

ADAS Graduation Project

Generated on Sat Jan 21 2023 15:01:00 for ADAS Graduation Project by Doxygen 1.9.5

Sat Jan 21 2023 15:01:00

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 Health Care	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	9
5.1 DC Motor Directions	9
5.1.1 Detailed Description	9
5.1.2 Macro Definition Documentation	9
5.1.2.1 FORWARD	9
5.1.2.2 BACKWARD	9
5.2 DC Motor Rotation Directions	10
5.2.1 Detailed Description	10
5.2.2 Macro Definition Documentation	10
5.2.2.1 CW	10
5.2.2.2 CCW	10
5.3 Bit Manipulation Math Macros	10
5.3.1 Detailed Description	11
5.3.2 Macro Definition Documentation	11
5.3.2.1 SET_BIT	11
5.3.2.2 CLR_BIT	11
5.3.2.3 TOGGLE_BIT	11
5.3.2.4 GET_BIT	12
5.4 Bit Group Manipulation Math Macros	12
5.4.1 Detailed Description	12
5.4.2 Macro Definition Documentation	12
5.4.2.1 SET_BITS	12
5.4.2.2 CLR_BITS	13
5.4.2.3 TOGGLE_BITS	13
5.4.2.4 GET_BITS	13
5.5 Compiler standard macros	14
5.5.1 Detailed Description	14
5.5.2 Macro Definition Documentation	14

5.5.2.1 VAR	14
5.5.2.2 FUNC	14
5.5.2.3 P2VAR	15
5.5.2.4 P2CONST	15
5.5.2.5 CONSTP2VAR	15
5.5.2.6 CONSTP2CONST	15
5.5.2.7 P2FUNC	16
5.5.2.8 CONST	16
5.5.2.9 STATIC	16
5.6 Utilities macros	16
5.6.1 Detailed Description	16
5.6.2 Macro Definition Documentation	16
5.6.2.1 MY_MS_DELAY	17
5.7 Standard types	17
5.7.1 Detailed Description	17
5.7.2 Typedef Documentation	17
5.7.2.1 bool_t	17
5.7.2.2 u8_t	18
5.7.2.3 c8_t	18
5.7.2.4 u16_t	18
5.7.2.5 u32_t	18
5.7.2.6 u64_t	19
5.7.2.7 s64_t	19
5.7.2.8 s8_t	19
5.7.2.9 s16_t	19
5.7.2.10 s32_t	20
5.7.2.11 f32_t	20
5.7.2.12 f64_t	20
5.8 Standard values	20
5.8.1 Detailed Description	21
5.8.2 Macro Definition Documentation	21
5.8.2.1 TRUE	21
5.8.2.2 FALSE	21
5.8.2.3 NULL	21
5.8.2.4 INITIAL_ZERO	21
5.8.2.5 FLAG_SET	22
5.8.2.6 FLAG_CLEARED	22
5.8.2.7 RUN	22
5.8.2.8 STOP	22
5.8.2.9 PRESSED	23
5.8.2.10 RELEASED	23
5.8.2.11 SAME_STRING	23

5.8.2.12 DIFFERENT_STRING	23
5.9 Interrupt line IDs	24
5.9.1 Detailed Description	24
5.9.2 Macro Definition Documentation	24
5.9.2.1 EXTI_LINE0	24
5.9.2.2 EXTI_LINE1	25
5.9.2.3 EXTI_LINE2	25
5.9.2.4 EXTI_LINE3	25
5.9.2.5 EXTI_LINE4	25
5.9.2.6 EXTI_LINE5	26
5.9.2.7 EXTI_LINE6	26
5.9.2.8 EXTI_LINE7	26
5.9.2.9 EXTI_LINE8	26
5.9.2.10 EXTI_LINE9	27
5.9.2.11 EXTI_LINE10	27
5.9.2.12 EXTI_LINE11	27
5.9.2.13 EXTI_LINE12	27
5.9.2.14 EXTI_LINE13	28
5.9.2.15 EXTI_LINE14	28
5.9.2.16 EXTI_LINE15	28
5.10 Interrupt trigger status	28
5.10.1 Detailed Description	28
5.10.2 Macro Definