
01841         .Port          = T1CHx_PORT           , .Pin          = T1CH2_PIN           ,
01842         .Mode          = GPIOx_MODE_AF        , .OutputType   = GPIOx_PUSH_PULL   ,
01843         .OutputSpeed   = GPIOx_LowSpeed      , .InputType    = GPIOx_NoPull      ,
01844         .AF_Type       = GPIOx_AF1          ,           .
01845     } ;
01846
01847     MGPIox_vInit( &T1_CH2 ) ;
01848 }
01849 else
01850 {
01851     CLR_BIT( TIM1->CCER, CC2E ) ;
01852 }
01853
01854 if( A_u8T1CHx == TIM1_CH3 )
01855 {
01856     CLR_BIT( TIM1->CCER, CC3P ) ;

```

```

01857     SET_BIT( TIM1->CCER, CC3E ) ;
01858
01859     MGPIox_ConfigType T1_CH3 =
01860     {
01861         .Port          = T1CHx_PORT           ,   .Pin          = T1CH3_PIN
01862         .Mode          = GPIOx_MODE_AF        ,   .OutputType = GPIOx_PUSHPULL      ,
01863         .OutputSpeed   = GPIOx_LowSpeed       ,   .InputType  = GPIOx_NoPull
01864         .AF_Type       = GPIOx_AF1
01865     } ;
01866
01867     MGPIox_vInit( &T1_CH3 ) ;
01868 }
01869 else
01870 {
01871     CLR_BIT( TIM1->CCER, CC3E ) ;
01872 }
01873
01874 if( A_u8T1CHx == TIM1_CH4 )
01875 {
01876     CLR_BIT( TIM1->CCER, CC4P ) ;
01877
01878     SET_BIT( TIM1->CCER, CC4E ) ;
01879
01880     MGPIox_ConfigType T1_CH4 =
01881     {
01882         .Port          = T1CHx_PORT           ,   .Pin          = T1CH4_PIN
01883         .Mode          = GPIOx_MODE_AF        ,   .OutputType = GPIOx_PUSHPULL      ,
01884         .OutputSpeed   = GPIOx_LowSpeed       ,   .InputType  = GPIOx_NoPull
01885         .AF_Type       = GPIOx_AF1
01886     } ;
01887
01888     MGPIox_vInit( &T1_CH4 ) ;
01889 }
01890 else
01891 {
01892     CLR_BIT( TIM1->CCER, CC4E ) ;
01893 }
01894
01895 // Enable timer main output:
01896 SET_BIT( TIM1->BDTR, MOE ) ;
01897
01898 }
01899 }
01900
01901 //*****
01902 //*****
01903
01904 void MTIM1_vReadPWM( void )
01905 {
01906
01907     // Enable ICU PINS:
01908     MGPIox_ConfigType T1_CH1 =
01909     {
01910         .Port          = T1CHx_PORT           ,   .Pin          = T1CH1_PIN
01911         .Mode          = GPIOx_MODE_AF        ,   .OutputType = GPIOx_PUSHPULL      ,
01912         .OutputSpeed   = GPIOx_LowSpeed       ,   .InputType  = GPIOx_NoPull
01913         .AF_Type       = GPIOx_AF1
01914     } ;
01915
01916     MGPIox_ConfigType T1_CH2 =
01917     {
01918         .Port          = T1CHx_PORT           ,   .Pin          = T1CH2_PIN
01919         .Mode          = GPIOx_MODE_AF        ,   .OutputType = GPIOx_PUSHPULL      ,
01920         .OutputSpeed   = GPIOx_LowSpeed       ,   .InputType  = GPIOx_NoPull
01921         .AF_Type       = GPIOx_AF1
01922     } ;
01923
01924     MGPIox_vInit( &T1_CH1 ) ;
01925     MGPIox_vInit( &T1_CH2 ) ;
01926
01927     // TIM1 IC1 Configurations:
01928     // Select TI1 for CCR1:
01929     SET_BIT( TIM1->CCMR1, CC1S0 ) ;
01930     CLR_BIT( TIM1->CCMR1, CC1S1 ) ;
01931
01932     // Select active Rising edge for TI1FP1:
01933     CLR_BIT( TIM1->CCER, CC1P ) ;
01934     CLR_BIT( TIM1->CCER, CC1NP ) ;
01935
01936     // TIM1 IC2 Configurations:
01937     // Select TI1 for CCR2:
01938     CLR_BIT( TIM1->CCMR1, CC2S0 ) ;
01939     SET_BIT( TIM1->CCMR1, CC2S1 ) ;
01940
01941     // Select active Falling edge for TI1FP2:

```

```

01942     SET_BIT( TIM1->CCER, CC2P      ) ;
01943     SET_BIT( TIM1->CCER, CC2NP      ) ;
01944
01945     // TIM1 Trigger Configurations:
01946     // Select TI1FP1 as trigger input:
01947     SET_BIT( TIM1->SMCR, TS0 ) ;
01948     CLR_BIT( TIM1->SMCR, TS1 ) ;
01949     SET_BIT( TIM1->SMCR, TS2 ) ;
01950
01951     // Configure Slave mode in reset mode:
01952     CLR_BIT( TIM1->SMCR, SMS0 ) ;
01953     CLR_BIT( TIM1->SMCR, SMS1 ) ;
01954     SET_BIT( TIM1->SMCR, SMS2 ) ;
01955
01956     // Enable the Captures:
01957     // Enable CC1:
01958     SET_BIT( TIM1->CCER, CC1E ) ;
01959
01960     // Enable CC2:
01961     SET_BIT( TIM1->CCER, CC2E ) ;
01962
01963 }
01964
01965
01966 //*****
01967 //*****
01968 void MTIM1_vEnableCounter( void )
01969 {
01970
01971     // Enable the counter:
01972     SET_BIT( TIM1->CR1, CEN ) ;
01973
01974 }
01975
01976
01977 //*****
01978
01979 void MTIM1_vDisableCounter( void )
01980 {
01981
01982     // Disable the counter:
01983     CLR_BIT( TIM1->CR1, CEN ) ;
01984
01985 }
01986
01987
01988 //*****
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013

```

7.134 UART_config.h

```

00001 /* FILENAME: UART_config
00002 * Author:    Ali El Bana

```

```
00003 * Version: V1.0
00004 * DATE: Fri 10/14/2022
00005 */
00006 #ifndef _UART_config_H
00007 #define _UART_config_H
00008
00009
00010
00011 #define __PCLK__ 8000000
00012
00013 #define THRESHOLD_VALUE 5000000
00014
00015 /****** USART1 configurations *****/
00016 /****** USART1 configurations *****/
00017 /****** USART1 configurations *****/
00018
00019 /*options:
00020 *GPIO_PORTA
00021 *GPIO_PORTB
00022 */
00023 #define USART1_PORT GPIO_PORTB
00024
00025 /****** USART1 configurations *****/
00026
00027 /*options:
00028 *GPIOx_PIN9
00029 *GPIOx_PIN6
00030 */
00031 #define TX1_PIN GPIOx_PIN6
00032
00033 /****** USART1 configurations *****/
00034
00035 /*options:
00036 *GPIOx_PIN10
00037 *GPIOx_PIN7
00038 */
00039 #define RX1_PIN GPIOx_PIN7
00040
00041 /****** USART2 configurations *****/
00042 /****** USART2 configurations *****/
00043 /****** USART2 configurations *****/
00044
00045 /*options:
00046 *GPIO_PORTA
00047 */
00048 #define USART2_PORT GPIO_PORTA
00049
00050 /****** USART2 configurations *****/
00051
00052 /*options:
00053 *GPIOx_PIN2
00054 */
00055 #define TX2_PIN GPIOx_PIN2
00056
00057 /****** USART2 configurations *****/
00058
00059 /*options:
00060 *GPIOx_PIN3
00061 */
00062 #define RX2_PIN GPIOx_PIN3
00063
00064 /****** USART3 configurations *****/
00065 /****** USART3 configurations *****/
00066 /****** USART3 configurations *****/
00067
00068 /*options:
00069 *GPIO_PORTA
00070 */
00071 #define USART6_PORT GPIO_PORTA
00072
00073 /****** USART3 configurations *****/
00074
00075 /*options:
00076 *GPIOx_PIN2
00077 */
00078 #define TX6_PIN GPIOx_PIN11
00079
00080 /****** USART3 configurations *****/
00081
00082 /*options:
00083 *GPIOx_PIN3
00084 */
00085 #define RX6_PIN GPIOx_PIN12
00086
00087 /****** USART3 configurations *****/
00088
00089
```

```

00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116 #endif // _UART_config_H

```

7.135 COTS/MCAL/UART/UART_interface.h File Reference

This file contains the interfacing information of UART driver.

Data Structures

- struct [USART_MemoryMapType](#)
USART declaration structure for its registers.
- struct [USART_InitType](#)
USART Clock declaration structure for.
- struct [USART_ClockInitTypeDef](#)
USART Clock declaration structure for its registers.

Macros

- [#define USART1_BASE_ADDRESS \(0x40011000\)](#)
- [#define USART2_BASE_ADDRESS \(0x40004400\)](#)
- [#define USART6_BASE_ADDRESS \(0x40011400\)](#)
- [#define USART1_REG \(\(USART_MemoryMapType *\)USART1_BASE_ADDRESS\)](#)
- [#define USART2_REG \(\(USART_MemoryMapType *\)USART2_BASE_ADDRESS\)](#)
- [#define USART6_REG \(\(USART_MemoryMapType *\)USART6_BASE_ADDRESS\)](#)
- [#define OVER_SAMPLING_16 \(0\)](#)
- [#define OVER_SAMPLING_8 \(1\)](#)
- [#define TX_ONLY \(0\)](#)
- [#define RX_ONLY \(1\)](#)
- [#define TX_RX \(2\)](#)
- [#define EVEN_PARITY \(0\)](#)
- [#define ODD_PARITY \(1\)](#)
- [#define MODE_8BIT \(0\)](#)
- [#define MODE_9BIT \(1\)](#)
- [#define STOP_BIT_1 \(0\)](#)
- [#define STOP_BIT_0_5 \(1\)](#)
- [#define STOP_BIT_2 \(2\)](#)
- [#define STOP_BIT_1_5 \(3\)](#)
- [#define ENABLE 1](#)
- [#define DISABLE 0](#)
- [#define __BAUDRATE__ 9600](#)

Functions

- void **MUSART_vInit** (**USART_InitType** *A_InitStruct, **USART_ClockInitTypeDef** *A_ClockInitStruct, **USART_MemoryMapType** *A_USARTx)

Sets a certain pin's output value on a specific port.
- void **MUSART_vEnable** (**USART_MemoryMapType** *A_USARTx)

Enables the USART peripheral.
- void **MUSART_vDisable** (**USART_MemoryMapType** *A_USARTx)

Disables the USART peripheral.
- void **MUSART_vTransmitByte** (**USART_MemoryMapType** *A_USARTx, **u8_t** A_u8Byte)

Transmits a byte using the USART peripheral.
- void **MUSART_vTransmitString** (**USART_MemoryMapType** *A_USARTx, **c8_t** *A_ptrc8String)

Transmits a string using the USART peripheral.
- **u8_t MUSART_u8ReceiveByteSynchNonBlocking** (**USART_MemoryMapType** *A_USARTx)

Receives a byte using the USART peripheral (non-blocking)
- **u8_t MUSART_u8ReceiveByteSynchBlocking** (**USART_MemoryMapType** *A_USARTx)

Receives a byte using the USART peripheral (blocking)
- **u8_t * MUSART_ptrReceiveStringSynchNonBlocking** (**USART_MemoryMapType** *A_USARTx)

Receives a string using the USART peripheral (non-blocking)
- void **MUSART_vRecieveString** (**USART_MemoryMapType** *A_USARTx, **c8_t** A_c8YourString[])

Receives a string using the USART peripheral (blocking)
- **u8_t MUSART_u8CompareString** (**c8_t** *String1, **c8_t** *String2)

Compares two strings using the USART peripheral.
- **u8_t MUSART_u8ReadDataRegister** (**USART_MemoryMapType** *A_USARTx)

Reads the status register of the USART peripheral.
- void **MUSART_vClearFlags** (**USART_MemoryMapType** *A_USARTx)

Clears the status register of the USART peripheral.
- void **MUSART_vRxIntSetStatus** (**USART_MemoryMapType** *A_USARTx, **u8_t** A_u8Status)

Set the status of the RX interrupt.
- void **MUSART1_vSetCallBack** (**void(*Fptr)(void)**)

Set UART1 RX callback function.
- void **MUSART2_vSetCallBack** (**void(*Fptr)(void)**)

Set UART2 RX callback function.
- void **MUSART6_vSetCallBack** (**void(*Fptr)(void)**)

Set UART6 RX callback function.

7.135.1 Detailed Description

This file contains the interfacing information of UART driver.

Author

Ali El Bana

Version

1.0

Date

10/14/2022

Definition in file [UART_interface.h](#).

7.135.2 Macro Definition Documentation

7.135.2.1 ENABLE

```
#define ENABLE 1
```

Enable Status

Definition at line 400 of file [UART_interface.h](#).

7.135.2.2 DISABLE

```
#define DISABLE 0
```

Disable Status

Definition at line 406 of file [UART_interface.h](#).

7.135.2.3 __BAUDRATE__

```
#define __BAUDRATE__ 9600
```

Default baud rate of the UART peripheral

Definition at line 412 of file [UART_interface.h](#).

7.135.3 Function Documentation

7.135.3.1 MUSART_vInit()

```
void MUSART_vInit (
    USART_InitType * A_InitStruct,
    USART_ClockInitTypeDef * A_ClockInitStruct,
    USART_MemoryMapType * A_USARTx )
```

Sets a certain pin's output value on a specific port.

Parameters

in	<i>A_InitStruct</i>	Pointer to a structure that contains the configuration information for the specified USART peripheral.
in	<i>A_ClockInitStruct</i>	Pointer to a structure that contains the configuration information for the specified USART peripheral.
in	<i>A_USARTx</i>	Pointer to USARTx registers map

Definition at line 37 of file [UART_program.c](#).

```

00038 {
00039
00040     if (A_USARTx == USART1_REG)
00041     {
00042
00043         MRCC_vEnablePeriphralCLK(RCC_APB2, APB2ENR_USART1EN);
00044
00045         /*configer Tx1 as alt fun*/
00046         MGPIox_ConfigType TX1 =
00047             {
00048             .Port = USART1_PORT, .Pin = TX1_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00049             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00050
00051         MGPIox_vInit(&TX1);
00052
00053         /*configer Rx1 as alt fun*/
00054         MGPIox_ConfigType RX1 =
00055             {
00056             .Port = USART1_PORT, .Pin = RX1_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00057             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00058
00059         MGPIox_vInit(&RX1);
00060     }
00061
00062     else if (A_USARTx == USART2_REG)
00063     {
00064
00065         MRCC_vEnablePeriphralCLK(RCC_APB2, APB1ENR_USART2EN);
00066
00067         /*configer Tx2 as alt fun*/
00068         MGPIox_ConfigType TX2 =
00069             {
00070             .Port = USART2_PORT, .Pin = TX2_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00071             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00072
00073         MGPIox_vInit(&TX2);
00074
00075         /*configer Rx2 as alt fun*/
00076         MGPIox_ConfigType RX2 =
00077             {
00078             .Port = USART2_PORT, .Pin = RX2_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00079             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00080
00081         MGPIox_vInit(&RX2);
00082     }
00083
00084     else if (A_USARTx == USART6_REG)
00085     {
00086
00087         MRCC_vEnablePeriphralCLK(RCC_APB2, APB2ENR_USART6EN);
00088
00089         /*configer Tx6 as alt fun*/
00090         MGPIox_ConfigType TX6 =
00091             {
00092             .Port = USART6_PORT, .Pin = TX6_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00093             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00094
00095         MGPIox_vInit(&TX6);
00096     }
00097
00098     // BuadRate/Over-sampling selections:
00099     switch (A_InitStruct->Oversampling)
00100     {
00101     case OVER_SAMPLING_16:
00102
00103

```

```

00104     A_USARTx->BRR_REG = UART_BRR_SAMPLING16(__PCLK__, A_InitStruct->BaudRate);
00105
00106     break;
00107
00108 case OVER_SAMPLING_8:
00109     A_USARTx->BRR_REG = UART_BRR_SAMPLING16(__PCLK__, A_InitStruct->BaudRate);
00110
00111     break;
00112 }
00113
00114 A_USARTx->CR1_REG = (A_InitStruct->Oversampling << MUSART_CR1_OVER8_BIT) |
00115     (A_InitStruct->DataWidth << MUSART_CR1_M_BIT) |
00116     (A_InitStruct->Parity_Enable << MUSART_CR1_PCE_BIT) |
00117     (A_InitStruct->Parity_Selection << MUSART_CR1_PS_BIT);
00118
00119 switch (A_InitStruct->TransferDirection)
00120 {
00121 case TX_ONLY:
00122
00123     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_TE_BIT);
00124
00125     break;
00126
00127 case RX_ONLY:
00128
00129     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_RE_BIT);
00130
00131     break;
00132
00133 case TX_RX:
00134
00135     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_TE_BIT);
00136     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_RE_BIT);
00137
00138     break;
00139 }
00140
00141 A_USARTx->CR2_REG = (A_InitStruct->StopBits << MUSART_CR2_STOP_BIT) |
00142     (A_ClockInitStruct->ClockOutput << MUSART_CR2_CLKEN_BIT) |
00143     (A_ClockInitStruct->ClockPhase << MUSART_CR2_CPHA_BIT) |
00144     (A_ClockInitStruct->ClockPolarity << MUSART_CR2_CPOL_BIT) |
00145     (A_ClockInitStruct->LastBitClockPulse << MUSART_CR2_LBCL_BIT);
00146
00147 A_USARTx->SR_REG = 0; // Clear all the flags.
00148
00149 SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_UE_BIT); // EN the peripheral.
00150
00151 }

```

References APB1ENR_USART2EN, APB2ENR_USART1EN, APB2ENR_USART6EN, USART_InitType::BaudRate, USART_MemoryMapType::BRR_REG, USART_ClockInitTypeDef::ClockOutput, USART_ClockInitTypeDef::ClockPhase, USART_ClockInitTypeDef::ClockPolarity, USART_MemoryMapType::CR1_REG, USART_MemoryMapType::CR2_REG, USART_InitType::DataWidth, GPIOx_AF7, GPIOx_MediumSpeed, GPIOx_MODE_AF, GPIOx_NoPull, GPIOx_PUSH_PULL, USART_ClockInitTypeDef::LastBitClockPulse, MGPIOP_vInit(), MRCC_vEnablePeripheralCLK(), MUSART_vInit(), OVER_SAMPLING_16, OVER_SAMPLING_8, USART_InitType::Oversampling, USART_InitType::Parity_Enable, USART_InitType::Parity_Selection, MGPIOP_ConfigType::Port, RCC_APB2, SET_BIT, USART_MemoryMapType::SR_REG, USART_InitType::StopBits, USART_InitType::TransferDirection, TX_ONLY, TX_RX, USART1_REG, USART2_REG, and USART6_REG.

Referenced by [MUSART_vInit\(\)](#).

7.135.3.2 MUSART_vEnable()

```

void MUSART_vEnable (
    USART_MemoryMapType * A_USARTx )

```

Enables the USART peripheral.

Parameters

in	A_USARTx	Pointer to USARTx registers map
----	----------	---------------------------------

Definition at line 156 of file [UART_program.c](#).

```
00157 {
00158     SET_BIT(A_USARTTx->CR1_REG, MUSART_CR1_UE_BIT);
00159 }
```

References [USART_MemoryMapType::CR1_REG](#), [MUSART_vEnable\(\)](#), and [SET_BIT](#).

Referenced by [MUSART_vEnable\(\)](#).

7.135.3.3 MUSART_vDisable()

```
void MUSART_vDisable (
    USART_MemoryMapType * A_USARTTx )
```

Disables the USART peripheral.

Parameters

in	A_USARTTx	Pointer to USARTTx registers map
----	-----------	----------------------------------

Definition at line 164 of file [UART_program.c](#).

```
00165 {
00166     CLR_BIT(A_USARTTx->CR1_REG, MUSART_CR1_UE_BIT);
00167 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), and [MUSART_vDisable\(\)](#).

Referenced by [MUSART_vDisable\(\)](#).

7.135.3.4 MUSART_vTransmitByte()

```
void MUSART_vTransmitByte (
    USART_MemoryMapType * A_USARTTx,
    u8_t A_u8Byte )
```

Transmits a byte using the USART peripheral.

Parameters

in	A_USARTTx	Pointer to USARTTx registers map
in	A_u8Byte	Byte to be transmitted

Definition at line 172 of file [UART_program.c](#).

```
00173 {
00174
00175     while (GET_BIT(A_USARTTx->SR_REG, MUSART_SR_TXE_BIT) == 0)
00176         ;
00177
00178     A_USARTTx->DR_REG = A_u8Byte;
00179
00180     while (GET_BIT(A_USARTTx->SR_REG, MUSART_SR_TC_BIT) == 0)
00181         ;
```

```

00182
00183     // Clear the TC flag:
00184     CLR_BIT(A_USARTTx->SR_REG, MUSART_SR_TC_BIT);
00185 }
```

References [CLR_BIT](#), [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_vTransmitByte\(\)](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [MUSART_vTransmitByte\(\)](#), and [MUSART_vTransmitString\(\)](#).

7.135.3.5 MUSART_vTransmitString()

```

void MUSART_vTransmitString (
    USART_MemoryMapType * A_USARTTx,
    c8_t * A_ptrc8String )
```

Transmits a string using the USART peripheral.

Parameters

in	<i>A_USARTTx</i>	Pointer to USARTTx registers map
in	<i>A_ptrc8String</i>	Pointer to the string to be transmitted

Definition at line 190 of file [UART_program.c](#).

```

00191 {
00192
00193     u8_t loc_u8Iterator = INITIAL_ZERO;
00194
00195     while (A_ptrc8String[loc_u8Iterator] != '\0')
00196     {
00197         MUSART_vTransmitByte(A_USARTTx, A_ptrc8String[loc_u8Iterator]);
00198         loc_u8Iterator++;
00199     }
00200 }
```

References [INITIAL_ZERO](#), [MUSART_vTransmitByte\(\)](#), and [MUSART_vTransmitString\(\)](#).

Referenced by [MUSART_vTransmitString\(\)](#).

7.135.3.6 MUSART_u8ReceiveByteSynchNonBlocking()

```

u8_t MUSART_u8ReceiveByteSynchNonBlocking (
    USART_MemoryMapType * A_USARTTx )
```

Receives a byte using the USART peripheral (non-blocking)

Parameters

in	<i>A_USARTTx</i>	Pointer to USARTTx registers map
----	------------------	----------------------------------

Returns

Received byte from the USART peripheral

Definition at line 207 of file [UART_program.c](#).

```
00208 {
00209     u8_t loc_u8Data = INITIAL_ZERO;
00210     u32_t loc_u8TimeOut = INITIAL_ZERO;
00211
00212     while ((GET_BIT(A_USARTTx->SR_REG, MUSART_SR_RXNE_BIT) == 0) && (loc_u8TimeOut < THRESHOLD_VALUE))
00213     {
00214         loc_u8TimeOut++;
00215     }
00216
00217     if (loc_u8TimeOut == THRESHOLD_VALUE)
00218     {
00219         loc_u8Data = 255; // To detect the error.
00220     }
00221     else
00222     {
00223         loc_u8Data = A_USARTTx->DR_REG;
00224     }
00225
00226     return loc_u8Data;
00227 }
00228 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [INITIAL_ZERO](#), [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#), and [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#).

7.135.3.7 MUSART_u8ReceiveByteSynchBlocking()

```
u8_t MUSART_u8ReceiveByteSynchBlocking (
    USART_MemoryMapType * A_USARTTx )
```

Receives a byte using the USART peripheral (blocking)

Parameters

in	A_USARTTx	Pointer to USARTTx registers map
----	-----------	----------------------------------

Returns

Received byte from the USART peripheral

Definition at line 234 of file [UART_program.c](#).

```
00235 {
00236
00237     while (GET_BIT(A_USARTTx->SR_REG, MUSART_SR_RXNE_BIT) == 0)
00238     ;
00239
00240     return A_USARTTx->DR_REG;
00241 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_u8ReceiveByteSynchBlocking\(\)](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [MUSART_u8ReceiveByteSynchBlocking\(\)](#), and [MUSART_vRecieveString\(\)](#).

7.135.3.8 MUSART_ptrReceiveStringSynchNonBlocking()

```
u8_t * MUSART_ptrReceiveStringSynchNonBlocking (
    USART_MemoryMapType * A_USARTx )
```

Receives a string using the USART peripheral (non-blocking)

Parameters

in	A_USARTx	Pointer to USARTx registers map
----	----------	---------------------------------

Returns

Pointer to the received string from the USART peripheral

Definition at line 246 of file [UART_program.c](#).

```
00247 {
00248
00249     u8_t loc_u8Iterator = INITIAL_ZERO;
00250
00251     u8_t loc_u8DataCome;
00252
00253     while ((loc_u8DataCome = MUSART_u8ReceiveByteSynchNonBlocking(A_USARTx)) != 13)
00254     {
00255         G_u8String[loc_u8Iterator] = loc_u8DataCome;
00256
00257         loc_u8Iterator++;
00258     }
00259
00260     G_u8String[loc_u8Iterator] = '\0';
00261
00262     return (G_u8String);
00263 }
```

References [INITIAL_ZERO](#), [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#), and [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#).

Referenced by [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#).

7.135.3.9 MUSART_vRecieveString()

```
void MUSART_vRecieveString (
    USART_MemoryMapType * A_USARTx,
    c8_t A_c8YourString[] )
```

Receives a string using the USART peripheral (blocking)

Parameters

in	A_USARTx	Pointer to USARTx registers map
in	A_c8YourString	Pointer to the string to be transmitted

Definition at line 268 of file [UART_program.c](#).

```
00269 {
00270
00271     u32_t L_u32Counter = INITIAL_ZERO;
00272
00273     A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking(A_USARTx);
```

```

00274
00275     while (A_c8YourString[L_u32Counter] != '\0')
00276     {
00277         L_u32Counter++;
00278
00279         A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking(A_USARTTx);
00280
00281         if ((A_c8YourString[L_u32Counter] == '\n') || (A_c8YourString[L_u32Counter] == '\r'))
00282         {
00283             A_c8YourString[L_u32Counter] = '\0';
00284         }
00285     }
00286 }
00287 }
```

References [INITIAL_ZERO](#), [MUSART_u8ReceiveByteSynchBlocking\(\)](#), and [MUSART_vRecieveString\(\)](#).

Referenced by [MUSART_vRecieveString\(\)](#).

7.135.3.10 MUSART_u8CompareString()

```

u8_t MUSART_u8CompareString (
    c8_t * String1,
    c8_t * String2 )
```

Compares two strings using the USART peripheral.

Parameters

in	<i>String1</i>	Pointer to the first string
in	<i>String2</i>	Pointer to the second string

Returns

SAME_STRING if the strings are equal, DIFFERENT_STRING if not

See also

[SAME_STRING](#) [DIFFERENT_STRING](#)

Definition at line 292 of file [UART_program.c](#).

```

00293 {
00294
00295     u8_t L_u8TheComparingResult = INITIAL_ZERO;
00296
00297     if (strcmp(String1, String2) == SAME_STRING)
00298     {
00299         L_u8TheComparingResult = SAME_STRING;
00300     }
00301     else
00302     {
00303         L_u8TheComparingResult = DIFFERENT_STRING;
00304     }
00305
00306     return L_u8TheComparingResult;
00307 }
```

References [DIFFERENT_STRING](#), [INITIAL_ZERO](#), [MUSART_u8CompareString\(\)](#), and [SAME_STRING](#).

Referenced by [MUSART_u8CompareString\(\)](#).

7.135.3.11 MUSART_u8ReadDataRegister()

```
u8_t MUSART_u8ReadDataRegister (
    USART_MemoryMapType * A_USARTx )
```

Reads the status register of the USART peripheral.

Parameters

in	A_USARTx	Pointer to USARTx registers map
----	----------	---------------------------------

Returns

Status register value

Definition at line 312 of file [UART_program.c](#).

```
00313 {
00314     return A_USARTx->DR_REG;
00315 }
```

References [USART_MemoryMapType::DR_REG](#), and [MUSART_u8ReadDataRegister\(\)](#).

Referenced by [MUSART_u8ReadDataRegister\(\)](#).

7.135.3.12 MUSART_vClearFlags()

```
void MUSART_vClearFlags (
    USART_MemoryMapType * A_USARTx )
```

Clears the status register of the USART peripheral.

Parameters

in	A_USARTx	Pointer to USARTx registers map
----	----------	---------------------------------

Definition at line 320 of file [UART_program.c](#).

```
00321 {
00322     A_USARTx->SR_REG = 0;
00323 }
```

References [MUSART_vClearFlags\(\)](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [MUSART_vClearFlags\(\)](#).

7.135.3.13 MUSART_vRxIntSetStatus()

```
void MUSART_vRxIntSetStatus (
    USART_MemoryMapType * A_USARTx,
    u8_t A_u8Status )
```

Set the status of the RX interrupt.

Parameters

in	<i>A_USARTx</i>	Pointer to USARTx registers map
in	<i>A_u8Status</i>	Status of the RX interrupt (ENABLE/DISABLE)

See also

[ENABLE DISABLE](#)

Definition at line 328 of file [UART_program.c](#).

```
00329 {
00330
00331     switch (A_u8Status)
00332     {
00333
00334         case ENABLE:
00335             SET_BIT(A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT));
00336             break;
00337
00338         case DISABLE:
00339             CLR_BIT(A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT));
00340             break;
00341     }
00342 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), [MUSART_vRxIntSetStatus\(\)](#), and [SET_BIT](#).

Referenced by [MUSART_vRxIntSetStatus\(\)](#).

7.135.3.14 MUSART1_vSetCallBack()

```
void MUSART1_vSetCallBack (
    void(*) (void) Fptr )
```

Set UART1 RX callback function.

Parameters

in	<i>Fptr</i>	Pointer to the callback function
----	-------------	----------------------------------

Definition at line 347 of file [UART_program.c](#).

```
00348 {
00349     MUSART1_CallBack = Fptr;
00350 }
```

References [MUSART1_vSetCallBack\(\)](#).

Referenced by [MUSART1_vSetCallBack\(\)](#).

7.135.3.15 MUSART2_vSetCallBack()

```
void MUSART2_vSetCallBack (
    void(*) (void) Fptr )
```

Set UART2 RX callback function.

Parameters

in	<i>Fptr</i>	Pointer to the callback function
----	-------------	----------------------------------

Definition at line 355 of file [UART_program.c](#).

```
00356 {
00357     MUSART2_CallBack = Fptr;
00358 }
```

References [MUSART2_vSetCallBack\(\)](#).

Referenced by [MUSART2_vSetCallBack\(\)](#).

7.135.3.16 MUSART6_vSetCallBack()

```
void MUSART6_vSetCallBack (
    void(*)(void) Fptr )
```

Set UART6 RX callback function.

Parameters

in	<i>Fptr</i>	Pointer to the callback function
----	-------------	----------------------------------

Definition at line 363 of file [UART_program.c](#).

```
00364 {
00365     MUSART6_CallBack = Fptr;
00366 }
```

References [MUSART6_vSetCallBack\(\)](#).

Referenced by [MUSART6_vSetCallBack\(\)](#).

7.136 UART_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _UART_interface_H
00011 #define _UART_interface_H
00012
00013 ****
00014 /* Peripherals declaration */
00015 ****
00016
00026 #define USART1_BASE_ADDRESS (0x40011000)
00027
00032 #define USART2_BASE_ADDRESS (0x40004400)
00033
00038 #define USART6_BASE_ADDRESS (0x40011400)
00039
00051 #define USART1_REG ((USART_MemoryMapType *)USART1_BASE_ADDRESS)
00052
00057 #define USART2_REG ((USART_MemoryMapType *)USART2_BASE_ADDRESS)
00058
00063 #define USART6_REG ((USART_MemoryMapType *)USART6_BASE_ADDRESS)
00064
00071 typedef struct
```

```

00072 {
00076     volatile u32_t SR_REG;
00080     volatile u32_t DR_REG;
00084     volatile u32_t BRR_REG;
00088     volatile u32_t CR1_REG;
00092     volatile u32_t CR2_REG;
00096     volatile u32_t CR3_REG;
00100     volatile u32_t GTPR_REG;
00101 } USART_MemoryMapType;
00102
00108 typedef struct
00109 {
00113     u32_t BaudRate;
00117     u8_t DataWidth;
00121     u8_t StopBits;
00125     u8_t Parity_Enable;
00129     u8_t Parity_Selection;
00134     u8_t TransferDirection;
00138     u8_t HardwareFlowControl;
00142     u8_t Oversampling;
00143 } USART_InitType;
00144
00149 typedef struct
00150 {
00154     u8_t ClockOutput;
00158     u8_t ClockPolarity;
00162     u8_t ClockPhase;
00166     u8_t LastBitClockPulse;
00167 } USART_ClockInitTypeDef;
00168
00169 /***** Functions prototypes *****/
00170 /*          */
00171 /***** Interfacing macros *****/
00172
00179 void MUSART_vInit(USART_InitType * A_InitStruct, USART_ClockInitTypeDef * A_ClockInitStruct,
                      USART_MemoryMapType * A_USARTTx);
00180
00185 void MUSART_vEnable(USART_MemoryMapType * A_USARTTx);
00186
00191 void MUSART_vDisable(USART_MemoryMapType * A_USARTTx);
00192
00198 void MUSART_vTransmitByte(USART_MemoryMapType * A_USARTTx, u8_t A_u8Byte);
00199
00205 void MUSART_vTransmitString(USART_MemoryMapType * A_USARTTx, c8_t * A_ptrc8String);
00206
00212 u8_t MUSART_u8ReceiveByteSyncNonBlocking(USART_MemoryMapType * A_USARTTx);
00213
00219 u8_t MUSART_u8ReceiveByteSyncBlocking(USART_MemoryMapType * A_USARTTx);
00220
00226 u8_t *MUSART_ptrReceiveStringSyncNonBlocking(USART_MemoryMapType * A_USARTTx);
00227
00233 void MUSART_vRecieveString(USART_MemoryMapType * A_USARTTx, c8_t A_c8YourString[]);
00234
00242 u8_t MUSART_u8CompareString(c8_t * String1, c8_t * String2);
00243
00249 u8_t MUSART_u8ReadDataRegister(USART_MemoryMapType * A_USARTTx);
00250
00255 void MUSART_vClearFlags(USART_MemoryMapType * A_USARTTx);
00256
00263 void MUSART_vRxIntSetStatus(USART_MemoryMapType * A_USARTTx, u8_t A_u8Status);
00264
00269 void MUSART1_vSetCallBack(void (*Fptr)(void));
00270
00275 void MUSART2_vSetCallBack(void (*Fptr)(void));
00276
00281 void MUSART6_vSetCallBack(void (*Fptr)(void));
00282
00283 /***** Interfacing macros *****/
00284 /*          */
00285 /***** Macros *****/
00286
00296 #define OVER_SAMPLING_16 (0)
00297
00302 #define OVER_SAMPLING_8 (1)
00314 #define TX_ONLY (0)
00315
00320 #define RX_ONLY (1)
00321
00326 #define TX_RX (2)
00338 #define EVEN_PARITY (0)
00339
00344 #define ODD_PARITY (1)
00356 #define MODE_8BIT (0)
00357
00362 #define MODE_9BIT (1)
00374 #define STOP_BIT_1 (0)
00375

```

```

00380 #define STOP_BIT_0_5 (1)
00381
00386 #define STOP_BIT_2 (2)
00387
00392 #define STOP_BIT_1_5 (3)
00393
00400 #define ENABLE 1
00401
00406 #define DISABLE 0
00407
00412 #define __BAUDRATE__ 9600
00413
00414 #endif // _UART_interface_H

```

7.137 COTS/MCAL/UART/UART_private.h File Reference

This file contains the private information of UART driver.

Macros

- `#define UART_DIV_SAMPLING16(_PCLK_, _BAUD_) ((u32_t)((((u64_t)(_PCLK_))*25U)/(4U*((u64_t)(← BAUD_))))`
- `#define UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_) (UART_DIV_SAMPLING16(_PCLK_), (_← BAUD_)/100U)`
- `#define UART_DIVFRAQ_SAMPLING16(_PCLK_, _BAUD_) (((UART_DIV_SAMPLING16(_PCLK_), (_← BAUD_)) - (UART_DIVMANT_SAMPLING16(_PCLK_), (_BAUD_)) * 100U) * 16U) / 100U)`
- `#define UART_BRR_SAMPLING16(_PCLK_, _BAUD_)`
- `#define UART_DIV_SAMPLING8(_PCLK_, _BAUD_) ((u32_t)((((u64_t)(_PCLK_))*25U)/(2U*((u64_t)(← BAUD_))))`
- `#define UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_) (UART_DIV_SAMPLING8(_PCLK_), (_BAUD←_)/100U)`
- `#define UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_) (((UART_DIV_SAMPLING8(_PCLK_), (_← BAUD_)) - (UART_DIVMANT_SAMPLING8(_PCLK_), (_BAUD_)) * 100U) * 8U) / 100U)`
- `#define UART_BRR_SAMPLING8(_PCLK_, _BAUD_)`
- `#define MUSART_SR_PE_BIT 0`
- `#define MUSART_SR_FE_BIT 1`
- `#define MUSART_SR_NE_BIT 2`
- `#define MUSART_SR_ORE_BIT 3`
- `#define MUSART_SR_IDLE_BIT 4`
- `#define MUSART_SR_RXNE_BIT 5`
- `#define MUSART_SR_TC_BIT 6`
- `#define MUSART_SR_TXE_BIT 7`
- `#define MUSART_SR_LBD_BIT 8`
- `#define MUSART_SR_CTS_BIT 9`
- `#define MUSART_CR1_SBK_BIT 0`
- `#define MUSART_CR1_RWU_BIT 1`
- `#define MUSART_CR1_RE_BIT 2`
- `#define MUSART_CR1_TE_BIT 3`
- `#define MUSART_CR1_IDLEIE_BIT 4`
- `#define MUSART_CR1_RXNEIE_BIT 5`
- `#define MUSART_CR1_TCIE_BIT 6`
- `#define MUSART_CR1_TXEIE_BIT 7`
- `#define MUSART_CR1_PEIE_BIT 8`
- `#define MUSART_CR1_PS_BIT 9`
- `#define MUSART_CR1_PCE_BIT 10`
- `#define MUSART_CR1_WAKE_BIT 11`

- #define MUSART_CR1_M_BIT 12
- #define MUSART_CR1_UE_BIT 13
- #define MUSART_CR1_OVER8_BIT 15
- #define MUSART_CR2_ADD0_BIT 0
- #define MUSART_CR2_ADD1_BIT 1
- #define MUSART_CR2_ADD2_BIT 2
- #define MUSART_CR2_ADD3_BIT 3
- #define MUSART_CR2_LBDL_BIT 5
- #define MUSART_CR2_LBDIE_BIT 6
- #define MUSART_CR2_LBCL_BIT 8
- #define MUSART_CR2_CPHA_BIT 9
- #define MUSART_CR2_CPOL_BIT 10
- #define MUSART_CR2_CLKEN_BIT 11
- #define MUSART_CR2_STOP_BIT 12
- #define MUSART_CR2_STOP0_BIT 12
- #define MUSART_CR2_STOP1_BIT 13
- #define MUSART_CR2_LINEN_BIT 14
- #define MUSART_CR3_CTSIE_BIT 10
- #define MUSART_CR3_CTSE_BIT 9
- #define MUSART_CR3_RTSE_BIT 8
- #define MUSART_CR3_DMAT_BIT 7
- #define MUSART_CR3_DMAR_BIT 6
- #define MUSART_CR3_SCEN_BIT 5
- #define MUSART_CR3_NACK_BIT 4
- #define MUSART_CR3_HDSEL_BIT 3
- #define MUSART_CR3_IRLP_BIT 2
- #define MUSART_CR3_IREN_BIT 1
- #define MUSART_CR3_EIE_BIT 0

7.137.1 Detailed Description

This file contains the private information of UART driver.

Author

Ali El Bana

Version

1.0

Date

10/14/2022

Definition in file [UART_private.h](#).

7.137.2 Macro Definition Documentation

7.137.2.1 UART_DIV_SAMPLING16

```
#define UART_DIV_SAMPLING16(
    _PCLK_,
    _BAUD_ ) (((u32_t) (((u64_t) (_PCLK_)) * 25U) / (4U * (u64_t) (_BAUD_))))
```

Definition at line 14 of file [UART_private.h](#).

7.137.2.2 UART_DIVMANT_SAMPLING16

```
#define UART_DIVMANT_SAMPLING16(
    _PCLK_,
    _BAUD_ ) (UART_DIV_SAMPLING16(_PCLK_, _BAUD_) / 100U)
```

Definition at line 15 of file [UART_private.h](#).

7.137.2.3 UART_DIVFRAQ_SAMPLING16

```
#define UART_DIVFRAQ_SAMPLING16(
    _PCLK_,
    _BAUD_ ) (((UART_DIV_SAMPLING16(_PCLK_, _BAUD_) - (UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_) * 100U)) * 16U) + 50U) / 100U)
```

Definition at line 16 of file [UART_private.h](#).

7.137.2.4 UART_BRR_SAMPLING16

```
#define UART_BRR_SAMPLING16(
    _PCLK_,
    _BAUD_ )
```

Value:

```
+ \
0xF0U) + \
0x0FU)
```

$$\begin{aligned} & ((\text{UART_DIVMANT_SAMPLING16}(_PCLK_, _BAUD_) \ll 4U) \\ & (\text{UART_DIVFRAQ_SAMPLING16}(_PCLK_, _BAUD_) \& \\ & (\text{UART_DIVFRAQ_SAMPLING16}(_PCLK_, _BAUD_) \& \end{aligned}$$

Definition at line 19 of file [UART_private.h](#).

7.137.2.5 UART_DIV_SAMPLING8

```
#define UART_DIV_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (((u32_t) (((u64_t) (_PCLK_)) * 25U) / (2U * ((u64_t) (_BAUD_)))))
```

Definition at line 23 of file [UART_private.h](#).

7.137.2.6 UART_DIVMANT_SAMPLING8

```
#define UART_DIVMANT_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (UART_DIV_SAMPLING8(_PCLK_, _BAUD_) / 100U)
```

Definition at line 24 of file [UART_private.h](#).

7.137.2.7 UART_DIVFRAQ_SAMPLING8

```
#define UART_DIVFRAQ_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (((UART_DIV_SAMPLING8(_PCLK_, _BAUD_) - (UART_DIVMANT_SAMPLING8((←
    _PCLK_), _BAUD_) * 100U)) * 8U) + 500) / 100U)
```

Definition at line 25 of file [UART_private.h](#).

7.137.2.8 UART_BRR_SAMPLING8

```
#define UART_BRR_SAMPLING8(
    _PCLK_,
    _BAUD_ )
```

Value:

```
\n
0xF8U) << 1U) + \
((UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_) << 4U) +
(UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_) &
(UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_) &
```

Definition at line 28 of file [UART_private.h](#).

7.137.2.9 MUSART_SR_PE_BIT

```
#define MUSART_SR_PE_BIT 0
```

Definition at line 37 of file [UART_private.h](#).

7.137.2.10 MUSART_SR_FE_BIT

```
#define MUSART_SR_FE_BIT 1
```

Definition at line 39 of file [UART_private.h](#).

7.137.2.11 MUSART_SR_NE_BIT

```
#define MUSART_SR_NE_BIT 2
```

Definition at line 41 of file [UART_private.h](#).

7.137.2.12 MUSART_SR_ORE_BIT

```
#define MUSART_SR_ORE_BIT 3
```

Definition at line 43 of file [UART_private.h](#).

7.137.2.13 MUSART_SR_IDLE_BIT

```
#define MUSART_SR_IDLE_BIT 4
```

Definition at line 45 of file [UART_private.h](#).

7.137.2.14 MUSART_SR_RXNE_BIT

```
#define MUSART_SR_RXNE_BIT 5
```

Definition at line 47 of file [UART_private.h](#).

7.137.2.15 MUSART_SR_TC_BIT

```
#define MUSART_SR_TC_BIT 6
```

Definition at line 49 of file [UART_private.h](#).

7.137.2.16 MUSART_SR_TXE_BIT

```
#define MUSART_SR_TXE_BIT 7
```

Definition at line 51 of file [UART_private.h](#).

7.137.2.17 MUSART_SR_LBD_BIT

```
#define MUSART_SR_LBD_BIT 8
```

Definition at line 53 of file [UART_private.h](#).

7.137.2.18 MUSART_SR_CTS_BIT

```
#define MUSART_SR_CTS_BIT 9
```

Definition at line 55 of file [UART_private.h](#).

7.137.2.19 MUSART_CR1_SBK_BIT

```
#define MUSART_CR1_SBK_BIT 0
```

Definition at line 61 of file [UART_private.h](#).

7.137.2.20 MUSART_CR1_RWU_BIT

```
#define MUSART_CR1_RWU_BIT 1
```

Definition at line 63 of file [UART_private.h](#).

7.137.2.21 MUSART_CR1_RE_BIT

```
#define MUSART_CR1_RE_BIT 2
```

Definition at line 65 of file [UART_private.h](#).

7.137.2.22 MUSART_CR1_TE_BIT

```
#define MUSART_CR1_TE_BIT 3
```

Definition at line 67 of file [UART_private.h](#).

7.137.2.23 MUSART_CR1_IDLEIE_BIT

```
#define MUSART_CR1_IDLEIE_BIT 4
```

Definition at line 69 of file [UART_private.h](#).

7.137.2.24 MUSART_CR1_RXNEIE_BIT

```
#define MUSART_CR1_RXNEIE_BIT 5
```

Definition at line 71 of file [UART_private.h](#).

7.137.2.25 MUSART_CR1_TCIE_BIT

```
#define MUSART_CR1_TCIE_BIT 6
```

Definition at line 73 of file [UART_private.h](#).

7.137.2.26 MUSART_CR1_TXEIE_BIT

```
#define MUSART_CR1_TXEIE_BIT 7
```

Definition at line 75 of file [UART_private.h](#).

7.137.2.27 MUSART_CR1_PEIE_BIT

```
#define MUSART_CR1_PEIE_BIT 8
```

Definition at line 77 of file [UART_private.h](#).

7.137.2.28 MUSART_CR1_PS_BIT

```
#define MUSART_CR1_PS_BIT 9
```

Definition at line 79 of file [UART_private.h](#).

7.137.2.29 MUSART_CR1_PCE_BIT

```
#define MUSART_CR1_PCE_BIT 10
```

Definition at line 81 of file [UART_private.h](#).

7.137.2.30 MUSART_CR1_WAKE_BIT

```
#define MUSART_CR1_WAKE_BIT 11
```

Definition at line 83 of file [UART_private.h](#).

7.137.2.31 MUSART_CR1_M_BIT

```
#define MUSART_CR1_M_BIT 12
```

Definition at line 85 of file [UART_private.h](#).

7.137.2.32 MUSART_CR1_UE_BIT

```
#define MUSART_CR1_UE_BIT 13
```

Definition at line 87 of file [UART_private.h](#).

7.137.2.33 MUSART_CR1_OVER8_BIT

```
#define MUSART_CR1_OVER8_BIT 15
```

Definition at line 89 of file [UART_private.h](#).

7.137.2.34 MUSART_CR2_ADD0_BIT

```
#define MUSART_CR2_ADD0_BIT 0
```

Definition at line 95 of file [UART_private.h](#).

7.137.2.35 MUSART_CR2_ADD1_BIT

```
#define MUSART_CR2_ADD1_BIT 1
```

Definition at line 96 of file [UART_private.h](#).

7.137.2.36 MUSART_CR2_ADD2_BIT

```
#define MUSART_CR2_ADD2_BIT 2
```

Definition at line 97 of file [UART_private.h](#).

7.137.2.37 MUSART_CR2_ADD3_BIT

```
#define MUSART_CR2_ADD3_BIT 3
```

Definition at line 98 of file [UART_private.h](#).

7.137.2.38 MUSART_CR2_LBDL_BIT

```
#define MUSART_CR2_LBDL_BIT 5
```

Definition at line 100 of file [UART_private.h](#).

7.137.2.39 MUSART_CR2_LBDIE_BIT

```
#define MUSART_CR2_LBDIE_BIT 6
```

Definition at line 102 of file [UART_private.h](#).

7.137.2.40 MUSART_CR2_LBCL_BIT

```
#define MUSART_CR2_LBCL_BIT 8
```

Definition at line 104 of file [UART_private.h](#).

7.137.2.41 MUSART_CR2_CPHA_BIT

```
#define MUSART_CR2_CPHA_BIT 9
```

Definition at line 106 of file [UART_private.h](#).

7.137.2.42 MUSART_CR2_CPOL_BIT

```
#define MUSART_CR2_CPOL_BIT 10
```

Definition at line 108 of file [UART_private.h](#).

7.137.2.43 MUSART_CR2_CLKEN_BIT

```
#define MUSART_CR2_CLKEN_BIT 11
```

Definition at line 110 of file [UART_private.h](#).

7.137.2.44 MUSART_CR2_STOP_BIT

```
#define MUSART_CR2_STOP_BIT 12
```

Definition at line 112 of file [UART_private.h](#).

7.137.2.45 MUSART_CR2_STOP0_BIT

```
#define MUSART_CR2_STOP0_BIT 12
```

Definition at line 114 of file [UART_private.h](#).

7.137.2.46 MUSART_CR2_STOP1_BIT

```
#define MUSART_CR2_STOP1_BIT 13
```

Definition at line 115 of file [UART_private.h](#).

7.137.2.47 MUSART_CR2_LINEN_BIT

```
#define MUSART_CR2_LINEN_BIT 14
```

Definition at line 117 of file [UART_private.h](#).

7.137.2.48 MUSART_CR3_CTSIE_BIT

```
#define MUSART_CR3_CTSIE_BIT 10
```

Definition at line 123 of file [UART_private.h](#).

7.137.2.49 MUSART_CR3_CTSE_BIT

```
#define MUSART_CR3_CTSE_BIT 9
```

Definition at line 125 of file [UART_private.h](#).

7.137.2.50 MUSART_CR3_RTSE_BIT

```
#define MUSART_CR3_RTSE_BIT 8
```

Definition at line 127 of file [UART_private.h](#).

7.137.2.51 MUSART_CR3_DMAT_BIT

```
#define MUSART_CR3_DMAT_BIT 7
```

Definition at line 129 of file [UART_private.h](#).

7.137.2.52 MUSART_CR3_DMAR_BIT

```
#define MUSART_CR3_DMAR_BIT 6
```

Definition at line 131 of file [UART_private.h](#).

7.137.2.53 MUSART_CR3_SCEN_BIT

```
#define MUSART_CR3_SCEN_BIT 5
```

Definition at line 133 of file [UART_private.h](#).

7.137.2.54 MUSART_CR3_NACK_BIT

```
#define MUSART_CR3_NACK_BIT 4
```

Definition at line 135 of file [UART_private.h](#).

7.137.2.55 MUSART_CR3_HDSEL_BIT

```
#define MUSART_CR3_HDSEL_BIT 3
```

Definition at line 137 of file [UART_private.h](#).

7.137.2.56 MUSART_CR3_IRLP_BIT

```
#define MUSART_CR3_IRLP_BIT 2
```

Definition at line 139 of file [UART_private.h](#).

7.137.2.57 MUSART_CR3_IREN_BIT

```
#define MUSART_CR3_IREN_BIT 1
```

Definition at line 141 of file [UART_private.h](#).

7.137.2.58 MUSART_CR3_EIE_BIT

```
#define MUSART_CR3_EIE_BIT 0
```

Definition at line 143 of file [UART_private.h](#).

7.138 UART_private.h

[Go to the documentation of this file.](#)

```
00009 /* Header file guard */
00010 #ifndef _UART_private_H
00011 #define _UART_private_H
00012
00013
00014 #define UART_DIV_SAMPLING16(_PCLK_, _BAUD_)
    ((u32_t) (((u64_t) (_PCLK_)) *25U) / (4U*((u64_t) (_BAUD_))))
00015 #define UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_)          (UART_DIV_SAMPLING16(_PCLK_, (_BAUD_))/100U)
00016 #define UART_DIVFRAQ_SAMPLING16(_PCLK_, _BAUD_)         (((UART_DIV_SAMPLING16(_PCLK_, (_BAUD_)) -
    (UART_DIVMANT_SAMPLING16(_PCLK_, (_BAUD_)) * 100U)) * 16U) + 50U) / 100U)
00017 /* UART BRR = mantissa + overflow + fraction
00018 = (UART DIVMANT << 4) + (UART DIVFRAQ & 0xF0) + (UART DIVFRAQ & 0x0FU) */
00019 #define UART_BRR_SAMPLING16(_PCLK_, _BAUD_)           ((UART_DIVMANT_SAMPLING16(_PCLK_, (_BAUD_)) <<
    4U) + \
00020 (UART_DIVFRAQ_SAMPLING16(_PCLK_, (_BAUD_)) & 0xF0U) + \
00021 (UART_DIVFRAQ_SAMPLING16(_PCLK_, (_BAUD_)) & 0x0FU))
00022
00023 #define UART_DIV_SAMPLING8(_PCLK_, _BAUD_)
    ((u32_t) (((u64_t) (_PCLK_)) *25U) / (2U*((u64_t) (_BAUD_))))
00024 #define UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_)          (UART_DIV_SAMPLING8(_PCLK_, (_BAUD_))/100U)
00025 #define UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_)         (((UART_DIV_SAMPLING8(_PCLK_, (_BAUD_)) -
    (UART_DIVMANT_SAMPLING8(_PCLK_, (_BAUD_)) * 100U)) * 8U) + 50U) / 100U)
00026 /* UART BRR = mantissa + overflow + fraction
00027 = (UART DIVMANT << 4) + ((UART DIVFRAQ & 0xF8) << 1) + (UART DIVFRAQ & 0x07U) */
00028 #define UART_BRR_SAMPLING8(_PCLK_, _BAUD_)           ((UART_DIVMANT_SAMPLING8(_PCLK_, (_BAUD_)) <<
    4U) + \
00029 ((UART_DIVFRAQ_SAMPLING8(_PCLK_, (_BAUD_)) & 0xF8U) << 1U) + \
00030 (UART_DIVFRAQ_SAMPLING8(_PCLK_, (_BAUD_)) & 0x07U))
00031
00032
00033 /*****
00034 /*          SR BITS Mapping          */
00035 /*****
00036 /*  Parity error                  */
00037 #define MUSART_SR_PE_BIT 0
00038 /*  Framing error                */
00039 #define MUSART_SR_FE_BIT 1
00040 /*  Noise error flag             */
00041 #define MUSART_SR_NE_BIT 2
00042 /*  Overrun error                */
00043 #define MUSART_SR_ORE_BIT 3
00044 /*  IDLE line detected           */
00045 #define MUSART_SR_IDLE_BIT 4
00046 /*  Read data register not empty */
00047 #define MUSART_SR_RXNE_BIT 5
00048 /*  Transmission complete        */
00049 #define MUSART_SR_TC_BIT 6
00050 /*  Transmit data register empty */
00051 #define MUSART_SR_TXE_BIT 7
00052 /*  LIN break detection flag    */
00053 #define MUSART_SR_LBD_BIT 8
00054 /*  CTS flag                     */
00055 #define MUSART_SR_CTS_BIT 9
00056
00057 /*****
00058 /*          CR1 BITS Mapping          */
00059 /*****
00060 /*  Send break bit               */
00061 #define MUSART_CRL_SBK_BIT 0
00062 /*  Recevier Wakeup bit          */
00063 #define MUSART_CRL_RWU_BIT 1
00064 /*  Recevier Enable bit          */
00065 #define MUSART_CRL_RE_BIT 2
00066 /*  Transmitter Enable bit      */
00067 #define MUSART_CRL_TE_BIT 3
00068 /*  IDLE interrupt enable bit   */
00069 #define MUSART_CRL_IDLEIE_BIT 4
00070 /*  RXNEIE interrupt enable bit */
```

```
00071 #define MUSART_CR1_RXNEIE_BIT      5
00072 /*  Transmission complete interrupt enable bit */
00073 #define MUSART_CR1_TCIE_BIT        6
00074 /*  TXE interrupt enable bit */
00075 #define MUSART_CR1_TXEIE_BIT       7
00076 /*  PE interrupt enable bit */
00077 #define MUSART_CR1_PEIE_BIT        8
00078 /*  Parity selection bit */
00079 #define MUSART_CR1_PS_BIT          9
00080 /*  Parity control enable bit */
00081 #define MUSART_CR1_PCE_BIT         10
00082 /*  Wakeup method bit */
00083 #define MUSART_CR1_WAKE_BIT        11
00084 /*  Word length bit */
00085 #define MUSART_CR1_M_BIT          12
00086 /*  USART enable bit */
00087 #define MUSART_CR1_UE_BIT          13
00088 /*  USART Oversampling bit */
00089 #define MUSART_CR1_OVER8_BIT       15
00090
00091 /*****
00092 /*          CR2 BITS Mapping */
00093 *****/
00094 /*  Address of the USART node bits */
00095 #define MUSART_CR2_ADD0_BIT        0
00096 #define MUSART_CR2_ADD1_BIT        1
00097 #define MUSART_CR2_ADD2_BIT        2
00098 #define MUSART_CR2_ADD3_BIT        3
00099 /*  lin break detection length bit */
00100 #define MUSART_CR2_LBDL_BIT        5
00101 /*  LIN break detection interrupt enable bit */
00102 #define MUSART_CR2_LBDIE_BIT       6
00103 /*  Last bit clock pulse bit */
00104 #define MUSART_CR2_LBCL_BIT        8
00105 /*  Clock phase bit */
00106 #define MUSART_CR2_CPHA_BIT       9
00107 /*  Clock polarity bit */
00108 #define MUSART_CR2_CPOL_BIT       10
00109 /*  Clock enable bit */
00110 #define MUSART_CR2_CLKEN_BIT       11
00111 /*  STOP bit start */
00112 #define MUSART_CR2_STOP_BIT        12
00113 /*  STOP bits */
00114 #define MUSART_CR2_STOP0_BIT       12
00115 #define MUSART_CR2_STOP1_BIT       13
00116 /*  LIN mode enable bit */
00117 #define MUSART_CR2_LINEN_BIT       14
00118
00119 /*****
00120 /*          CR3 BITS Mapping */
00121 *****/
00122 /*  CTS interrupt enable bit */
00123 #define MUSART_CR3_CTSIE_BIT       10
00124 /*  CTS enable bit */
00125 #define MUSART_CR3_CTSE_BIT        9
00126 /*  RTS enable bit */
00127 #define MUSART_CR3_RTSE_BIT        8
00128 /*  DMA enable transmitter bit */
00129 #define MUSART_CR3_DMAT_BIT        7
00130 /*  DMA enable receiver bit */
00131 #define MUSART_CR3_DMAR_BIT        6
00132 /*  Smartcard mode enable bit */
00133 #define MUSART_CR3_SCEN_BIT        5
00134 /*  Smartcard NACK enable bit */
00135 #define MUSART_CR3_NACK_BIT        4
00136 /*  Half-duplex selection bit */
00137 #define MUSART_CR3_HDSEL_BIT       3
00138 /*  IrDA low-power bit */
00139 #define MUSART_CR3_IRLP_BIT        2
00140 /*  IrDA mode enable bit */
00141 #define MUSART_CR3_IREN_BIT        1
00142 /*  Error interrupt enable bit */
00143 #define MUSART_CR3_EIE_BIT         0
00144
00145
00146
00147
00148
00149
00150
00151 #endif // _UART_private_H
```

7.139 UART_program.c

```

00001 /* FILENAME: UART_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 10/14/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010 #include "../../../LIB/LSTD_TYPES.h"
00011 #include "../../../LIB/LSTD_COMPILER.h"
00012 #include "../../../LIB/LSTD_VALUES.h"
00013 #include "../../../LIB/LSTD_BITMATH.h"
00014
00015 #include "string.h"
00016
00017 #include "../RCC/MRCC_interface.h"
00018 #include "../GPIO(GPIO_interface.h"
00019
00020 #include "UART_interface.h"
00021 #include "UART_private.h"
00022 #include "UART_config.h"
00023
00024 /***** Functions implementations *****/
00025 /*           Functions implementations           */
00026 /***** Functions implementations *****/
00027
00028 u8_t G_u8String[20];
00029
00030 void (*MUSART1_CallBack)(void);
00031 void (*MUSART2_CallBack)(void);
00032 void (*MUSART6_CallBack)(void);
00033
00034 /***** MUSART vInit *****/
00035 /***** MUSART vInit *****/
00036
00037 void MUSART_vInit(USART_InitType *A_InitStruct, USART_ClockInitTypeDef *A_ClockInitStruct,
                      USART_MemoryMapType *A_USARTTx)
00038 {
00039     if (A_USARTTx == USART1_REG)
00040     {
00041         MRCC_vEnablePeriphralCLK(RCC_APB2, APB2ENR_USART1EN);
00042
00043         /*configer Tx1 as alt fun*/
00044         MGPIoX_ConfigType TX1 =
00045             {
00046                 .Port = USART1_PORT, .Pin = TX1_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00047                     GPIOx_PUSHPULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00048
00049         MGPIoX_vInit(&TX1);
00050
00051         /*configer Rx1 as alt fun*/
00052         MGPIoX_ConfigType RX1 =
00053             {
00054                 .Port = USART1_PORT, .Pin = RX1_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00055                     GPIOx_PUSHPULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00056
00057         MGPIoX_vInit(&RX1);
00058     }
00059     else if (A_USARTTx == USART2_REG)
00060     {
00061         MRCC_vEnablePeriphralCLK(RCC_APB2, APB1ENR_USART2EN);
00062
00063         /*configer Tx2 as alt fun*/
00064         MGPIoX_ConfigType TX2 =
00065             {
00066                 .Port = USART2_PORT, .Pin = TX2_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00067                     GPIOx_PUSHPULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00068
00069         MGPIoX_vInit(&TX2);
00070
00071         /*configer Rx2 as alt fun*/
00072         MGPIoX_ConfigType RX2 =
00073             {
00074                 .Port = USART2_PORT, .Pin = RX2_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00075                     GPIOx_PUSHPULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7};
00076
00077         MGPIoX_vInit(&RX2);
00078     }
00079     else if (A_USARTTx == USART6_REG)
00080     {

```

```

00081     MRCC_vEnablePeripheralCLK(RCC_APB2, APB2ENR_USART6EN);
00082
00083     /*configer Tx6 as alt fun*/
00084     GPIOx_ConfigType TX6 =
00085     {
00086         .Port = USART6_PORT, .Pin = TX6_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00087         GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7;
00088     GPIOx_vInit(&TX6);
00089
00090     /*configer Rx6 as alt fun*/
00091     GPIOx_ConfigType RX6 =
00092     {
00093         .Port = USART6_PORT, .Pin = RX6_PIN, .Mode = GPIOx_MODE_AF, .OutputType =
00094         GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_MediumSpeed, .InputType = GPIOx_NoPull, .AF_Type = GPIOx_AF7;
00095     GPIOx_vInit(&RX6);
00096 }
00097
00098 // BuadRate/Over-sampling selections:
00099 switch (A_InitStruct->Oversampling)
00100 {
00101
00102 case OVER_SAMPLING_16:
00103
00104     A_USARTx->BRR_REG = UART_BRR_SAMPLING16(__PCLK__, A_InitStruct->BaudRate);
00105
00106     break;
00107
00108 case OVER_SAMPLING_8:
00109
00110     A_USARTx->BRR_REG = UART_BRR_SAMPLING16(__PCLK__, A_InitStruct->BaudRate);
00111
00112     break;
00113 }
00114
00115 A_USARTx->CR1_REG = (A_InitStruct->Oversampling << MUSART_CR1_OVER8_BIT) |
00116     (A_InitStruct->DataWidth << MUSART_CR1_M_BIT) |
00117     (A_InitStruct->Parity_Enable << MUSART_CR1_PCE_BIT) |
00118     (A_InitStruct->Parity_Selection << MUSART_CR1_PS_BIT);
00119
00120 switch (A_InitStruct->TransferDirection)
00121 {
00122 case TX_ONLY:
00123
00124     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_TE_BIT);
00125
00126     break;
00127
00128 case RX_ONLY:
00129
00130     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_RE_BIT);
00131
00132     break;
00133
00134 case TX_RX:
00135
00136     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_TE_BIT);
00137     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_RE_BIT);
00138
00139     break;
00140 }
00141
00142 A_USARTx->CR2_REG = (A_InitStruct->StopBits << MUSART_CR2_STOP_BIT) |
00143     (A_ClockInitStruct->ClockOutput << MUSART_CR2_CLKEN_BIT) |
00144     (A_ClockInitStruct->ClockPhase << MUSART_CR2_CPHA_BIT) |
00145     (A_ClockInitStruct->ClockPolarity << MUSART_CR2_CPOL_BIT) |
00146     (A_ClockInitStruct->LastBitClockPulse << MUSART_CR2_LBCL_BIT);
00147
00148 A_USARTx->SR_REG = 0; // Clear all the flags.
00149
00150 SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_UE_BIT); // EN the peripheral.
00151 }
00152
00153 //*****
00154 //*****
00155
00156 void MUSART_vEnable(USART_MemoryMapType *A_USARTx)
00157 {
00158     SET_BIT(A_USARTx->CR1_REG, MUSART_CR1_UE_BIT);
00159 }
00160
00161 //*****
00162 //*****
00163
00164 void MUSART_vDisable(USART_MemoryMapType *A_USARTx)
00165 {

```

```

00166     CLR_BIT(A_USARTx->CR1_REG, MUSART_CR1_UE_BIT);
00167 }
00168
00169 /*****
00170 *****/
00171
00172 void MUSART_vTransmitByte(USART_MemoryMapType *A_USARTx, u8_t A_u8Byte)
00173 {
00174     while (GET_BIT(A_USARTx->SR_REG, MUSART_SR_TXE_BIT) == 0)
00175         ;
00176
00177     A_USARTx->DR_REG = A_u8Byte;
00178
00179     while (GET_BIT(A_USARTx->SR_REG, MUSART_SR_TC_BIT) == 0)
00180         ;
00181
00182     // Clear the TC flag:
00183     CLR_BIT(A_USARTx->SR_REG, MUSART_SR_TC_BIT);
00184
00185 }
00186
00187 /*****
00188 *****/
00189
00190 void MUSART_vTransmitString(USART_MemoryMapType *A_USARTx, c8_t *A_ptrc8String)
00191 {
00192     u8_t loc_u8Iterator = INITIAL_ZERO;
00193
00194     while (A_ptrc8String[loc_u8Iterator] != '\0')
00195     {
00196
00197         MUSART_vTransmitByte(A_USARTx, A_ptrc8String[loc_u8Iterator]);
00198
00199         loc_u8Iterator++;
00200     }
00201 }
00202
00203
00204 /*****
00205 *****/
00206
00207 u8_t MUSART_u8ReceiveByteSyncNonBlocking(USART_MemoryMapType *A_USARTx)
00208 {
00209     u8_t loc_u8Data = INITIAL_ZERO;
00210
00211     u32_t loc_u8TimeOut = INITIAL_ZERO;
00212
00213     while ((GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0) && (loc_u8TimeOut < THRESHOLD_VALUE))
00214     {
00215         loc_u8TimeOut++;
00216     }
00217
00218     if (loc_u8TimeOut == THRESHOLD_VALUE)
00219     {
00220         loc_u8Data = 255; // To detect the error.
00221     }
00222     else
00223     {
00224         loc_u8Data = A_USARTx->DR_REG;
00225     }
00226
00227     return loc_u8Data;
00228 }
00229
00230
00231 /*****
00232 *****/
00233
00234 u8_t MUSART_u8ReceiveByteSyncBlocking(USART_MemoryMapType *A_USARTx)
00235 {
00236
00237     while (GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0)
00238         ;
00239
00240     return A_USARTx->DR_REG;
00241 }
00242
00243 /*****
00244 *****/
00245
00246 u8_t *MUSART_ptrReceiveStringSyncNonBlocking(USART_MemoryMapType *A_USARTx)
00247 {
00248     u8_t loc_u8Iterator = INITIAL_ZERO;
00249
00250     u8_t loc_u8DataCome;
00251
00252

```

```

00253     while ((loc_u8DataCome = MUSART_u8ReceiveByteSynchNonBlocking(A_USARTx)) != 13)
00254     {
00255         G_u8String[loc_u8Iterator] = loc_u8DataCome;
00256         loc_u8Iterator++;
00257     }
00258     G_u8String[loc_u8Iterator] = '\0';
00259
00260     return (G_u8String);
00261
00262 }
00263
00264
00265 /*****
00266 *****/
00267
00268 void MUSART_vRecieveString(USART_MemoryMapType *A_USARTx, c8_t A_c8YourString[])
00269 {
00270
00271     u32_t L_u32Counter = INITIAL_ZERO;
00272
00273     A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking(A_USARTx);
00274
00275     while (A_c8YourString[L_u32Counter] != '\0')
00276     {
00277
00278         L_u32Counter++;
00279
00280         A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking(A_USARTx);
00281
00282         if ((A_c8YourString[L_u32Counter] == '\n') || (A_c8YourString[L_u32Counter] == '\r'))
00283         {
00284             A_c8YourString[L_u32Counter] = '\0';
00285         }
00286     }
00287 }
00288
00289 /*****
00290 *****/
00291
00292 u8_t MUSART_u8CompareString(c8_t *String1, c8_t *String2)
00293 {
00294
00295     u8_t L_u8TheComparingResult = INITIAL_ZERO;
00296
00297     if (strcmp(String1, String2) == SAME_STRING)
00298     {
00299         L_u8TheComparingResult = SAME_STRING;
00300     }
00301     else
00302     {
00303         L_u8TheComparingResult = DIFFERENT_STRING;
00304     }
00305
00306     return L_u8TheComparingResult;
00307 }
00308
00309 /*****
00310 *****/
00311
00312 u8_t MUSART_u8ReadDataRegister(USART_MemoryMapType *A_USARTx)
00313 {
00314     return A_USARTx->DR_REG;
00315 }
00316
00317 /*****
00318 *****/
00319
00320 void MUSART_vClearFlags(USART_MemoryMapType *A_USARTx)
00321 {
00322     A_USARTx->SR_REG = 0;
00323 }
00324
00325 /*****
00326 *****/
00327
00328 void MUSART_vRxIntSetStatus(USART_MemoryMapType *A_USARTx, u8_t A_u8Status)
00329 {
00330
00331     switch (A_u8Status)
00332     {
00333
00334     case ENABLE:
00335         SET_BIT(A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT));
00336         break;
00337
00338     case DISABLE:
00339         CLR_BIT(A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT));

```

```

00340         break;
00341     }
00342 }
00343
00344 /*****
00345 ****/
00346
00347 void MUSART1_vSetCallBack(void (*Fptr) (void))
00348 {
00349     MUSART1_CallBack = Fptr;
00350 }
00351
00352 ****
00353 ****
00354
00355 void MUSART2_vSetCallBack(void (*Fptr) (void))
00356 {
00357     MUSART2_CallBack = Fptr;
00358 }
00359
00360 ****
00361 ****
00362
00363 void MUSART6_vSetCallBack(void (*Fptr) (void))
00364 {
00365     MUSART6_CallBack = Fptr;
00366 }
00367
00368 ****
00369 ****
00370
00371 void USART1_IRQHandler(void)
00372 {
00373
00374     if (MUSART1_CallBack != NULL)
00375     {
00376         MUSART1_CallBack();
00377     }
00378 }
00379
00380 ****
00381 ****
00382
00383 void USART2_IRQHandler(void)
00384 {
00385
00386     USART2_REG->SR_REG = 0;
00387
00388     MUSART2_CallBack();
00389 }
00390
00391 ****
00392 ****
00393
00394 void USART6_IRQHandler(void)
00395 {
00396
00397     USART6_REG->SR_REG = 0;
00398
00399     MUSART6_CallBack();
00400 }
00401
00402 ****
00403 ****

```

7.140 MyRTOS_config.h

```

00001 /* FILENAME: RTOS_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 09/07/2022
00005 */
00006 #ifndef _RTOS_config_H
00007 #define _RTOS_config_H
00008
00009
00010 typedef enum
00011 {
00012
00013     NO_ERROR           ,
00014
00015     PRIORITY_ERROR    ,
00016

```

```

00017     TASK_RESERVED_ERROR ,
00018
00019     TASK_EMPTY_ERROR      ,
00020
00021     TASK_SUSP_ERROR
00022
00023 } Error_enState;
00024
00025 typedef enum
00026 {
00027
00028     EMPTY      ,
00029
00030     NOT_EMPTY  ,
00031
00032 } Priority_enState;
00033
00034
00035 /* Write the maximum and minimum number of tasks in your container system */
00036 #define MAX_TASK_NUM        20U
00037 #define MIN_TASK_NUM        0
00038
00039 /* Define your Tick time */
00040 #define RTOS_TICK_TIME       100000U /* 100 MilliSecond */
00041
00042
00043 #endif ///_RTOS_config_H

```

7.141 MyRTOS_interface.h

```

00001 /* FILENAME: RTOS_interface
00002 * Author:    Ali El Bana
00003 * Version:   V1.0
00004 * DATE:     Wed 09/07/2022
00005 */
00006 #ifndef _RTOS_interface_H
00007 define _RTOS_interface_H
00008
00009 #include "MyRTOS_config.h"
00010
00011 *****
00012 /*          Functions prototypes           */
00013 *****
00014
00015 Error_enState RTOS_u8CreateTask( void A_CopyHandler(void), u8_t A_u8Periodicity,
00016                                     u8_t A_u8Priority, u8_t A_u8FirstDelay ) ;
00017
00018 void Scheduler( void ) ;
00019
00020 void RTOS_vStartOS( void ) ;
00021
00022 Error_enState RTOS_u8DeleteTask( u8_t A_u8Priority )      ;
00023
00024 Error_enState RTOS_u8SuspendTask( u8_t A_u8Priority )      ;
00025
00026 Error_enState RTOS_u8ResumeTask( u8_t A_u8Priority )      ;
00027
00028
00029 #endif ///_RTOS_interface_H

```

7.142 MyRTOS_private.h

```

00001 /* FILENAME: RTOS_private
00002 * Author:    Ali El-Bana
00003 * Version:   V1.0
00004 * DATE:     Wed 09/07/2022
00005 * Description: Private file for OS module
00006 */
00007 #ifndef _RTOS_private_H
00008 define _RTOS_private_H
00009
00010
00011 typedef enum
00012 {
00013
00014     SUSPENDED ,
00015
00016     RUNNING
00017

```

```

00018 } Running_enState;
00019
00020 typedef struct
00021 {
00022     void (*TaskHandler) ( void )          ;
00024     u8_t periodicity                  ;
00026     Running_enState Task_RunningState ;
00028
00029 } Task ;
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042 #endif // _RTOS_private_H

```

7.143 MyRTOS_program.c

```

00001 /* FILENAME: RTOS_program
00002 * Author:    Ali El Bana
00003 * Version:   V1.0
00004 * DATE:     Wed 09/07/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Include headers           */
00009 /***** Include headers *****/
00010
00011 #include "../../COTS/LIB/LSTD_TYPES.h"
00012 #include "../../COTS/LIB/LSTD_COMPILER.h"
00013 #include "../../COTS/LIB/LSTD_VALUES.h"
00014 #include "../../COTS/LIB/LSTD_BITMATH.h"
00015
00016 #include "../../../../COTS/MCAL/RCC/MRCC_interface.h"
00017 #include "../../../../COTS/MCAL/GPIO/GPIO_interface.h"
00018 #include "../../../../COTS/MCAL/SysTick/SysTick_interface.h"
00019
00020 #include "MyRTOS_config.h"
00021 #include "MyRTOS_interface.h"
00022 #include "MyRTOS_private.h"
00023
00024
00025
00026 Task SystemTasks[ MAX_TASK_NUM ]      = { INITIAL_ZERO } ;
00027
00028 Task Empty_TaskSystem                = { INITIAL_ZERO } ;
00029
00030 u8_t TaskTiming[ MAX_TASK_NUM ]      = { INITIAL_ZERO } ;
00031
00032 /***** Functions implementations *****/
00033 /*           Functions implementations           */
00034 /***** Functions implementations *****/
00035
00036 Error_enState RTOS_u8CreateTask( void A_CopyHandler(void), u8_t A_u8Periodicity,
00037                                     u8_t A_u8Priority        , u8_t A_u8FirstDelay )
00038 {
00039
00040     Error_enState L_enuTaskErrorState = NO_ERROR ;
00041
00042     /* Verify that the priority index within the correct range */
00043     if( (A_u8Priority >= MIN_TASK_NUM) && (A_u8Priority < MAX_TASK_NUM) )
00044     {
00045
00046         /* Verify that the required priority is free */
00047         if( SystemTasks[A_u8Priority].TaskHandler == EMPTY )
00048         {
00049
00050             SystemTasks[A_u8Priority].TaskHandler      = A_CopyHandler      ;
00051             SystemTasks[A_u8Priority].periodicity       = A_u8Periodicity    ;
00052             TaskTiming[A_u8Priority]                   = A_u8FirstDelay     ;
00053
00054             TaskTiming[A_u8Priority]                   = A_u8FirstDelay     ;
00055

```

```

00056     SystemTasks[A_u8Priority].Task_RunningState = RUNNING ;
00057
00058 }
00059 else
00060 {
00061     /* This task is already reserved */
00062     L_enuTaskErrorState = TASK_RESERVED_ERROR ;
00063 }
00064
00065 }
00066 else
00067 {
00068     /* Wrong priority number */
00069     L_enuTaskErrorState = PRIORITY_ERROR ;
00070 }
00071
00072 return L_enuTaskErrorState ;
00073
00074 }
00075
00076
00077 /*****
00078 *****/
00079 void Scheduler( void )
00080 {
00081
00082     u8_t L_u8Counter = INITIAL_ZERO ;
00083
00084     /* Loop on all tasks */
00085     for( L_u8Counter = MIN_TASK_NUM; L_u8Counter < MAX_TASK_NUM; L_u8Counter++ )
00086     {
00087
00088         /* Verify that the required priority is reserved */
00089         if( SystemTasks[L_u8Counter].TaskHandler != EMPTY )
00090         {
00091
00092             if( SystemTasks[L_u8Counter].Task_RunningState == RUNNING )
00093             {
00094
00095                 /* Verify that it is the tick time to execute it */
00096                 if( TaskTiming[L_u8Counter] == 0 )
00097                 {
00098                     /* Run the task */
00099                     SystemTasks[L_u8Counter].TaskHandler( ) ;
00100
00101                     /* Reload the periodicity */
00102                     TaskTiming[L_u8Counter] = SystemTasks[L_u8Counter].periodicity ;
00103
00104                 else
00105                 {
00106                     /* Decrement the timing until it reach to the tick time to be executed */
00107                     TaskTiming[L_u8Counter]--;
00108
00109                 }
00110
00111             else
00112             {
00113                 /* Do nothing */
00114             }
00115
00116         }
00117
00118     else
00119     {
00120         /* Do nothing */
00121     }
00122 }
00123
00124 }
00125
00126
00127 /*****
00128 *****/
00129 Error_enState RTOS_u8DeleteTask( u8_t A_u8Priority )
00130 {
00131     Error_enState L_enuErrorState = NO_ERROR ;
00132
00133     /* Verify that the required priority is registered */
00134     if( SystemTasks[A_u8Priority].TaskHandler != EMPTY )
00135     {
00136
00137         /* Task exist, execute the removal */
00138         SystemTasks[A_u8Priority] = Empty_TaskSystem ;

```

```

00139     }
00140   }
00141   else
00142   {
00143     /* Task not exist */
00144     L_enErrorState = TASK_EMPTY_ERROR ;
00145   }
00146
00147   return L_enErrorState ;
00148
00149 }
00150
00151 /*****
00152 *****/
00153
00154 Error_enState RTOS_u8SuspendTask( u8_t A_u8Priority )
00155 {
00156
00157   Error_enState L_enErrorState = NO_ERROR ;
00158
00159   /* Verify that the required priority is registered */
00160   if( SystemTasks[A_u8Priority].TaskHandler != EMPTY )
00161   {
00162
00163     /* Task exist, suspend it */
00164     SystemTasks[A_u8Priority].Task_RunningState == SUSPENDED ;
00165
00166   }
00167   else
00168   {
00169     /* Task not exist */
00170     L_enErrorState = TASK_EMPTY_ERROR ;
00171   }
00172
00173   return L_enErrorState ;
00174
00175 }
00176
00177 /*****
00178 *****/
00179
00180 Error_enState RTOS_u8ResumeTask( u8_t A_u8Priority )
00181 {
00182
00183   Error_enState L_enErrorState = NO_ERROR ;
00184
00185   /* Verify that the required priority is registered */
00186   if( SystemTasks[A_u8Priority].TaskHandler != EMPTY )
00187   {
00188
00189     /* Task exist, run it */
00190     SystemTasks[A_u8Priority].Task_RunningState == RUNNING ;
00191
00192   }
00193   else
00194   {
00195     /* Task not exist */
00196     L_enErrorState = TASK_EMPTY_ERROR ;
00197   }
00198
00199   return L_enErrorState ;
00200
00201 }
00202
00203 /*****
00204 *****/
00205
00206 void RTOS_vStartOS( void )
00207 {
00208
00209   MSysTick_vSetPeriodicInterval( RTOS_TICK_TIME, &Scheduler ) ;
00210
00211 }
00212
00213 /*****
00214 *****/
00215
00216
00217

```

00218
00219
00220

Index

_16GROUP_NoSub_Priorities
 Interrupt priority grouping, 62

_2GROUP_8Sub_Priorities
 Interrupt priority grouping, 63

_4GROUP_4Sub_Priorities
 Interrupt priority grouping, 62

_8GROUP_2Sub_Priorities
 Interrupt priority grouping, 62

_BAUDRATE
 UART_interface.h, 544

ADC
 NVIC Vector Table, 82

ADC Channels, 29
 CHANNEL0, 29
 CHANNEL1, 30
 CHANNEL10, 32
 CHANNEL11, 32
 CHANNEL12, 32
 CHANNEL13, 33
 CHANNEL14, 33
 CHANNEL15, 33
 CHANNEL16, 33
 CHANNEL17, 34
 CHANNEL18, 34
 CHANNEL2, 30
 CHANNEL3, 30
 CHANNEL4, 30
 CHANNEL5, 31
 CHANNEL6, 31
 CHANNEL7, 31
 CHANNEL8, 31
 CHANNEL9, 32

ADC1EN
 MRCC_config.h, 381

ADC_interface.h
 MADC_u16ConvertToDigital, 269
 MADC_vDisable, 270
 MADC_vEnable, 270
 MADC_vInit, 265
 MADC_vRegINT_Disable, 270
 MADC_vRegINTEnable, 271
 MADC_vSelectChannel, 271
 MADC_vSetCallBack, 271

ADC_MemoryMapType, 125
 CCR, 130
 CR1, 126
 CR2, 126
 DR, 130
 HTR, 128

JDR1, 129
JDR2, 129
JDR3, 129
JDR4, 130

JOFR1, 127
JOFR2, 127
JOFR3, 127
JOFR4, 128

JSQR, 129
LTR, 128

SMPR1, 127
SMPR2, 127

SQR1, 128
SQR2, 128
SQR3, 129

SR, 126

AF_Type
 GPIOx_ConfigType, 144

AFRHx
 GPIOx_MemoryMapType, 139

AFRLx
 GPIOx_MemoryMapType, 139

AHB
 Systick Clock Sources, 117

AHB1ENR
 RCC_MemoryMapType, 154

AHB1ENR_CRCEN
 ID options, 99

AHB1ENR_DMA1EN
 ID options, 99

AHB1ENR_DMA2EN
 ID options, 99

AHB1ENR_GPIOAEN
 ID options, 101

AHB1ENR_GPIOBEN
 ID options, 101

AHB1ENR_GPIOCEN
 ID options, 100

AHB1ENR_GPIODEN
 ID options, 100

AHB1ENR_GPIOEEN
 ID options, 100

AHB1ENR_GPIOHEN
 ID options, 100

AHB1LPENR
 RCC_MemoryMapType, 156

AHB1LPENR_FLITFLPEN
 ID options, 107

AHB1RSTR

RCC_MemoryMapType, 153
 AHB2ENR
 RCC_MemoryMapType, 155
 AHB2ENR_OTGFSEN
 ID options, 101
 AHB2LPENR
 RCC_MemoryMapType, 156
 AHB2RSTR
 RCC_MemoryMapType, 153
 AHB_DividedBy8
 Systick Clock Sources, 116
 AHBby16
 MRCC_private.h, 430
 AHBby2
 MRCC_private.h, 430
 AHBby4
 MRCC_private.h, 430
 AHBby8
 MRCC_private.h, 430
 AIRCR
 SCB_MemoryMapType, 161
 APB1ENR
 RCC_MemoryMapType, 155
 APB1ENR_I2C1EN
 ID options, 102
 APB1ENR_I2C2EN
 ID options, 102
 APB1ENR_I2C3EN
 ID options, 102
 APB1ENR_PWREN
 ID options, 101
 APB1ENR_SPI2EN
 ID options, 103
 APB1ENR_SPI3EN
 ID options, 103
 APB1ENR_TIM2EN
 ID options, 104
 APB1ENR_TIM3EN
 ID options, 104
 APB1ENR_TIM4EN
 ID options, 104
 APB1ENR_TIM5EN
 ID options, 103
 APB1ENR_USART2EN
 ID options, 102
 APB1ENR_WWDGEN
 ID options, 103
 APB1LPENR
 RCC_MemoryMapType, 157
 APB1RSTR
 RCC_MemoryMapType, 154
 APB2ENR
 RCC_MemoryMapType, 155
 APB2ENR_ADC1EN
 ID options, 106
 APB2ENR_SDIOEN
 ID options, 106
 APB2ENR_SPI1EN

 ID options, 106
 APB2ENR_SPI4EN
 ID options, 105
 APB2ENR_SYSCFGEN
 ID options, 105
 APB2ENR_TIM10EN
 ID options, 105
 APB2ENR_TIM11EN
 ID options, 104
 APB2ENR_TIM1EN
 ID options, 107
 APB2ENR_TIM9EN
 ID options, 105
 APB2ENR_USART1EN
 ID options, 107
 APB2ENR_USART6EN
 ID options, 106
 APB2LPENR
 RCC_MemoryMapType, 157
 APB2RSTR
 RCC_MemoryMapType, 154
 ARR
 TIM1_MemoryMapType, 172
 AssertRequest
 Systick Exception (Interrupt) Status, 117
 BACKWARD
 DC Motor Directions, 11
 BaudRate
 USART_InitType, 176
 BDCR
 RCC_MemoryMapType, 158
 BDTR
 TIM1_MemoryMapType, 173
 BFAR
 SCB_MemoryMapType, 165
 Bit Group Manipulation Math Macros, 17
 CLR_BITs, 17
 GET_BITs, 18
 SET_BITs, 17
 TOGGLE_BITs, 18
 Bit Manipulation Math Macros, 16
 CLR_BIT, 16
 GET_BIT, 17
 SET_BIT, 16
 TOGGLE_BIT, 16
 bool_t
 Standard types, 22
 BRR_REG
 USART_MemoryMapType, 179
 BSRRx
 GPIOx_MemoryMapType, 139
 BUS_FAULT
 NVIC Settable Priorities, 95
 BUSY_TICK_TIME
 SysTick_interface.h, 475
 BUZZER_BuzzerConfiguration, 130
 u8Pin, 131
 u8Port, 131

BUZZER_interface.h
 HBUZZER_vInit, 203
 HBUZZER_vSoundOff, 204
 HBUZZER_vSoundOn, 203
 HBUZZER_vToggleSound, 204

BUZZER_program.c
 HBUZZER_vInit, 206
 HBUZZER_vSoundOff, 207
 HBUZZER_vSoundOn, 207
 HBUZZER_vToggleSound, 207

BYBASED
 MRCC_private.h, 421

c8_t
 Standard types, 23

CALIB
 SysTick_Type, 169

CCER
 TIM1_MemoryMapType, 172

CCMR1
 TIM1_MemoryMapType, 172

CCMR2
 TIM1_MemoryMapType, 172

CCR
 ADC_MemoryMapType, 130
 SCB_MemoryMapType, 162

CCR1
 TIM1_MemoryMapType, 173

CCR2
 TIM1_MemoryMapType, 173

CCR3
 TIM1_MemoryMapType, 173

CCR4
 TIM1_MemoryMapType, 173

CCW
 DC Motor Rotation Directions, 15

CFG_R
 RCC_MemoryMapType, 152

CFSR
 SCB_MemoryMapType, 163

CHANNEL0
 ADC Channels, 29

CHANNEL1
 ADC Channels, 30

CHANNEL10
 ADC Channels, 32

CHANNEL11
 ADC Channels, 32

CHANNEL12
 ADC Channels, 32

CHANNEL13
 ADC Channels, 33

CHANNEL14
 ADC Channels, 33

CHANNEL15
 ADC Channels, 33

CHANNEL16
 ADC Channels, 33

CHANNEL17

 ADC Channels, 34

 CHANNEL18
 ADC Channels, 34

 CHANNEL2
 ADC Channels, 30

 CHANNEL3
 ADC Channels, 30

 CHANNEL4
 ADC Channels, 30

 CHANNEL5
 ADC Channels, 31

 CHANNEL6
 ADC Channels, 31

 CHANNEL7
 ADC Channels, 31

 CHANNEL8
 ADC Channels, 31

 CHANNEL9
 ADC Channels, 32

 CIR
 RCC_MemoryMapType, 153

 CLK_SOURCE
 Systick Configuration, 111

 CLKSOURCE
 Systick Register Bits Positions, 115

 ClockOutput
 USART_ClockInitTypeDef, 175

 ClockPhase
 USART_ClockInitTypeDef, 175

 ClockPolarity
 USART_ClockInitTypeDef, 175

 CLR_BIT
 Bit Manipulation Math Macros, 16

 CLR_BITS
 Bit Group Manipulation Math Macros, 17

 CMPCR
 MSYSCFG_MemMap_t, 145

 CNT
 TIM1_MemoryMapType, 172

 Compiler standard macros, 19

 CONST, 21

 CONSTP2CONST, 20

 CONSTP2VAR, 20

 FUNC, 19

 P2CONST, 20

 P2FUNC, 20

 P2VAR, 19

 STATIC, 21

 VAR, 19

 CONST
 Compiler standard macros, 21

 CONSTP2CONST
 Compiler standard macros, 20

 CONSTP2VAR
 Compiler standard macros, 20

 COTS/APP/ACC/ACC_config.h, 181

 COTS/APP/ACC/ACC_interface.h, 181

 COTS/APP/ACC/ACC_private.h, 181

COTS/APP/ACC/ACC_program.c, 182
COTS/APP/Exit_State/Exit_State_config.h, 184
COTS/APP/Exit_State/Exit_State_interface.h, 184
COTS/APP/Exit_State/Exit_State_private.h, 184
COTS/APP/Exit_State/Exit_State_program.c, 185
COTS/APP/FCW/FCW_config.h, 186
COTS/APP/FCW/FCW_interface.h, 186
COTS/APP/FCW/FCW_private.h, 186
COTS/APP/FCW/FCW_program.c, 186
COTS/APP/Mob_APP/Mob_APP_config.h, 189
COTS/APP/Mob_APP/Mob_APP_interface.h, 190
COTS/APP/Mob_APP/Mob_APP_private.h, 190
COTS/APP/Mob_APP/Mob_APP_program.c, 190
COTS/APP/NCC/NCC_config.h, 192
COTS/APP/NCC/NCC_interface.h, 192
COTS/APP/NCC/NCC_private.h, 193
COTS/APP/NCC/NCC_program.c, 193
COTS/APP/Traditional_Mode/Traditional_Mode_config.h, 195
COTS/APP/Traditional_Mode/Traditional_Mode_interface.h, 195
COTS/APP/Traditional_Mode/Traditional_Mode_private.h, 195
COTS/APP/Traditional_Mode/Traditional_Mode_program.c, 195
COTS/HAL/Bluetooth/Bluetooth_config.c, 197
COTS/HAL/Bluetooth/Bluetooth_config.h, 198
COTS/HAL/Bluetooth/Bluetooth_interface.h, 198
COTS/HAL/Bluetooth/Bluetooth_private.h, 199
COTS/HAL/Bluetooth/Bluetooth_program.c, 199
COTS/HAL/BUZZER/BUZZER_config.h, 202
COTS/HAL/BUZZER/BUZZER_interface.h, 202, 205
COTS/HAL/BUZZER/BUZZER_private.h, 205
COTS/HAL/BUZZER/BUZZER_program.c, 205, 208
COTS/HAL/Car_Movement/Car_Movement_config.h, 209
COTS/HAL/Car_Movement/Car_Movement_interface.h, 209
COTS/HAL/Car_Movement/Car_Movement_private.h, 210
COTS/HAL/Car_Movement/Car_Movement_program.c, 210
COTS/HAL/DCMOTOR/DCM_config.h, 212, 213
COTS/HAL/DCMOTOR/DCM_interface.h, 213, 218
COTS/HAL/DCMOTOR/DCM_private.h, 219, 220
COTS/HAL/DCMOTOR/DCM_program.c, 220, 225
COTS/HAL/LCD/LCD_config.h, 227
COTS/HAL/LCD/LCD_interface.h, 228
COTS/HAL/LCD/LCD_private.h, 229
COTS/HAL/LCD/LCD_program.c, 230
COTS/HAL/LDR/LDR_config.c, 236
COTS/HAL/LDR/LDR_config.h, 237
COTS/HAL/LDR/LDR_interface.h, 237
COTS/HAL/LDR/LDR_private.h, 238
COTS/HAL/LDR/LDR_program.c, 238
COTS/HAL/LED/LED_config.h, 239
COTS/HAL/LED/LED_interface.h, 239
COTS/HAL/LED/LED_private.h, 240
COTS/HAL/LED/LED_program.c, 240
COTS/HAL/TFT/TFT_Config.c, 241
COTS/HAL/TFT/TFT_config.h, 242
COTS/HAL/TFT/TFT_interface.h, 242
COTS/HAL/TFT/TFT_private.h, 243
COTS/HAL/TFT/TFT_program.c, 243
COTS/HAL/UltraSonic/UltraSonic_config.h, 246
COTS/HAL/UltraSonic/UltraSonic_interface.h, 246, 249
COTS/HAL/UltraSonic/UltraSonic_private.h, 249
COTS/HAL/UltraSonic/UltraSonic_program.c, 250
COTS/LIB/LSTD_BITMATH.h, 251, 252
COTS/LIB/LSTD_COMPILER.h, 252, 253
COTS/LIB/LSTD MCU UTILITIES.h, 253, 254
COTS/LIB/LSTD_TYPES.h, 254, 255
COTS/LIB/LSTD_VALUES.h, 255, 256
COTS/MCAL/ADC/ADC_config.h, 257
COTS/MCAL/ADC/ADC_interface.h, 264, 272
COTS/MCAL/ADC/ADC_private.h, 273
COTS/MCAL/ADC/ADC_program.c, 278
COTS/MCAL/EXTI/EXTI_config.h, 284
COTS/MCAL/EXTI/EXTI_interface.h, 286, 294
COTS/MCAL/EXTI/EXTI_private.h, 295, 296
COTS/MCAL/EXTI/EXTI_program.c, 297, 307
COTS/MCAL/EXTI/SYSCFG_private.h, 312, 313
COTS/MCAL/GPIO/GPIO_config.h, 313, 315
COTS/MCAL/GPIO/GPIO_interface.h, 315, 329
COTS/MCAL/GPIO/GPIO_private.h, 331, 332
COTS/MCAL/GPIO/GPIO_program.c, 332, 343
COTS/MCAL/NVIC/NVIC_config.h, 349
COTS/MCAL/NVIC/NVIC_interface.h, 349, 359
COTS/MCAL/NVIC/NVIC_private.h, 360, 362
COTS/MCAL/NVIC/NVIC_program.c, 363, 369
COTS/MCAL/RCC/MRCC_config.h, 371, 382
COTS/MCAL/RCC/MRCC_interface.h, 389, 398
COTS/MCAL/RCC/MRCC_private.h, 399, 442
COTS/MCAL/RCC/MRCC_program.c, 446, 453
COTS/MCAL/SPI/SPI_config.h, 459
COTS/MCAL/SPI/SPI_interface.h, 462
COTS/MCAL/SPI/SPI_private.h, 463
COTS/MCAL/SPI/SPI_program.c, 465
COTS/MCAL/SysTick/SysTick_config.h, 473
COTS/MCAL/SysTick/SysTick_interface.h, 474, 484
COTS/MCAL/SysTick/SysTick_private.h, 485, 486
COTS/MCAL/SysTick/SysTick_program.c, 486, 496
COTS/MCAL/TIM1/TIM1_config.h, 501
COTS/MCAL/TIM1/TIM1_interface.h, 510
COTS/MCAL/TIM1/TIM1_private.h, 511
COTS/MCAL/TIM1/TIM1_program.c, 517
COTS/MCAL/UART/UART_config.h, 540
COTS/MCAL/UART/UART_interface.h, 542, 554
COTS/MCAL/UART/UART_private.h, 556, 568
COTS/MCAL/UART/UART_program.c, 570
COTS/MyRTOS/MyRTOS_config.h, 574
COTS/MyRTOS/MyRTOS_interface.h, 575
COTS/MyRTOS/MyRTOS_private.h, 575
COTS/MyRTOS/MyRTOS_program.c, 576
COUNTER_ENABLE
Systick Register Bits Positions, 116

COUNTFLAG
 Systick Register Bits Positions, 115

CPUID
 SCB_MemoryMapType, 160

CR
 RCC_MemoryMapType, 152

CR1
 ADC_MemoryMapType, 126
 SPI_MemoryMapType, 166
 TIM1_MemoryMapType, 171

CR1_REG
 USART_MemoryMapType, 180

CR2
 ADC_MemoryMapType, 126
 SPI_MemoryMapType, 166
 TIM1_MemoryMapType, 171

CR2_REG
 USART_MemoryMapType, 180

CR3_REG
 USART_MemoryMapType, 180

CRCCLK
 MRCC_config.h, 376

CRCPR
 SPI_MemoryMapType, 166

CSR
 RCC_MemoryMapType, 158

CSS
 MRCC_config.h, 373

CTRL
 SysTick_Type, 168

CW
 DC Motor Rotation Directions, 15

DataWidth
 USART_InitType, 176

DC Motor Directions, 11
 BACKWARD, 11
 FORWARD, 11

DC Motor Rotation Directions, 15
 CCW, 15
 CW, 15

DC Motor Speeds, 12
 SPEED_0_PERCENT, 12
 SPEED_100_PERCENT, 15
 SPEED_10_PERCENT, 12
 SPEED_20_PERCENT, 13
 SPEED_30_PERCENT, 13
 SPEED_40_PERCENT, 13
 SPEED_50_PERCENT, 13
 SPEED_60_PERCENT, 14
 SPEED_70_PERCENT, 14
 SPEED_80_PERCENT, 14
 SPEED_90_PERCENT, 14

DCKCFG
 RCC_MemoryMapType, 159

DCM_config.h
 MAX_SPEED, 212

DCM_interface.h
 HDCM_vGetSpeedValue, 217

HDCM_vInitMotor, 214
HDCM_vMotorSpeedCntrl, 217
HDCM_vMoveBackward, 216
HDCM_vMoveForward, 215
HDCM_vStopMotor, 216

DCM_MotorConfiguration, 132
 u32SpeedRatio, 133
 u8Direction, 133
 u8Pin1, 132
 u8Pin2, 133
 u8Port, 132
 u8SpeedPin, 133

DCM_program.c
 HDCM_vGetSpeedValue, 224
 HDCM_vInitMotor, 221
 HDCM_vMotorSpeedCntrl, 223
 HDCM_vMoveBackward, 222
 HDCM_vMoveForward, 222
 HDCM_vStopMotor, 223

DCR
 TIM1_MemoryMapType, 174

Delay Units, 113
 MICRO_SEC, 113
 MILLI_SEC, 113
 SEC, 113

DIER
 TIM1_MemoryMapType, 171

DIFFERENT_STRING
 Standard values, 28

DISABLE
 EXTI Lines Status, 42
 MRCC_private.h, 421
 UART_interface.h, 544

DivisionBy2
 MRCC_private.h, 422

DivisionBy3
 MRCC_private.h, 422

DivisionBy4
 MRCC_private.h, 422

DivisionBy5
 MRCC_private.h, 422

DMA1_STREAM0
 NVIC Vector Table, 81

DMA1_STREAM1
 NVIC Vector Table, 81

DMA1_STREAM2
 NVIC Vector Table, 81

DMA1_STREAM3
 NVIC Vector Table, 81

DMA1_STREAM4
 NVIC Vector Table, 82

DMA1_STREAM5
 NVIC Vector Table, 82

DMA1_STREAM6
 NVIC Vector Table, 82

DMA1_STREAM7
 NVIC Vector Table, 87

DMA1CLK

MRCC_config.h, 376
 DMA2_STREAM0
 NVIC Vector Table, 88
 DMA2_STREAM1
 NVIC Vector Table, 89
 DMA2_STREAM2
 NVIC Vector Table, 89
 DMA2_STREAM3
 NVIC Vector Table, 89
 DMA2_STREAM4
 NVIC Vector Table, 89
 DMA2_STREAM5
 NVIC Vector Table, 90
 DMA2_STREAM6
 NVIC Vector Table, 90
 DMA2_STREAM7
 NVIC Vector Table, 90
 DMA2CLK
 MRCC_config.h, 376
 DMAR
 TIM1_MemoryMapType, 174
 DontAssertRequest
 Systick Exception (Interrupt) Status, 117
 DR
 ADC_MemoryMapType, 130
 SPI_MemoryMapType, 166
 DR_REG
 USART_MemoryMapType, 179
 EGR
 TIM1_MemoryMapType, 171
 EMR
 EXTI_Type, 136
 ENABLE
 EXTI Lines Status, 41
 MRCC_private.h, 420
 UART_interface.h, 544
 Equal_0
 MRCC_private.h, 431
 Equal_1
 MRCC_private.h, 431
 Equal_10
 MRCC_private.h, 432
 Equal_11
 MRCC_private.h, 432
 Equal_12
 MRCC_private.h, 433
 Equal_13
 MRCC_private.h, 433
 Equal_14
 MRCC_private.h, 433
 Equal_15
 MRCC_private.h, 433
 Equal_16
 MRCC_private.h, 433
 Equal_17
 MRCC_private.h, 433
 Equal_18
 MRCC_private.h, 434
 Equal_19
 MRCC_private.h, 434
 Equal_2
 MRCC_private.h, 431
 Equal_20
 MRCC_private.h, 434
 Equal_21
 MRCC_private.h, 434
 Equal_22
 MRCC_private.h, 434
 Equal_23
 MRCC_private.h, 434
 Equal_24
 MRCC_private.h, 435
 Equal_25
 MRCC_private.h, 435
 Equal_26
 MRCC_private.h, 435
 Equal_27
 MRCC_private.h, 435
 Equal_28
 MRCC_private.h, 435
 Equal_29
 MRCC_private.h, 435
 Equal_3
 MRCC_private.h, 431
 Equal_30
 MRCC_private.h, 436
 Equal_31
 MRCC_private.h, 436
 Equal_32
 MRCC_private.h, 436
 Equal_33
 MRCC_private.h, 436
 Equal_34
 MRCC_private.h, 436
 Equal_35
 MRCC_private.h, 436
 Equal_36
 MRCC_private.h, 437
 Equal_37
 MRCC_private.h, 437
 Equal_38
 MRCC_private.h, 437
 Equal_39
 MRCC_private.h, 437
 Equal_4
 MRCC_private.h, 431
 Equal_40
 MRCC_private.h, 437
 Equal_41
 MRCC_private.h, 437
 Equal_42
 MRCC_private.h, 438
 Equal_43
 MRCC_private.h, 438
 Equal_44
 MRCC_private.h, 438

Equal_45
 MRCC_private.h, 438
Equal_46
 MRCC_private.h, 438
Equal_47
 MRCC_private.h, 438
Equal_48
 MRCC_private.h, 439
Equal_49
 MRCC_private.h, 439
Equal_5
 MRCC_private.h, 431
Equal_50
 MRCC_private.h, 439
Equal_51
 MRCC_private.h, 439
Equal_52
 MRCC_private.h, 439
Equal_53
 MRCC_private.h, 439
Equal_54
 MRCC_private.h, 440
Equal_55
 MRCC_private.h, 440
Equal_56
 MRCC_private.h, 440
Equal_57
 MRCC_private.h, 440
Equal_58
 MRCC_private.h, 440
Equal_59
 MRCC_private.h, 440
Equal_6
 MRCC_private.h, 432
Equal_60
 MRCC_private.h, 441
Equal_61
 MRCC_private.h, 441
Equal_62
 MRCC_private.h, 441
Equal_63
 MRCC_private.h, 441
Equal_7
 MRCC_private.h, 432
Equal_8
 MRCC_private.h, 432
Equal_9
 MRCC_private.h, 432
EVEN_PARITY
 UART Parity, 122
Exception_Request
 Systick Configuration, 112
EXTI Line Settings, 42
 EXTI_MAX_EXTI_NUM, 42
EXTI Lines Status, 41
 DISABLE, 42
 ENABLE, 41
EXTI Memory Addresses, 40
 EXTI_BASE_ADDRESS, 40
 EXTI Registers, 41
 MEXTI, 41
 EXTI0
 NVIC Vector Table, 79
 EXTI0_IRQHandler
 EXTI_program.c, 304
 EXTI1
 NVIC Vector Table, 80
 EXTI15_10
 NVIC Vector Table, 87
 EXTI15_10_IRQHandler
 EXTI_program.c, 306
 EXTI16
 NVIC Vector Table, 78
 EXTI17
 NVIC Vector Table, 87
 EXTI18
 NVIC Vector Table, 87
 EXTI1_IRQHandler
 EXTI_program.c, 304
 EXTI2
 NVIC Vector Table, 80
 EXTI21
 NVIC Vector Table, 78
 EXTI22
 NVIC Vector Table, 79
 EXTI2_IRQHandler
 EXTI_program.c, 305
 EXTI3
 NVIC Vector Table, 80
 EXTI3_IRQHandler
 EXTI_program.c, 305
 EXTI4
 NVIC Vector Table, 80
 EXTI4_IRQHandler
 EXTI_program.c, 305
 EXTI9
 NVIC Vector Table, 83
 EXTI9_5_IRQHandler
 EXTI_program.c, 306
 EXTI_BASE_ADDRESS
 EXTI Memory Addresses, 40
 EXTI_ConfigType, 134
 LineNum, 134
 PortNum, 134
 TriggerStatus, 135
 EXTI_FallingEdge
 Interrupt trigger status, 39
 EXTI_interface.h
 MEXTI_vDisableLine, 292
 MEXTI_vEnableLine, 291
 MEXTI_vInit, 288
 MEXTI_vInit_WithStruct, 291
 MEXTI_vSetCallback, 293
 MEXTI_vSetTrigger, 293
 MEXTI_vSWITrigger, 292
 MSYSCFG_vSetEXTIPort, 294

EXTI_IRQs
 EXTI_private.h, 296

EXTI_LINE0
 Interrupt line IDs, 35

EXTI_LINE1
 Interrupt line IDs, 35

EXTI_LINE10
 Interrupt line IDs, 38

EXTI_LINE11
 Interrupt line IDs, 38

EXTI_LINE12
 Interrupt line IDs, 38

EXTI_LINE13
 Interrupt line IDs, 38

EXTI_LINE14
 Interrupt line IDs, 39

EXTI_LINE15
 Interrupt line IDs, 39

EXTI_LINE2
 Interrupt line IDs, 36

EXTI_LINE3
 Interrupt line IDs, 36

EXTI_LINE4
 Interrupt line IDs, 36

EXTI_LINE5
 Interrupt line IDs, 36

EXTI_LINE6
 Interrupt line IDs, 37

EXTI_LINE7
 Interrupt line IDs, 37

EXTI_LINE8
 Interrupt line IDs, 37

EXTI_LINE9
 Interrupt line IDs, 37

EXTI_MAX_EXTI_NUM
 EXTI Line Settings, 42

EXTI_OnChange
 Interrupt trigger status, 40

EXTI_private.h
 EXTI_IRQs, 296

EXTI_program.c
 EXTI0_IRQHandler, 304
 EXTI15_10_IRQHandler, 306
 EXTI1_IRQHandler, 304
 EXTI2_IRQHandler, 305
 EXTI3_IRQHandler, 305
 EXTI4_IRQHandler, 305
 EXTI9_5_IRQHandler, 306
 MEXTI_vDisableLine, 302
 MEXTI_vEnableLine, 302
 MEXTI_vInit, 298
 MEXTI_vInit_WithStruct, 301
 MEXTI_vSetCallback, 304
 MEXTI_vSetTrigger, 303
 MEXTI_vSWITrigger, 302
 MSYSCFG_vSetEXTIPort, 298

EXTI_RisingEdge
 Interrupt trigger status, 40

EXTI_Type, 135

EMR, 136

FTSR, 136

IMR, 136

PR, 137

RTSR, 136

SWIER, 136

EXTICR
 MSYSCFG_MemMap_t, 145

f32_t
 Standard types, 25

f64_t
 Standard types, 25

FALSE
 Standard values, 26

FLAG_CLEARED
 Standard values, 27

FLAG_SET
 Standard values, 26

FLASH
 NVIC Vector Table, 79

FORWARD
 DC Motor Directions, 11

FPU
 NVIC Vector Table, 91

FTSR
 EXTI_Type, 136

FUNC
 Compiler standard macros, 19

GET_BIT
 Bit Manipulation Math Macros, 17

GET_BITS
 Bit Group Manipulation Math Macros, 18

GPIO Addresses, 59
 GPIOA_BASE_ADDRESS, 59
 GPIOB_BASE_ADDRESS, 59
 GPIOC_BASE_ADDRESS, 60

GPIO Alternate Functions, 54
 GPIOx_AF0, 55
 GPIOx_AF1, 55
 GPIOx_AF10, 57
 GPIOx_AF11, 58
 GPIOx_AF12, 58
 GPIOx_AF13, 58
 GPIOx_AF14, 58
 GPIOx_AF15, 59
 GPIOx_AF2, 55
 GPIOx_AF3, 56
 GPIOx_AF4, 56
 GPIOx_AF5, 56
 GPIOx_AF6, 56
 GPIOx_AF7, 57
 GPIOx_AF8, 57
 GPIOx_AF9, 57

GPIO Modes, 44
 GPIOx_MODE_AF, 44
 GPIOx_MODE_ANALOG, 44

GPIOx_MODE_INPUT, 44
GPIOx_MODE_OUTPUT, 44
GPIO Output PINs, 50
 GPIOx_PIN0, 50
 GPIOx_PIN1, 50
 GPIOx_PIN10, 53
 GPIOx_PIN11, 53
 GPIOx_PIN12, 53
 GPIOx_PIN13, 53
 GPIOx_PIN14, 54
 GPIOx_PIN15, 54
 GPIOx_PIN2, 51
 GPIOx_PIN3, 51
 GPIOx_PIN4, 51
 GPIOx_PIN5, 51
 GPIOx_PIN6, 52
 GPIOx_PIN7, 52
 GPIOx_PIN8, 52
 GPIOx_PIN9, 52
GPIO Output Types, 46
 GPIOx_OPENDRAIN, 46
 GPIOx_PUSH_PULL, 46
GPIO Output Values, 49
 GPIOx_HIGH, 49
 GPIOx_LOW, 49
GPIO PIN Speed, 47
 GPIOx_HighSpeed, 47
 GPIOx_LowSpeed, 47
 GPIOx_MediumSpeed, 47
 GPIOx_VeryHighSpeed, 47
GPIO Ports, 45
 GPIO_PORTA, 45
 GPIO_PORTB, 45
 GPIO_PORTC, 45
GPIO Pull Types, 48
 GPIOx_NoPull, 48
 GPIOx_PullDown, 48
 GPIOx_PullUp, 48
GPIO Registers, 60
 GPIOA, 60
 GPIOB, 60
 GPIOC, 61
GPIO_config.h
 GPIOA_PIN_POS, 314
 GPIOB_PIN_POS, 314
 LCKK_BIT_POS, 314
 PORTA_BIT_MANIPULATION, 314
 PORTB_BIT_MANIPULATION, 314
GPIO_interface.h
 GPIO_vSetNibbleLowValue, 328
 GPIOA_LOW_NIBBLE_HIGH, 318
 GPIOA_LOW_NIBBLE_LOW, 318
 MGPIOx_u8GetPinValue, 322
 MGPIOx_vInit, 327
 MGPIOx_vLockedPins, 319
 MGPIOx_vSetAlternateFunctionON, 325
 MGPIOx_vSetPinInputPullType, 322
 MGPIOx_vSetPinMode, 319
 MGPIOx_vSetPinOutputSpeed, 321
 MGPIOx_vSetPinOutputType, 320
 MGPIOx_vSetPinValue, 323
 MGPIOx_vSetPortConfigLock, 327
 MGPIOx_vSetResetAtomic, 324
 MGPIOx_vTogglePinValue, 327
 GPIO_PORTA
 GPIO Ports, 45
 GPIO_PORTB
 GPIO Ports, 45
 GPIO_PORTC
 GPIO Ports, 45
 GPIO_program.c
 GPIO_vSetNibbleLowValue, 343
 MGPIOx_u8GetPinValue, 337
 MGPIOx_vInit, 341
 MGPIOx_vLockedPins, 333
 MGPIOx_vSetAlternateFunctionON, 340
 MGPIOx_vSetPinInputPullType, 336
 MGPIOx_vSetPinMode, 334
 MGPIOx_vSetPinOutputSpeed, 336
 MGPIOx_vSetPinOutputType, 335
 MGPIOx_vSetPinValue, 338
 MGPIOx_vSetResetAtomic, 339
 MGPIOx_vTogglePinValue, 342
 GPIO_vSetNibbleLowValue
 GPIO_interface.h, 328
 GPIO_program.c, 343
 GPIOA
 GPIO Registers, 60
 GPIOA_BASE_ADDRESS
 GPIO Addresses, 59
 GPIOA_LOW_NIBBLE_HIGH
 GPIO_interface.h, 318
 GPIOA_LOW_NIBBLE_LOW
 GPIO_interface.h, 318
 GPIOA_PIN_POS
 GPIO_config.h, 314
 GPIOACLK
 MRCC_config.h, 377
 GPIOB
 GPIO Registers, 60
 GPIOB_BASE_ADDRESS
 GPIO Addresses, 59
 GPIOB_PIN_POS
 GPIO_config.h, 314
 GPIOBCLK
 MRCC_config.h, 377
 GPIOC
 GPIO Registers, 61
 GPIOC_BASE_ADDRESS
 GPIO Addresses, 60
 GPIOCCLK
 MRCC_config.h, 377
 GPIODCLK
 MRCC_config.h, 377
 GPIOECLK
 MRCC_config.h, 377

GPIOHCLK
 MRCC_config.h, 376
GPIOx_AF0
 GPIO Alternate Functions, 55
GPIOx_AF1
 GPIO Alternate Functions, 55
GPIOx_AF10
 GPIO Alternate Functions, 57
GPIOx_AF11
 GPIO Alternate Functions, 58
GPIOx_AF12
 GPIO Alternate Functions, 58
GPIOx_AF13
 GPIO Alternate Functions, 58
GPIOx_AF14
 GPIO Alternate Functions, 58
GPIOx_AF15
 GPIO Alternate Functions, 59
GPIOx_AF2
 GPIO Alternate Functions, 55
GPIOx_AF3
 GPIO Alternate Functions, 56
GPIOx_AF4
 GPIO Alternate Functions, 56
GPIOx_AF5
 GPIO Alternate Functions, 56
GPIOx_AF6
 GPIO Alternate Functions, 56
GPIOx_AF7
 GPIO Alternate Functions, 57
GPIOx_AF8
 GPIO Alternate Functions, 57
GPIOx_AF9
 GPIO Alternate Functions, 57
GPIOx_HIGH
 GPIO Output Values, 49
GPIOx_HighSpeed
 GPIO PIN Speed, 47
GPIOx_LOW
 GPIO Output Values, 49
GPIOx_LowSpeed
 GPIO PIN Speed, 47
GPIOx_MediumSpeed
 GPIO PIN Speed, 47
GPIOx_MemoryMapType, 137
 AFRHx, 139
 AFRLx, 139
 BSRRx, 139
 IDRx, 138
 LCKRx, 139
 MODERx, 138
 ODRx, 139
 OSPEEDRx, 138
 OTYPERx, 138
 PUPDRx, 138
GPIOx_MODE_AF
 GPIO Modes, 44
GPIOx_MODE_ANALOG
 GPIO Modes, 44
GPIOx_MODE_INPUT
 GPIO Modes, 44
GPIOx_MODE_OUTPUT
 GPIO Modes, 44
GPIOx_NoPull
 GPIO Pull Types, 48
GPIOx_OPENDRAIN
 GPIO Output Types, 46
GPIOx_PIN0
 GPIO Output PINs, 50
GPIOx_PIN1
 GPIO Output PINs, 50
GPIOx_PIN10
 GPIO Output PINs, 53
GPIOx_PIN11
 GPIO Output PINs, 53
GPIOx_PIN12
 GPIO Output PINs, 53
GPIOx_PIN13
 GPIO Output PINs, 53
GPIOx_PIN14
 GPIO Output PINs, 54
GPIOx_PIN15
 GPIO Output PINs, 54
GPIOx_PIN2
 GPIO Output PINs, 51
GPIOx_PIN3
 GPIO Output PINs, 51
GPIOx_PIN4
 GPIO Output PINs, 51
GPIOx_PIN5
 GPIO Output PINs, 51
GPIOx_PIN6
 GPIO Output PINs, 52
GPIOx_PIN7
 GPIO Output PINs, 52
GPIOx_PIN8
 GPIO Output PINs, 52
GPIOx_PIN9
 GPIO Output PINs, 52
GPIOx_PullDown
 GPIO Pull Types, 48
GPIOx_PullUp
 GPIO Pull Types, 48
GPIOx_PUSHPULL
 GPIO Output Types, 46
GPIOx_VeryHighSpeed
 GPIO PIN Speed, 47
Group priorities, 65
 GROUP_PRIORITY_0, 66
 GROUP_PRIORITY_1, 66
 GROUP_PRIORITY_10, 68
 GROUP_PRIORITY_11, 69
 GROUP_PRIORITY_12, 69
 GROUP_PRIORITY_13, 69
 GROUP_PRIORITY_14, 69
 GROUP_PRIORITY_15, 70

GROUP_PRIORITY_2, 66
GROUP_PRIORITY_3, 67
GROUP_PRIORITY_4, 67
GROUP_PRIORITY_5, 67
GROUP_PRIORITY_6, 67
GROUP_PRIORITY_7, 68
GROUP_PRIORITY_8, 68
GROUP_PRIORITY_9, 68
NO_GROUP_PRIORITY, 66
GROUP_0BITS
 Interrupt priority grouping, 64
GROUP_1BITS
 Interrupt priority grouping, 64
GROUP_2BITS
 Interrupt priority grouping, 64
GROUP_3BITS
 Interrupt priority grouping, 64
GROUP_4BITS
 Interrupt priority grouping, 63
GROUP_PRIORITY_0
 Group priorities, 66
GROUP_PRIORITY_1
 Group priorities, 66
GROUP_PRIORITY_10
 Group priorities, 68
GROUP_PRIORITY_11
 Group priorities, 69
GROUP_PRIORITY_12
 Group priorities, 69
GROUP_PRIORITY_13
 Group priorities, 69
GROUP_PRIORITY_14
 Group priorities, 69
GROUP_PRIORITY_15
 Group priorities, 70
GROUP_PRIORITY_2
 Group priorities, 66
GROUP_PRIORITY_3
 Group priorities, 67
GROUP_PRIORITY_4
 Group priorities, 67
GROUP_PRIORITY_5
 Group priorities, 67
GROUP_PRIORITY_6
 Group priorities, 67
GROUP_PRIORITY_7
 Group priorities, 68
GROUP_PRIORITY_8
 Group priorities, 68
GROUP_PRIORITY_9
 Group priorities, 68
GTPR_REG
 USART_MemoryMapType, 180
HardwareFlowControl
 USART_InitType, 178
HBUZZER_vInit
 BUZZER_interface.h, 203
 BUZZER_program.c, 206
 HBUZZER_vSoundOff
 BUZZER_interface.h, 204
 BUZZER_program.c, 207
 HBUZZER_vSoundOn
 BUZZER_interface.h, 203
 BUZZER_program.c, 207
 HBUZZER_vToggleSound
 BUZZER_interface.h, 204
 BUZZER_program.c, 207
HDCM_vGetSpeedValue
 DCM_interface.h, 217
 DCM_program.c, 224
HDCM_vInitMotor
 DCM_interface.h, 214
 DCM_program.c, 221
HDCM_vMotorSpeedCntrl
 DCM_interface.h, 217
 DCM_program.c, 223
HDCM_vMoveBackward
 DCM_interface.h, 216
 DCM_program.c, 222
HDCM_vMoveForward
 DCM_interface.h, 215
 DCM_program.c, 222
HDCM_vStopMotor
 DCM_interface.h, 216
 DCM_program.c, 223
HFSR
 SCB_MemoryMapType, 164
HPRE
 MRCC_config.h, 375
HSE
 MRCC_private.h, 421
HSE_EN
 MRCC_config.h, 373
HSEby10
 MRCC_private.h, 426
HSEby11
 MRCC_private.h, 426
HSEby12
 MRCC_private.h, 427
HSEby13
 MRCC_private.h, 427
HSEby14
 MRCC_private.h, 427
HSEby15
 MRCC_private.h, 427
HSEby16
 MRCC_private.h, 427
HSEby17
 MRCC_private.h, 427
HSEby18
 MRCC_private.h, 428
HSEby19
 MRCC_private.h, 428
HSEby2
 MRCC_private.h, 425
HSEby20

MRCC_private.h, 428
 HSEby21
 MRCC_private.h, 428
 HSEby22
 MRCC_private.h, 428
 HSEby23
 MRCC_private.h, 428
 HSEby24
 MRCC_private.h, 429
 HSEby25
 MRCC_private.h, 429
 HSEby26
 MRCC_private.h, 429
 HSEby27
 MRCC_private.h, 429
 HSEby28
 MRCC_private.h, 429
 HSEby29
 MRCC_private.h, 429
 HSEby3
 MRCC_private.h, 425
 HSEby30
 MRCC_private.h, 430
 HSEby31
 MRCC_private.h, 430
 HSEby4
 MRCC_private.h, 425
 HSEby5
 MRCC_private.h, 425
 HSEby6
 MRCC_private.h, 426
 HSEby7
 MRCC_private.h, 426
 HSEby8
 MRCC_private.h, 426
 HSEby9
 MRCC_private.h, 426
 HSEBYP
 MRCC_config.h, 373
 HSI
 MRCC_private.h, 423
 HSI_EN
 MRCC_config.h, 373
 HTR
 ADC_MemoryMapType, 128
 HULTSNC_ConfigType, 140
 u8Pin, 140
 u8Port, 140
 HULTSNC_f32GetDistance
 UltraSonic_interface.h, 248
 HULTSNC_vInit
 UltraSonic_interface.h, 247
 HULTSNC_vTrigger
 UltraSonic_interface.h, 248
 I2C1_ER
 NVIC_Vector_Table, 85
 I2C1_EV
 NVIC_Vector_Table, 85
 I2C1EN
 MRCC_config.h, 378
 I2C2_ER
 NVIC_Vector_Table, 85
 I2C2_EV
 NVIC_Vector_Table, 85
 I2C2EN
 MRCC_config.h, 378
 I2C3_ER
 NVIC_Vector_Table, 91
 I2C3_EV
 NVIC_Vector_Table, 91
 I2C3EN
 MRCC_config.h, 378
 I2S_CKIN
 MRCC_private.h, 423
 I2SCFGR
 SPI_MemoryMapType, 167
 I2SPR
 SPI_MemoryMapType, 167
 I2SSRC
 MRCC_config.h, 375
 IABRx
 NVIC_MemoryMapType, 149
 ICERx
 NVIC_MemoryMapType, 147
 ICPRx
 NVIC_MemoryMapType, 148
 ICSR
 SCB_MemoryMapType, 161
 ID options, 96, 98
 AHB1ENR_CRCEN, 99
 AHB1ENR_DMA1EN, 99
 AHB1ENR_DMA2EN, 99
 AHB1ENR_GPIOAEN, 101
 AHB1ENR_GPIOBEN, 101
 AHB1ENR_GPIOCEN, 100
 AHB1ENR_GPIODEN, 100
 AHB1ENR_GPIOEEN, 100
 AHB1ENR_GPIOHEN, 100
 AHB1LPENR_FLITFLPEN, 107
 AHB2ENR_OTGFSEN, 101
 APB1ENR_I2C1EN, 102
 APB1ENR_I2C2EN, 102
 APB1ENR_I2C3EN, 102
 APB1ENR_PWREN, 101
 APB1ENR_SPI2EN, 103
 APB1ENR_SPI3EN, 103
 APB1ENR_TIM2EN, 104
 APB1ENR_TIM3EN, 104
 APB1ENR_TIM4EN, 104
 APB1ENR_TIM5EN, 103
 APB1ENR_USART2EN, 102
 APB1ENR_WWDGEN, 103
 APB2ENR_ADC1EN, 106
 APB2ENR_SDIOEN, 106
 APB2ENR_SPI1EN, 106
 APB2ENR_SPI4EN, 105

APB2ENR_SYSCFGEN, 105
APB2ENR_TIM10EN, 105
APB2ENR_TIM11EN, 104
APB2ENR_TIM1EN, 107
APB2ENR_TIM9EN, 105
APB2ENR_USART1EN, 107
APB2ENR_USART6EN, 106
RCC_AHB1, 96
RCC_AHB1LPENR, 97
RCC_AHB2, 96
RCC_APB1, 97
RCC_APB2, 97
IDRx
 GPIOx_MemoryMapType, 138
IMR
 EXTI_Type, 136
INITIAL_ZERO
 Standard values, 26
InputType
 MGPIOx_ConfigType, 144
Interrupt line IDs, 35
 EXTI_LINE0, 35
 EXTI_LINE1, 35
 EXTI_LINE10, 38
 EXTI_LINE11, 38
 EXTI_LINE12, 38
 EXTI_LINE13, 38
 EXTI_LINE14, 39
 EXTI_LINE15, 39
 EXTI_LINE2, 36
 EXTI_LINE3, 36
 EXTI_LINE4, 36
 EXTI_LINE5, 36
 EXTI_LINE6, 37
 EXTI_LINE7, 37
 EXTI_LINE8, 37
 EXTI_LINE9, 37
Interrupt priority grouping, 61
 _16GROUP_NoSub_Priorities, 62
 _2GROUP_8Sub_Priorities, 63
 _4GROUP_4Sub_Priorities, 62
 _8GROUP_2Sub_Priorities, 62
 GROUP_0BITS, 64
 GROUP_1BITS, 64
 GROUP_2BITS, 64
 GROUP_3BITS, 64
 GROUP_4BITS, 63
 NoGROUP_16Sub_Priorities, 63
Interrupt trigger status, 39
 EXTI_FallingEdge, 39
 EXTI_OnChange, 40
 EXTI_RisingEdge, 40
IPRx
 NVIC_MemoryMapType, 150
ISERx
 NVIC_MemoryMapType, 146
ISPRx
 NVIC_MemoryMapType, 148
JDR1
 ADC_MemoryMapType, 129
JDR2
 ADC_MemoryMapType, 129
JDR3
 ADC_MemoryMapType, 129
JDR4
 ADC_MemoryMapType, 130
JOFR1
 ADC_MemoryMapType, 127
JOFR2
 ADC_MemoryMapType, 127
JOFR3
 ADC_MemoryMapType, 127
JOFR4
 ADC_MemoryMapType, 128
JSQR
 ADC_MemoryMapType, 129
LastBitClockPulse
 USART_ClockInitTypeDef, 175
LCKK_BIT_POS
 GPIO_config.h, 314
LCKRx
 GPIOx_MemoryMapType, 139
LED_LEDConfiguration, 141
 u8Pin, 141
 u8Port, 141
LineNum
 EXTI_ConfigType, 134
LOAD
 SysTick_Type, 168
LSE
 MRCC_private.h, 423
LTR
 ADC_MemoryMapType, 128
MADC_u16ConvertToDigital
 ADC_interface.h, 269
MADC_vDisable
 ADC_interface.h, 270
MADC_vEnable
 ADC_interface.h, 270
MADC_vInit
 ADC_interface.h, 265
MADC_vRegINT_Disable
 ADC_interface.h, 270
MADC_vRegINTEnable
 ADC_interface.h, 271
MADC_vSelectChannel
 ADC_interface.h, 271
MADC_vSetCallBack
 ADC_interface.h, 271
MAX_SPEED
 DCM_config.h, 212
MAX TICKS
 Systick Configuration, 112
MCO1
 MRCC_config.h, 375

MCO1PRE
 MRCC_config.h, 374

MCO2
 MRCC_config.h, 374

MCO2PRE
 MRCC_config.h, 374

MEMORY_MANAGE
 NVIC Settable Priorities, 95

MEMRMP
 MSYSCFG_MemMap_t, 145

MEXTI
 EXTI Registers, 41

MEXTI_vDisableLine
 EXTI_interface.h, 292
 EXTI_program.c, 302

MEXTI_vEnableLine
 EXTI_interface.h, 291
 EXTI_program.c, 302

MEXTI_vInit
 EXTI_interface.h, 288
 EXTI_program.c, 298

MEXTI_vInit_WithStruct
 EXTI_interface.h, 291
 EXTI_program.c, 301

MEXTI_vSetCallback
 EXTI_interface.h, 293
 EXTI_program.c, 304

MEXTI_vSetTrigger
 EXTI_interface.h, 293
 EXTI_program.c, 303

MEXTI_vSWITrigger
 EXTI_interface.h, 292
 EXTI_program.c, 302

MGPIOx_ConfigType, 142
 AF_Type, 144
 InputType, 144
 Mode, 143
 OutputSpeed, 143
 OutputType, 143
 Pin, 143
 Port, 142

MGPIOx_u8GetPinValue
 GPIO_interface.h, 322
 GPIO_program.c, 337

MGPIOx_vInit
 GPIO_interface.h, 327
 GPIO_program.c, 341

MGPIOx_vLockedPins
 GPIO_interface.h, 319
 GPIO_program.c, 333

MGPIOx_vSetAlternateFunctionON
 GPIO_interface.h, 325
 GPIO_program.c, 340

MGPIOx_vSetPinInputPullType
 GPIO_interface.h, 322
 GPIO_program.c, 336

MGPIOx_vSetPinMode
 GPIO_interface.h, 319

 GPIO_program.c, 334

 GPIO_interface.h, 321

 GPIO_program.c, 336

 GPIO_interface.h, 320

 GPIO_program.c, 335

 GPIO_interface.h, 323

 GPIO_program.c, 338

 GPIO_interface.h, 327

 GPIO_program.c, 339

 GPIO_interface.h, 327

 GPIO_program.c, 342

 Delay Units, 113

 Delay Units, 113

MMFAR
 SCB_MemoryMapType, 164

MNVIC
 NVIC Registers, 93

MNVIC_STIR
 NVIC_private.h, 361

MNVIC_u8GetActive
 NVIC_interface.h, 356
 NVIC_program.c, 366

MNVIC_vClearPendingFlag
 NVIC_interface.h, 355
 NVIC_program.c, 365

MNVIC_vDisablePeriphral
 NVIC_interface.h, 354
 NVIC_program.c, 364

MNVIC_vEnablePeriphral
 NVIC_interface.h, 354
 NVIC_program.c, 364

MNVIC_vSetPendingFlag
 NVIC_interface.h, 355
 NVIC_program.c, 365

MNVIC_vSetPriority
 NVIC_interface.h, 357
 NVIC_program.c, 367

MNVIC_vSetPriorityConfig
 NVIC_interface.h, 356
 NVIC_program.c, 366

Mode
 MGPIOx_ConfigType, 143

MODE_8BIT
 UART Bit Sizes, 122

MODE_9BIT
 UART Bit Sizes, 123

MODERx
 GPIOx_MemoryMapType, 138

MRCC_config.h

ADC1EN, 381
CRCCCLK, 376
CSS, 373
DMA1CLK, 376
DMA2CLK, 376
GPIOACLK, 377
GPIOBCLK, 377
GPIOCCLK, 377
GPIODCLK, 377
GPIOECLK, 377
GPIOHCLK, 376
HPRE, 375
HSE_EN, 373
HSEBYP, 373
HSI_EN, 373
I2C1EN, 378
I2C2EN, 378
I2C3EN, 378
I2SSRC, 375
MCO1, 375
MCO1PRE, 374
MCO2, 374
MCO2PRE, 374
OTGFS, 377
PLL, 372
PLLI2S, 372
PLLM, 374
PLLN, 374
PLLP, 374
PLLQ, 373
PLLSRC, 373
PPRE1, 375
PPRE2, 375
PWREN, 378
RTCPRE, 375
SDIOEN, 381
SPI1EN, 380
SPI2EN, 379
SPI3EN, 378
SPI4EN, 380
SW, 376
SWS, 376
SYSCFGEN, 380
TIM10EN, 380
TIM11EN, 380
TIM1EN, 381
TIM2EN, 379
TIM3EN, 379
TIM4EN, 379
TIM5EN, 379
TIM9EN, 380
USART1EN, 381
USART2EN, 378
USART6EN, 381
WWDGEN, 379
MRCC_interface.h
 MRCC_vDisablePeriphralCLK, 397
 MRCC_vEnablePeriphralCLK, 396
 MRCC_vInit, 392
 MRCC_private.h
 AHBby16, 430
 AHBby2, 430
 AHBby4, 430
 AHBby8, 430
 BYBASED, 421
 DISABLE, 421
 DivisionBy2, 422
 DivisionBy3, 422
 DivisionBy4, 422
 DivisionBy5, 422
 ENABLE, 420
 Equal_0, 431
 Equal_1, 431
 Equal_10, 432
 Equal_11, 432
 Equal_12, 433
 Equal_13, 433
 Equal_14, 433
 Equal_15, 433
 Equal_16, 433
 Equal_17, 433
 Equal_18, 434
 Equal_19, 434
 Equal_2, 431
 Equal_20, 434
 Equal_21, 434
 Equal_22, 434
 Equal_23, 434
 Equal_24, 435
 Equal_25, 435
 Equal_26, 435
 Equal_27, 435
 Equal_28, 435
 Equal_29, 435
 Equal_3, 431
 Equal_30, 436
 Equal_31, 436
 Equal_32, 436
 Equal_33, 436
 Equal_34, 436
 Equal_35, 436
 Equal_36, 437
 Equal_37, 437
 Equal_38, 437
 Equal_39, 437
 Equal_4, 431
 Equal_40, 437
 Equal_41, 437
 Equal_42, 438
 Equal_43, 438
 Equal_44, 438
 Equal_45, 438
 Equal_46, 438
 Equal_47, 438
 Equal_48, 439
 Equal_49, 439

Equal_5, 431
 Equal_50, 439
 Equal_51, 439
 Equal_52, 439
 Equal_53, 439
 Equal_54, 440
 Equal_55, 440
 Equal_56, 440
 Equal_57, 440
 Equal_58, 440
 Equal_59, 440
 Equal_6, 432
 Equal_60, 441
 Equal_61, 441
 Equal_62, 441
 Equal_63, 441
 Equal_7, 432
 Equal_8, 432
 Equal_9, 432
 HSE, 421
 HSEby10, 426
 HSEby11, 426
 HSEby12, 427
 HSEby13, 427
 HSEby14, 427
 HSEby15, 427
 HSEby16, 427
 HSEby17, 427
 HSEby18, 428
 HSEby19, 428
 HSEby2, 425
 HSEby20, 428
 HSEby21, 428
 HSEby22, 428
 HSEby23, 428
 HSEby24, 429
 HSEby25, 429
 HSEby26, 429
 HSEby27, 429
 HSEby28, 429
 HSEby29, 429
 HSEby3, 425
 HSEby30, 430
 HSEby31, 430
 HSEby4, 425
 HSEby5, 425
 HSEby6, 426
 HSEby7, 426
 HSEby8, 426
 HSEby9, 426
 HSI, 423
 I2S_CKIN, 423
 LSE, 423
 NoCLK0, 425
 NoCLK1, 425
 NoDivision, 422
 NOTBYBASED, 421
 PLLCLK, 422, 423
 PLLI2SCLK, 421
 RCC, 404
 RCC_AHB1ENR_CRCEN, 415
 RCC_AHB1ENR_DMA1EN, 415
 RCC_AHB1ENR_DMA2EN, 415
 RCC_AHB1ENR_GPIOAEN, 416
 RCC_AHB1ENR_GPIOBEN, 416
 RCC_AHB1ENR_GPIOCEN, 416
 RCC_AHB1ENR_GPIODEN, 416
 RCC_AHB1ENR_GPIOEEN, 415
 RCC_AHB1ENR_GPIOHEN, 415
 RCC_AHB1LPENR_FLITFLPEN, 420
 RCC_AHB2ENR_OTGFSEN, 416
 RCC_APB1ENR_I2C1EN, 417
 RCC_APB1ENR_I2C2EN, 417
 RCC_APB1ENR_I2C3EN, 417
 RCC_APB1ENR_PWREN, 416
 RCC_APB1ENR_SPI2EN, 417
 RCC_APB1ENR_SPI3EN, 417
 RCC_APB1ENR_TIM2EN, 418
 RCC_APB1ENR_TIM3EN, 418
 RCC_APB1ENR_TIM4EN, 418
 RCC_APB1ENR_TIM5EN, 418
 RCC_APB1ENR_USART2EN, 417
 RCC_APB1ENR_WWDGEN, 418
 RCC_APB2ENR_ADC1EN, 420
 RCC_APB2ENR_SDIOEN, 419
 RCC_APB2ENR_SPI1EN, 419
 RCC_APB2ENR_SPI4EN, 419
 RCC_APB2ENR_SYSCFGEN, 419
 RCC_APB2ENR_TIM10EN, 419
 RCC_APB2ENR_TIM11EN, 418
 RCC_APB2ENR_TIM1EN, 420
 RCC_APB2ENR_TIM9EN, 419
 RCC_APB2ENR_USART1EN, 420
 RCC_APB2ENR_USART6EN, 420
 RCC_BASE_ADDRESS, 404
 RCC_CFGR_HPRE_b0, 413
 RCC_CFGR_HPRE_b1, 414
 RCC_CFGR_HPRE_b2, 414
 RCC_CFGR_HPRE_b3, 414
 RCC_CFGR_I2SSRC, 411
 RCC_CFGR_MOC1_b0, 411
 RCC_CFGR_MOC1_b1, 411
 RCC_CFGR_MOC1PRE_b0, 411
 RCC_CFGR_MOC1PRE_b1, 411
 RCC_CFGR_MOC1PRE_b2, 411
 RCC_CFGR_MOC2_b0, 410
 RCC_CFGR_MOC2_b1, 410
 RCC_CFGR_MOC2PRE_b0, 410
 RCC_CFGR_MOC2PRE_b1, 410
 RCC_CFGR_MOC2PRE_b2, 410
 RCC_CFGR_PPREG1_b0, 413
 RCC_CFGR_PPREG1_b1, 413
 RCC_CFGR_PPREG1_b2, 413
 RCC_CFGR_PPREG2_b0, 412
 RCC_CFGR_PPREG2_b1, 413
 RCC_CFGR_PPREG2_b2, 413

RCC_CFGR_RTCPRE_b0, 412
RCC_CFGR_RTCPRE_b1, 412
RCC_CFGR_RTCPRE_b2, 412
RCC_CFGR_RTCPRE_b3, 412
RCC_CFGR_RTCPRE_b4, 412
RCC_CFGR_SW_b0, 414
RCC_CFGR_SW_b1, 415
RCC_CFGR_SWS_b0, 414
RCC_CFGR_SWS_b1, 414
RCC_CR_CSSON, 405
RCC_CR_HSEBYP, 405
RCC_CR_HSEON, 406
RCC_CR_HSERDY, 405
RCC_CR_HSION, 406
RCC_CR_HSIRDY, 406
RCC_CR_PLLI2SON, 405
RCC_CR_PLLI2SRDY, 404
RCC_CR_PLLON, 405
RCC_CR_PLLRDY, 405
RCC_PLLCFG_R_PLLM_b0, 409
RCC_PLLCFG_R_PLLM_b1, 409
RCC_PLLCFG_R_PLLM_b2, 409
RCC_PLLCFG_R_PLLM_b3, 409
RCC_PLLCFG_R_PLLM_b4, 409
RCC_PLLCFG_R_PLLM_b5, 410
RCC_PLLCFG_R_PLLN_b0, 407
RCC_PLLCFG_R_PLLN_b1, 407
RCC_PLLCFG_R_PLLN_b2, 408
RCC_PLLCFG_R_PLLN_b3, 408
RCC_PLLCFG_R_PLLN_b4, 408
RCC_PLLCFG_R_PLLN_b5, 408
RCC_PLLCFG_R_PLLN_b6, 408
RCC_PLLCFG_R_PLLN_b7, 408
RCC_PLLCFG_R_PLLN_b8, 409
RCC_PLLCFG_R_PLLP_b0, 407
RCC_PLLCFG_R_PLLP_b1, 407
RCC_PLLCFG_R_PLLQ_b0, 406
RCC_PLLCFG_R_PLLQ_b1, 406
RCC_PLLCFG_R_PLLQ_b2, 406
RCC_PLLCFG_R_PLLQ_b3, 407
RCC_PLLCFG_R_PLLSRC, 407
SYSCLK, 421
SYSCLKby128, 424
SYSCLKby16, 424
SYSCLKby2, 423
SYSCLKby256, 424
SYSCLKby4, 423
SYSCLKby512, 424
SYSCLKby64, 424
SYSCLKby8, 424
MRCC_program.c
 MRCC_vDisablePeriphralCLK, 452
 MRCC_vEnablePeriphralCLK, 451
 MRCC_vInit, 447
MRCC_vDisablePeriphralCLK
 MRCC_interface.h, 397
 MRCC_program.c, 452
MRCC_vEnablePeriphralCLK
 MRCC_interface.h, 396
 MRCC_program.c, 451
 MRCC_vInit
 MRCC_interface.h, 392
 MRCC_program.c, 447
 MSCB
 NVIC Registers, 93
 MSYSCFG
 SYSCFG Memory Registers, 43
 MSYSCFG_MemMap_t, 144
 CMPCR, 145
 EXTICR, 145
 MEMRMP, 145
 PMC, 145
 MSYSCFG_vSetEXTIPort
 EXTI_interface.h, 294
 EXTI_program.c, 298
MSysTick_u32GetElapsedTime
 SysTick_interface.h, 482
 SysTick_program.c, 493
MSysTick_u32GetRemainingTime
 SysTick_interface.h, 482
 SysTick_program.c, 494
MSysTick_vDelay
 SysTick_interface.h, 477
 SysTick_program.c, 489
MSysTick_vDelayMicroSec
 SysTick_interface.h, 478
 SysTick_program.c, 489
MSysTick_vDelayMilliSec
 SysTick_interface.h, 478
 SysTick_program.c, 490
MSysTick_vDelaySec
 SysTick_interface.h, 479
 SysTick_program.c, 491
MSysTick_vDisable
 SysTick_interface.h, 483
 SysTick_program.c, 494
MSysTick_vDisableException
 SysTick_interface.h, 483
 SysTick_program.c, 495
MSysTick_vEnable
 SysTick_interface.h, 482
 SysTick_program.c, 494
MSysTick_vEnableException
 SysTick_interface.h, 483
 SysTick_program.c, 495
MSysTick_vInit
 SysTick_interface.h, 476
 SysTick_program.c, 487
MSysTick_vSetBusyWait
 SysTick_interface.h, 476
 SysTick_program.c, 488
MSysTick_vSetPeriodicInterval
 SysTick_interface.h, 480
 SysTick_program.c, 492
MSysTick_vSetSingleInterval
 SysTick_interface.h, 480

SysTick_program.c, 492
MSysTick_vStopInterval
 SysTick_interface.h, 481
 SysTick_program.c, 493
MUSART1_vSetCallBack
 UART_interface.h, 553
MUSART2_vSetCallBack
 UART_interface.h, 553
MUSART6_vSetCallBack
 UART_interface.h, 554
MUSART_CR1_IDLEIE_BIT
 UART_private.h, 562
MUSART_CR1_M_BIT
 UART_private.h, 563
MUSART_CR1_OVER8_BIT
 UART_private.h, 563
MUSART_CR1_PCE_BIT
 UART_private.h, 563
MUSART_CR1_PEIE_BIT
 UART_private.h, 562
MUSART_CR1_PS_BIT
 UART_private.h, 562
MUSART_CR1_RE_BIT
 UART_private.h, 561
MUSART_CR1_RWU_BIT
 UART_private.h, 561
MUSART_CR1_RXNEIE_BIT
 UART_private.h, 562
MUSART_CR1_SBK_BIT
 UART_private.h, 561
MUSART_CR1_TCIE_BIT
 UART_private.h, 562
MUSART_CR1_TE_BIT
 UART_private.h, 561
MUSART_CR1_TXEIE_BIT
 UART_private.h, 562
MUSART_CR1_UE_BIT
 UART_private.h, 563
MUSART_CR1_WAKE_BIT
 UART_private.h, 563
MUSART_CR2_ADD0_BIT
 UART_private.h, 563
MUSART_CR2_ADD1_BIT
 UART_private.h, 564
MUSART_CR2_ADD2_BIT
 UART_private.h, 564
MUSART_CR2_ADD3_BIT
 UART_private.h, 564
MUSART_CR2_CLKEN_BIT
 UART_private.h, 565
MUSART_CR2_CPHA_BIT
 UART_private.h, 565
MUSART_CR2_CPOL_BIT
 UART_private.h, 565
MUSART_CR2_LBCL_BIT
 UART_private.h, 564
MUSART_CR2_LBDIE_BIT
 UART_private.h, 564
MUSART_CR2_LBDL_BIT
 UART_private.h, 564
MUSART_CR2_LINEN_BIT
 UART_private.h, 566
MUSART_CR2_STOP0_BIT
 UART_private.h, 565
MUSART_CR2_STOP1_BIT
 UART_private.h, 565
MUSART_CR2_STOP_BIT
 UART_private.h, 565
MUSART_CR3_CTSE_BIT
 UART_private.h, 566
MUSART_CR3_CTSIE_BIT
 UART_private.h, 566
MUSART_CR3_DMAR_BIT
 UART_private.h, 566
MUSART_CR3_DMAT_BIT
 UART_private.h, 566
MUSART_CR3_EIE_BIT
 UART_private.h, 567
MUSART_CR3_HDSEL_BIT
 UART_private.h, 567
MUSART_CR3_IREN_BIT
 UART_private.h, 567
MUSART_CR3_IRLP_BIT
 UART_private.h, 567
MUSART_CR3_NACK_BIT
 UART_private.h, 567
MUSART_CR3_RTSE_BIT
 UART_private.h, 566
MUSART_CR3_SCEN_BIT
 UART_private.h, 567
MUSART_ptrReceiveStringSynchNonBlocking
 UART_interface.h, 549
MUSART_SR_CTS_BIT
 UART_private.h, 561
MUSART_SR_FE_BIT
 UART_private.h, 559
MUSART_SR_IDLE_BIT
 UART_private.h, 560
MUSART_SR_LBD_BIT
 UART_private.h, 561
MUSART_SR_NE_BIT
 UART_private.h, 560
MUSART_SR_ORE_BIT
 UART_private.h, 560
MUSART_SR_PE_BIT
 UART_private.h, 559
MUSART_SR_RXNE_BIT
 UART_private.h, 560
MUSART_SR_TC_BIT
 UART_private.h, 560
MUSART_SR_TXE_BIT
 UART_private.h, 560
MUSART_u8CompareString
 UART_interface.h, 551
MUSART_u8ReadDataRegister
 UART_interface.h, 551

MUSART_u8ReceiveByteSynchBlocking
 UART_interface.h, 549

MUSART_u8ReceiveByteSynchNonBlocking
 UART_interface.h, 548

MUSART_vClearFlags
 UART_interface.h, 552

MUSART_vDisable
 UART_interface.h, 547

MUSART_vEnable
 UART_interface.h, 546

MUSART_vInit
 UART_interface.h, 544

MUSART_vRecieveString
 UART_interface.h, 550

MUSART_vRxIntSetStatus
 UART_interface.h, 552

MUSART_vTransmitByte
 UART_interface.h, 547

MUSART_vTransmitString
 UART_interface.h, 548

MY_MS_DELAY
 Utilities macros, 21

NO_GROUP_PRIORITY
 Group priorities, 66

NO_SUB_PRIORITY
 Sub-Group priorities, 71

NoCLK0
 MRCC_private.h, 425

NoCLK1
 MRCC_private.h, 425

NoDivision
 MRCC_private.h, 422

NoGROUP_16Sub_Priorities
 Interrupt priority grouping, 63

NOTBYBASED
 MRCC_private.h, 421

NULL
 Standard values, 26

NVIC Addresses, 92
 NVIC_BASE_ADDRESS, 92
 SCB_BASE_ADDRESS, 93

NVIC Registers, 93
 MNVIC, 93
 MSCB, 93

NVIC Settable Priorities, 94
 BUS_FAULT, 95
 MEMORY_MANAGE, 95
 PEND_SV, 94
 SV_CALL, 95
 SYSTICK, 94
 USAGE_FAULT, 95

NVIC Vector Table, 75
 ADC, 82
 DMA1_STREAM0, 81
 DMA1_STREAM1, 81
 DMA1_STREAM2, 81
 DMA1_STREAM3, 81
 DMA1_STREAM4, 82

DMA1_STREAM5, 82

DMA1_STREAM6, 82

DMA1_STREAM7, 87

DMA2_STREAM0, 88

DMA2_STREAM1, 89

DMA2_STREAM2, 89

DMA2_STREAM3, 89

DMA2_STREAM4, 89

DMA2_STREAM5, 90

DMA2_STREAM6, 90

DMA2_STREAM7, 90

EXTI0, 79

EXTI1, 80

EXTI15_10, 87

EXTI16, 78

EXTI17, 87

EXTI18, 87

EXTI2, 80

EXTI21, 78

EXTI22, 79

EXTI3, 80

EXTI4, 80

EXTI9, 83

FLASH, 79

FPU, 91

I2C1_ER, 85

I2C1_EV, 85

I2C2_ER, 85

I2C2_EV, 85

I2C3_ER, 91

I2C3_EV, 91

OTG_FS, 90

RCC, 79

SDIO, 88

SPI1, 86

SPI2, 86

SPI3, 88

SPI4, 92

TIM1_BRK_TIM9, 83

TIM1_CC, 84

TIM1_TRG_COM_TIM11, 83

TIM1_UP_TIM10, 83

TIM2, 84

TIM3, 84

TIM4, 84

TIM5, 88

USART1, 86

USART2, 86

USART6, 91

WWDG, 78

NVIC_BASE_ADDRESS
 NVIC Addresses, 92

NVIC_GetPriority
 NVIC_interface.h, 358
 NVIC_program.c, 368

NVIC_interface.h
 MNVIC_u8GetActive, 356
 MNVIC_vClearPendingFlag, 355

MNVIC_vDisablePeriphral, 354
 MNVIC_vEnablePeriphral, 354
 MNVIC_vSetPendingFlag, 355
 MNVIC_vSetPriority, 357
 MNVIC_vSetPriorityConfig, 356
 NVIC_GetPriority, 358
NVIC_MemoryMapType, 146
 IABRx, 149
 ICERx, 147
 ICPRx, 148
 IPRx, 150
 ISERx, 146
 ISPRx, 148
 RESERVED0x, 147
 RESERVED2x, 148
 RESERVED3x, 149
 RESERVED4x, 149
 RSERVED1x, 147
NVIC_private.h
 MNVIC_STIR, 361
 VECTKEY_PASSWORD, 361
NVIC_program.c
 MNVIC_u8GetActive, 366
 MNVIC_vClearPendingFlag, 365
 MNVIC_vDisablePeriphral, 364
 MNVIC_vEnablePeriphral, 364
 MNVIC_vSetPendingFlag, 365
 MNVIC_vSetPriority, 367
 MNVIC_vSetPriorityConfig, 366
 NVIC_GetPriority, 368
 REGISTER_SIZE, 364
ODD_PARITY
 UART Parity, 122
ODRx
 GPIOx_MemoryMapType, 139
OSPEEDRx
 GPIOx_MemoryMapType, 138
OTG_FS
 NVIC Vector Table, 90
OTGFS
 MRCC_config.h, 377
OTYPERx
 GPIOx_MemoryMapType, 138
OutputSpeed
 GPIOx_ConfigType, 143
OutputType
 GPIOx_ConfigType, 143
OVER_SAMPLING_16
 UART Oversampling, 120
OVER_SAMPLING_8
 UART Oversampling, 120
Oversampling
 USART_InitType, 178
P2CONST
 Compiler standard macros, 20
P2FUNC
 Compiler standard macros, 20
P2VAR
 Compiler standard macros, 19
Parity_Enable
 USART_InitType, 177
Parity_Selection
 USART_InitType, 177
PEND_SV
 NVIC Settable Priorities, 94
PERIODIC_INTERVAL_MODE
 Systick Configuration, 112
PERIODIC_INTERVAL_TICK_TIME
 SysTick_interface.h, 475
periodicity
 Task, 169
Pin
 GPIOx_ConfigType, 143
PLL
 MRCC_config.h, 372
PLLCFGR
 RCC_MemoryMapType, 152
PLLCLK
 MRCC_private.h, 422, 423
PLLI2S
 MRCC_config.h, 372
PLLI2SCFGR
 RCC_MemoryMapType, 159
PLLI2SCLK
 MRCC_private.h, 421
PLLM
 MRCC_config.h, 374
PLLN
 MRCC_config.h, 374
PLLP
 MRCC_config.h, 374
PLLQ
 MRCC_config.h, 373
PLLSRC
 MRCC_config.h, 373
PMC
 MSYSCFG_MemMap_t, 145
Port
 GPIOx_ConfigType, 142
PORTA_BIT_MANIPULATION
 GPIO_config.h, 314
PORTB_BIT_MANIPULATION
 GPIO_config.h, 314
PortNum
 EXTI_ConfigType, 134
PPRE1
 MRCC_config.h, 375
PPRE2
 MRCC_config.h, 375
PR
 EXTI_Type, 137
PRESSED
 Standard values, 27
PSC
 TIM1_MemoryMapType, 172

PUPDRx
 GPIOx_MemoryMapType, 138

PWREN
 MRCC_config.h, 378

RCC
 MRCC_private.h, 404
 NVIC Vector Table, 79

RCC_AHB1
 ID options, 96

RCC_AHB1ENR_CRCEN
 MRCC_private.h, 415

RCC_AHB1ENR_DMA1EN
 MRCC_private.h, 415

RCC_AHB1ENR_DMA2EN
 MRCC_private.h, 415

RCC_AHB1ENR_GPIOAEN
 MRCC_private.h, 416

RCC_AHB1ENR_GPIOBEN
 MRCC_private.h, 416

RCC_AHB1ENR_GPIOCEN
 MRCC_private.h, 416

RCC_AHB1ENR_GPIODEN
 MRCC_private.h, 416

RCC_AHB1ENR_GPIOEEN
 MRCC_private.h, 415

RCC_AHB1ENR_GPIOHEN
 MRCC_private.h, 415

RCC_AHB1LPENR
 ID options, 97

RCC_AHB1LPENR_FLITFLPEN
 MRCC_private.h, 420

RCC_AHB2
 ID options, 96

RCC_AHB2ENR_OTGFSEN
 MRCC_private.h, 416

RCC_APB1
 ID options, 97

RCC_APB1ENR_I2C1EN
 MRCC_private.h, 417

RCC_APB1ENR_I2C2EN
 MRCC_private.h, 417

RCC_APB1ENR_I2C3EN
 MRCC_private.h, 417

RCC_APB1ENR_PWREN
 MRCC_private.h, 416

RCC_APB1ENR_SPI2EN
 MRCC_private.h, 417

RCC_APB1ENR_SPI3EN
 MRCC_private.h, 417

RCC_APB1ENR_TIM2EN
 MRCC_private.h, 418

RCC_APB1ENR_TIM3EN
 MRCC_private.h, 418

RCC_APB1ENR_TIM4EN
 MRCC_private.h, 418

RCC_APB1ENR_TIM5EN
 MRCC_private.h, 418

RCC_APB1ENR_USART2EN
 MRCC_private.h, 417

RCC_APB2
 ID options, 97

RCC_APB2ENR_ADC1EN
 MRCC_private.h, 420

RCC_APB2ENR_SDIOEN
 MRCC_private.h, 419

RCC_APB2ENR_SPI1EN
 MRCC_private.h, 419

RCC_APB2ENR_SPI4EN
 MRCC_private.h, 419

RCC_APB2ENR_SYSCFGEN
 MRCC_private.h, 419

RCC_APB2ENR_TIM10EN
 MRCC_private.h, 419

RCC_APB2ENR_TIM11EN
 MRCC_private.h, 418

RCC_APB2ENR_TIM1EN
 MRCC_private.h, 420

RCC_APB2ENR_TIM9EN
 MRCC_private.h, 419

RCC_APB2ENR_USART1EN
 MRCC_private.h, 420

RCC_APB2ENR_USART6EN
 MRCC_private.h, 420

RCC_BASE_ADDRESS
 MRCC_private.h, 404

RCC_CFGR_HPRE_b0
 MRCC_private.h, 413

RCC_CFGR_HPRE_b1
 MRCC_private.h, 414

RCC_CFGR_HPRE_b2
 MRCC_private.h, 414

RCC_CFGR_HPRE_b3
 MRCC_private.h, 414

RCC_CFGR_I2SSRC
 MRCC_private.h, 411

RCC_CFGR_MOC1_b0
 MRCC_private.h, 411

RCC_CFGR_MOC1_b1
 MRCC_private.h, 411

RCC_CFGR_MOC1PRE_b0
 MRCC_private.h, 411

RCC_CFGR_MOC1PRE_b1
 MRCC_private.h, 411

RCC_CFGR_MOC1PRE_b2
 MRCC_private.h, 411

RCC_CFGR_MOC2_b0
 MRCC_private.h, 410

RCC_CFGR_MOC2_b1
 MRCC_private.h, 410

RCC_CFGR_MOC2PRE_b0
 MRCC_private.h, 410

RCC_CFGR_MOC2PRE_b1
 MRCC_private.h, 410

RCC_CFGR_MOC2PRE_b2
 MRCC_private.h, 410

MRCC_private.h, 410
 RCC_CFGR_PPREG_b0
 MRCC_private.h, 413
 RCC_CFGR_PPREG_b1
 MRCC_private.h, 413
 RCC_CFGR_PPREG_b2
 MRCC_private.h, 413
 RCC_CFGR_PPREG_b0
 MRCC_private.h, 412
 RCC_CFGR_PPREG_b1
 MRCC_private.h, 413
 RCC_CFGR_PPREG_b2
 MRCC_private.h, 413
 RCC_CFGR_RTCPRE_b0
 MRCC_private.h, 412
 RCC_CFGR_RTCPRE_b1
 MRCC_private.h, 412
 RCC_CFGR_RTCPRE_b2
 MRCC_private.h, 412
 RCC_CFGR_RTCPRE_b3
 MRCC_private.h, 412
 RCC_CFGR_RTCPRE_b4
 MRCC_private.h, 412
 RCC_CFGR_SW_b0
 MRCC_private.h, 414
 RCC_CFGR_SW_b1
 MRCC_private.h, 415
 RCC_CFGR_SWS_b0
 MRCC_private.h, 414
 RCC_CFGR_SWS_b1
 MRCC_private.h, 414
 RCC_CR_CSSON
 MRCC_private.h, 405
 RCC_CR_HSEBYP
 MRCC_private.h, 405
 RCC_CR_HSEON
 MRCC_private.h, 406
 RCC_CR_HSIRDY
 MRCC_private.h, 405
 RCC_CR_HSION
 MRCC_private.h, 406
 RCC_CR_HSIRDY
 MRCC_private.h, 406
 RCC_CR_PLLI2SON
 MRCC_private.h, 405
 RCC_CR_PLLI2SRDY
 MRCC_private.h, 404
 RCC_CR_PLLON
 MRCC_private.h, 405
 RCC_CR_PLLRDY
 MRCC_private.h, 405
 RCC_MemoryMapType, 150
 AHB1ENR, 154
 AHB1LPENR, 156
 AHB1RSTR, 153
 AHB2ENR, 155
 AHB2LPENR, 156
 AHB2RSTR, 153
 APB1ENR, 155
 APB1LPENR, 157
 APB1RSTR, 154
 APB2ENR, 155
 APB2LPENR, 157
 APB2RSTR, 154
 BDCR, 158
 CFGR, 152
 CIR, 153
 CR, 152
 CSR, 158
 DCKCFG, 159
 PLLCFG, 152
 PLL2SCFG, 159
 Reserved1, 153
 Reserved10, 157
 Reserved11, 157
 Reserved12, 157
 Reserved13, 158
 Reserved14, 158
 Reserved2, 153
 Reserved3, 154
 Reserved4, 154
 Reserved5, 155
 Reserved6, 155
 Reserved7, 156
 Reserved8, 156
 Reserved9, 156
 SSCGR, 158
 RCC_PLLCFG_PLLM_b0
 MRCC_private.h, 409
 RCC_PLLCFG_PLLM_b1
 MRCC_private.h, 409
 RCC_PLLCFG_PLLM_b2
 MRCC_private.h, 409
 RCC_PLLCFG_PLLM_b3
 MRCC_private.h, 409
 RCC_PLLCFG_PLLM_b4
 MRCC_private.h, 409
 RCC_PLLCFG_PLLM_b5
 MRCC_private.h, 410
 RCC_PLLCFG_PLLN_b0
 MRCC_private.h, 407
 RCC_PLLCFG_PLLN_b1
 MRCC_private.h, 407
 RCC_PLLCFG_PLLN_b2
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b3
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b4
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b5
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b6
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b7
 MRCC_private.h, 408
 RCC_PLLCFG_PLLN_b8

MRCC_private.h, 409
RCC_PLLCFGR_PLLP_b0
 MRCC_private.h, 407
RCC_PLLCFGR_PLLP_b1
 MRCC_private.h, 407
RCC_PLLCFGR_PLLQ_b0
 MRCC_private.h, 406
RCC_PLLCFGR_PLLQ_b1
 MRCC_private.h, 406
RCC_PLLCFGR_PLLQ_b2
 MRCC_private.h, 406
RCC_PLLCFGR_PLLQ_b3
 MRCC_private.h, 407
RCC_PLLCFGR_PLLSRC
 MRCC_private.h, 407
RCR
 TIM1_MemoryMapType, 173
REGISTER_SIZE
 NVIC_program.c, 364
RELEASED
 Standard values, 28
RESERVED
 SCB_MemoryMapType, 164
RESERVED0x
 NVIC_MemoryMapType, 147
Reserved1
 RCC_MemoryMapType, 153
Reserved10
 RCC_MemoryMapType, 157
Reserved11
 RCC_MemoryMapType, 157
Reserved12
 RCC_MemoryMapType, 157
Reserved13
 RCC_MemoryMapType, 158
Reserved14
 RCC_MemoryMapType, 158
Reserved2
 RCC_MemoryMapType, 153
RESERVED2x
 NVIC_MemoryMapType, 148
Reserved3
 RCC_MemoryMapType, 154
RESERVED3x
 NVIC_MemoryMapType, 149
Reserved4
 RCC_MemoryMapType, 154
RESERVED4x
 NVIC_MemoryMapType, 149
Reserved5
 RCC_MemoryMapType, 155
Reserved6
 RCC_MemoryMapType, 155
Reserved7
 RCC_MemoryMapType, 156
Reserved8
 RCC_MemoryMapType, 156
Reserved9
 RCC_MemoryMapType, 156
RCC_MemoryMapType, 156
RSERVED1x
 NVIC_MemoryMapType, 147
RTCPRE
 MRCC_config.h, 375
RTSR
 EXTI_Type, 136
RUN
 Standard values, 27
RX_ONLY
 UART Operating Modes, 121
RXCRCR
 SPI_MemoryMapType, 167
s16_t
 Standard types, 24
s32_t
 Standard types, 24
s64_t
 Standard types, 24
s8_t
 Standard types, 24
SAME_STRING
 Standard values, 28
SCB_BASE_ADDRESS
 NVIC Addresses, 93
SCB_MemoryMapType, 159
 AIRCR, 161
 BFAR, 165
 CCR, 162
 CFSR, 163
 CPUID, 160
 HFSR, 164
 ICSR, 161
 MMFAR, 164
 RESERVED, 164
 SCR, 161
 SHCSR, 163
 SHPR1, 162
 SHPR2, 162
 SHPR3, 163
 VTOR, 161
SCR
 SCB_MemoryMapType, 161
SDIO
 NVIC Vector Table, 88
SDIOEN
 MRCC_config.h, 381
SEC
 Delay Units, 113
SET_BIT
 Bit Manipulation Math Macros, 16
SET_BITS
 Bit Group Manipulation Math Macros, 17
SHCSR
 SCB_MemoryMapType, 163
SHPR1
 SCB_MemoryMapType, 162
SHPR2

SCB_MemoryMapType, 162
 SHPR3
 SCB_MemoryMapType, 163
 SINGLE_INTERVAL_MODE
 Systick Configuration, 112
 SINGLE_INTERVAL_TICK_TIME
 SysTick_interface.h, 475
 SMCR
 TIM1_MemoryMapType, 171
 SMPR1
 ADC_MemoryMapType, 127
 SMPR2
 ADC_MemoryMapType, 127
 SPEED_0_PERCENT
 DC Motor Speeds, 12
 SPEED_100_PERCENT
 DC Motor Speeds, 15
 SPEED_10_PERCENT
 DC Motor Speeds, 12
 SPEED_20_PERCENT
 DC Motor Speeds, 13
 SPEED_30_PERCENT
 DC Motor Speeds, 13
 SPEED_40_PERCENT
 DC Motor Speeds, 13
 SPEED_50_PERCENT
 DC Motor Speeds, 13
 SPEED_60_PERCENT
 DC Motor Speeds, 14
 SPEED_70_PERCENT
 DC Motor Speeds, 14
 SPEED_80_PERCENT
 DC Motor Speeds, 14
 SPEED_90_PERCENT
 DC Motor Speeds, 14
 SPI Addresses, 108
 SPI1_BASE_ADDRESS, 108
 SPI2_BASE_ADDRESS, 108
 SPI3_BASE_ADDRESS, 108
 SPI4_BASE_ADDRESS, 108
 SPI Interfacing Macros, 110
 SPIx_FULL_DUPLEX, 111
 SPIx_HALF_DUPLEX, 111
 SPIx_MSTR, 110
 SPIx_SIMPLEX, 110
 SPIx_SLAVE, 110
 SPI Peripherals, 107
 SPI Registers, 109
 SPI1, 109
 SPI2, 109
 SPI3, 109
 SPI1
 NVIC Vector Table, 86
 SPI Registers, 109
 SPI1_BASE_ADDRESS
 SPI Addresses, 108
 SPI1EN
 MRCC_config.h, 380
 SPI2
 NVIC Vector Table, 86
 SPI Registers, 109
 SPI2_BASE_ADDRESS
 SPI Addresses, 108
 SPI2EN
 MRCC_config.h, 379
 SPI3
 NVIC Vector Table, 88
 SPI Registers, 109
 SPI3_BASE_ADDRESS
 SPI Addresses, 108
 SPI3EN
 MRCC_config.h, 378
 SPI4
 NVIC Vector Table, 92
 SPI4_BASE_ADDRESS
 SPI Addresses, 108
 SPI4EN
 MRCC_config.h, 380
 SPI_MemoryMapType, 165
 CR1, 166
 CR2, 166
 CRCPR, 166
 DR, 166
 I2SCFGR, 167
 I2SPR, 167
 RXCRCR, 167
 SR, 166
 TXCRCR, 167
 SPIx_FULL_DUPLEX
 SPI Interfacing Macros, 111
 SPIx_HALF_DUPLEX
 SPI Interfacing Macros, 111
 SPIx_MSTR
 SPI Interfacing Macros, 110
 SPIx_SIMPLEX
 SPI Interfacing Macros, 110
 SPIx_SLAVE
 SPI Interfacing Macros, 110
 SQR1
 ADC_MemoryMapType, 128
 SQR2
 ADC_MemoryMapType, 128
 SQR3
 ADC_MemoryMapType, 129
 SR
 ADC_MemoryMapType, 126
 SPI_MemoryMapType, 166
 TIM1_MemoryMapType, 171
 SR_REG
 USART_MemoryMapType, 179
 SSCGR
 RCC_MemoryMapType, 158
 Standard types, 22
 bool_t, 22
 c8_t, 23
 f32_t, 25

f64_t, 25
s16_t, 24
s32_t, 24
s64_t, 24
s8_t, 24
u16_t, 23
u32_t, 23
u64_t, 23
u8_t, 22
Standard values, 25
DIFFERENT_STRING, 28
FALSE, 26
FLAG_CLEARED, 27
FLAG_SET, 26
INITIAL_ZERO, 26
NULL, 26
PRESSED, 27
RELEASED, 28
RUN, 27
SAME_STRING, 28
STOP, 27
TRUE, 26
STATIC
 Compiler standard macros, 21
STOP
 Standard values, 27
STOP_BIT_0_5
 UART Stop Bits, 123
STOP_BIT_1
 UART Stop Bits, 123
STOP_BIT_1_5
 UART Stop Bits, 124
STOP_BIT_2
 UART Stop Bits, 124
StopBits
 USART_InitType, 177
Sub-Group priorities, 70
 NO_SUB_PRIORITY, 71
 SUB_PRIORITY_0, 71
 SUB_PRIORITY_1, 71
 SUB_PRIORITY_10, 74
 SUB_PRIORITY_11, 74
 SUB_PRIORITY_12, 74
 SUB_PRIORITY_13, 74
 SUB_PRIORITY_14, 75
 SUB_PRIORITY_15, 75
 SUB_PRIORITY_2, 72
 SUB_PRIORITY_3, 72
 SUB_PRIORITY_4, 72
 SUB_PRIORITY_5, 72
 SUB_PRIORITY_6, 73
 SUB_PRIORITY_7, 73
 SUB_PRIORITY_8, 73
 SUB_PRIORITY_9, 73
SUB_PRIORITY_0
 Sub-Group priorities, 71
SUB_PRIORITY_1
 Sub-Group priorities, 71
 SUB_PRIORITY_10
 Sub-Group priorities, 74
 SUB_PRIORITY_11
 Sub-Group priorities, 74
 SUB_PRIORITY_12
 Sub-Group priorities, 74
 SUB_PRIORITY_13
 Sub-Group priorities, 74
 SUB_PRIORITY_14
 Sub-Group priorities, 75
 SUB_PRIORITY_15
 Sub-Group priorities, 75
 SUB_PRIORITY_2
 Sub-Group priorities, 72
 SUB_PRIORITY_3
 Sub-Group priorities, 72
 SUB_PRIORITY_4
 Sub-Group priorities, 72
 SUB_PRIORITY_5
 Sub-Group priorities, 72
 SUB_PRIORITY_6
 Sub-Group priorities, 73
 SUB_PRIORITY_7
 Sub-Group priorities, 73
 SUB_PRIORITY_8
 Sub-Group priorities, 73
 SV_CALL
 NVIC Settable Priorities, 95
SW
 MRCC_config.h, 376
SWIER
 EXTI_Type, 136
SWS
 MRCC_config.h, 376
SYSCFG Memory Addresses, 43
 SYSCFG_BASE_ADDR, 43
SYSCFG Memory Registers, 43
 MSYSCFG, 43
SYSCFG_BASE_ADDR
 SYSCFG Memory Addresses, 43
SYSCFGEN
 MRCC_config.h, 380
SYSCLK
 MRCC_private.h, 421
SYSCLKby128
 MRCC_private.h, 424
SYSCLKby16
 MRCC_private.h, 424
SYSCLKby2
 MRCC_private.h, 423
SYSCLKby256
 MRCC_private.h, 424
SYSCLKby4
 MRCC_private.h, 423
SYSCLKby512
 MRCC_private.h, 424

SYSCLKby64
 MRCC_private.h, 424
 SYSCLKby8
 MRCC_private.h, 424
 SYSTICK
 NVIC Settable Priorities, 94
 SysTick
 Systick Registers, 114
 Systick Addresses, 114
 SysTick_BASE_ADDRESS, 114
 Systick Clock Sources, 116
 AHB, 117
 AHB_DividedBy8, 116
 Systick Configuration, 111
 CLK_SOURCE, 111
 Exception_Request, 112
 MAX_TICKS, 112
 PERIODIC_INTERVAL_MODE, 112
 SINGLE_INTERVAL_MODE, 112
 Systick Exception (Interrupt) Status, 117
 AssertRequest, 117
 Dont AssertRequest, 117
 Systick Register Bits Positions, 115
 CLKSOURCE, 115
 COUNTER_ENABLE, 116
 COUNTFLAG, 115
 TICKINT, 115
 Systick Registers, 114
 SysTick, 114
 SysTick_BASE_ADDRESS
 Systick Addresses, 114
 SysTick_Handler
 SysTick_program.c, 495
 SysTick_interface.h
 BUSY_TICK_TIME, 475
 MSysTick_u32GetElapsedTime, 482
 MSysTick_u32GetRemainingTime, 482
 MSysTick_vDelay, 477
 MSysTick_vDelayMicroSec, 478
 MSysTick_vDelayMilliSec, 478
 MSysTick_vDelaySec, 479
 MSysTick_vDisable, 483
 MSysTick_vDisableException, 483
 MSysTick_vEnable, 482
 MSysTick_vEnableException, 483
 MSysTick_vInit, 476
 MSysTick_vSetBusyWait, 476
 MSysTick_vSetPeriodicInterval, 480
 MSysTick_vSetSingleInterval, 480
 MSysTick_vStopInterval, 481
 PERIODIC_INTERVAL_TICK_TIME, 475
 SINGLE_INTERVAL_TICK_TIME, 475
 SysTick_program.c
 MSysTick_u32GetElapsedTime, 493
 MSysTick_u32GetRemainingTime, 494
 MSysTick_vDelay, 489
 MSysTick_vDelayMicroSec, 489
 MSysTick_vDelayMilliSec, 490
 MSysTick_vDelaySec, 491
 MSysTick_vDisable, 494
 MSysTick_vDisableException, 495
 MSysTick_vEnable, 494
 MSysTick_vEnableException, 495
 MSysTick_vInit, 487
 MSysTick_vSetBusyWait, 488
 MSysTick_vSetPeriodicInterval, 492
 MSysTick_vSetSingleInterval, 492
 MSysTick_vStopInterval, 493
 SysTick_Handler, 495
 SysTick_Type, 168
 CALIB, 169
 CTRL, 168
 LOAD, 168
 VAL, 168
 Task, 169
 periodicity, 169
 Task_RunningState, 170
 TaskHandler, 169
 Task_RunningState
 Task, 170
 TaskHandler
 Task, 169
 TICKINT
 Systick Register Bits Positions, 115
 TIM10EN
 MRCC_config.h, 380
 TIM11EN
 MRCC_config.h, 380
 TIM1_BRK_TIM9
 NVIC Vector Table, 83
 TIM1_CC
 NVIC Vector Table, 84
 TIM1_MemoryMapType, 170
 ARR, 172
 BDTR, 173
 CCER, 172
 CCMR1, 172
 CCMR2, 172
 CCR1, 173
 CCR2, 173
 CCR3, 173
 CCR4, 173
 CNT, 172
 CR1, 171
 CR2, 171
 DCR, 174
 DIER, 171
 DMAR, 174
 EGR, 171
 PSC, 172
 RCR, 173
 SMCR, 171
 SR, 171
 TIM1_TRG_COM_TIM11
 NVIC Vector Table, 83
 TIM1_UP_TIM10

NVIC Vector Table, 83

TIM1EN
 MRCC_config.h, 381

TIM2
 NVIC Vector Table, 84

TIM2EN
 MRCC_config.h, 379

TIM3
 NVIC Vector Table, 84

TIM3EN
 MRCC_config.h, 379

TIM4
 NVIC Vector Table, 84

TIM4EN
 MRCC_config.h, 379

TIM5
 NVIC Vector Table, 88

TIM5EN
 MRCC_config.h, 379

TIM9EN
 MRCC_config.h, 380

TOGGLE_BIT
 Bit Manipulation Math Macros, 16

TOGGLE_Bits
 Bit Group Manipulation Math Macros, 18

TransferDirection
 USART_InitType, 177

TriggerStatus
 EXTI_ConfigType, 135

TRUE
 Standard values, 26

TX_ONLY
 UART Operating Modes, 121

TX_RX
 UART Operating Modes, 121

TXCRCR
 SPI_MemoryMapType, 167

u16_t
 Standard types, 23

u32_t
 Standard types, 23

u32SpeedRatio
 DCM_MotorConfiguration, 133

u64_t
 Standard types, 23

u8_t
 Standard types, 22

u8Direction
 DCM_MotorConfiguration, 133

u8Pin
 BUZZER_BuzzerConfiguration, 131
 HULTSNC_ConfigType, 140
 LED_LEDConfiguration, 141

u8Pin1
 DCM_MotorConfiguration, 132

u8Pin2
 DCM_MotorConfiguration, 133

u8Port

 BUZZER_BuzzerConfiguration, 131

 DCM_MotorConfiguration, 132

 HULTSNC_ConfigType, 140

 LED_LEDConfiguration, 141

u8SpeedPin
 DCM_MotorConfiguration, 133

UART Addresses, 118
 USART1_BASE_ADDRESS, 118
 USART2_BASE_ADDRESS, 118
 USART6_BASE_ADDRESS, 118

UART Bit Sizes, 122
 MODE_8BIT, 122
 MODE_9BIT, 123

UART Operating Modes, 121
 RX_ONLY, 121
 TX_ONLY, 121
 TX_RX, 121

UART Oversampling, 120
 OVER_SAMPLING_16, 120
 OVER_SAMPLING_8, 120

UART Parity, 122
 EVEN_PARITY, 122
 ODD_PARITY, 122

UART Registers, 119
 USART1_REG, 119
 USART2_REG, 119
 USART6_REG, 119

UART Stop Bits, 123
 STOP_BIT_0_5, 123
 STOP_BIT_1, 123
 STOP_BIT_1_5, 124
 STOP_BIT_2, 124

UART_BRR_SAMPLING16
 UART_private.h, 558

UART_BRR_SAMPLING8
 UART_private.h, 559

UART_DIV_SAMPLING16
 UART_private.h, 557

UART_DIV_SAMPLING8
 UART_private.h, 558

UART_DIVFRAQ_SAMPLING16
 UART_private.h, 558

UART_DIVFRAQ_SAMPLING8
 UART_private.h, 559

UART_DIVMANT_SAMPLING16
 UART_private.h, 558

UART_DIVMANT_SAMPLING8
 UART_private.h, 559

UART_interface.h
 __BAUDRATE__, 544
 DISABLE, 544
 ENABLE, 544
 MUSART1_vSetCallBack, 553
 MUSART2_vSetCallBack, 553
 MUSART6_vSetCallBack, 554
 MUSART_ptrReceiveStringSynchNonBlocking,
 549
 MUSART_u8CompareString, 551

MUSART_u8ReadDataRegister, 551
 MUSART_u8ReceiveByteSynchBlocking, 549
 MUSART_u8ReceiveByteSynchNonBlocking, 548
 MUSART_vClearFlags, 552
 MUSART_vDisable, 547
 MUSART_vEnable, 546
 MUSART_vInit, 544
 MUSART_vRecieveString, 550
 MUSART_vRxIntSetStatus, 552
 MUSART_vTransmitByte, 547
 MUSART_vTransmitString, 548
UART_private.h
 MUSART_CR1_IDLEIE_BIT, 562
 MUSART_CR1_M_BIT, 563
 MUSART_CR1_OVER8_BIT, 563
 MUSART_CR1_PCE_BIT, 563
 MUSART_CR1_PEIE_BIT, 562
 MUSART_CR1_PS_BIT, 562
 MUSART_CR1_RE_BIT, 561
 MUSART_CR1_RWU_BIT, 561
 MUSART_CR1_RXNEIE_BIT, 562
 MUSART_CR1_SBK_BIT, 561
 MUSART_CR1_TCIE_BIT, 562
 MUSART_CR1_TE_BIT, 561
 MUSART_CR1_TXEIE_BIT, 562
 MUSART_CR1_UE_BIT, 563
 MUSART_CR1_WAKE_BIT, 563
 MUSART_CR2_ADD0_BIT, 563
 MUSART_CR2_ADD1_BIT, 564
 MUSART_CR2_ADD2_BIT, 564
 MUSART_CR2_ADD3_BIT, 564
 MUSART_CR2_CLKEN_BIT, 565
 MUSART_CR2_CPHA_BIT, 565
 MUSART_CR2_CPOL_BIT, 565
 MUSART_CR2_LBCL_BIT, 564
 MUSART_CR2_LBDIE_BIT, 564
 MUSART_CR2_LBDL_BIT, 564
 MUSART_CR2_LINEN_BIT, 566
 MUSART_CR2_STOP0_BIT, 565
 MUSART_CR2_STOP1_BIT, 565
 MUSART_CR2_STOP_BIT, 565
 MUSART_CR3_CTSE_BIT, 566
 MUSART_CR3_CTSIE_BIT, 566
 MUSART_CR3_DMAR_BIT, 566
 MUSART_CR3_DMAT_BIT, 566
 MUSART_CR3_EIE_BIT, 567
 MUSART_CR3_HDSEL_BIT, 567
 MUSART_CR3_IREN_BIT, 567
 MUSART_CR3_IRLP_BIT, 567
 MUSART_CR3_NACK_BIT, 567
 MUSART_CR3_RTSE_BIT, 566
 MUSART_CR3_SCEN_BIT, 567
 MUSART_SR_CTS_BIT, 561
 MUSART_SR_FE_BIT, 559
 MUSART_SR_IDLE_BIT, 560
 MUSART_SR_LBD_BIT, 561
 MUSART_SR_NE_BIT, 560
 MUSART_SR_ORE_BIT, 560
 MUSART_SR_PE_BIT, 559
 MUSART_SR_RXNE_BIT, 560
 MUSART_SR_TC_BIT, 560
 MUSART_SR_TXE_BIT, 560
 UART_BRR_SAMPLING16, 558
 UART_BRR_SAMPLING8, 559
 UART_DIV_SAMPLING16, 557
 UART_DIV_SAMPLING8, 558
 UART_DIVFRAQ_SAMPLING16, 558
 UART_DIVFRAQ_SAMPLING8, 559
 UART_DIVMANT_SAMPLING16, 558
 UART_DIVMANT_SAMPLING8, 559
UltraSonic_interface.h
 HULTSNC_f32GetDistance, 248
 HULTSNC_vInit, 247
 HULTSNC_vTrigger, 248
USAGE_FAULT
 NVIC Settable Priorities, 95
USART1
 NVIC Vector Table, 86
USART1_BASE_ADDRESS
 UART Addresses, 118
USART1_REG
 UART Registers, 119
USART1EN
 MRCC_config.h, 381
USART2
 NVIC Vector Table, 86
USART2_BASE_ADDRESS
 UART Addresses, 118
USART2_REG
 UART Registers, 119
USART2EN
 MRCC_config.h, 378
USART6
 NVIC Vector Table, 91
USART6_BASE_ADDRESS
 UART Addresses, 118
USART6_REG
 UART Registers, 119
USART6EN
 MRCC_config.h, 381
USART_ClockInitTypeDef, 174
 ClockOutput, 175
 ClockPhase, 175
 ClockPolarity, 175
 LastBitClockPulse, 175
USART_InitType, 176
 BaudRate, 176
 DataWidth, 176
 HardwareFlowControl, 178
 Oversampling, 178
 Parity_Enable, 177
 Parity_Selection, 177
 StopBits, 177
 TransferDirection, 177
USART_MemoryMapType, 178
 BRR_REG, 179

CR1_REG, [180](#)
CR2_REG, [180](#)
CR3_REG, [180](#)
DR_REG, [179](#)
GTPR_REG, [180](#)
SR_REG, [179](#)
Utilities macros, [21](#)
 MY_MS_DELAY, [21](#)

VAL
 SysTick_Type, [168](#)

VAR
 Compiler standard macros, [19](#)

VECTKEY_PASSWORD
 NVIC_private.h, [361](#)

VTOR
 SCB_MemoryMapType, [161](#)

WWDG
 NVIC Vector Table, [78](#)

WWDGEN
 MRCC_config.h, [379](#)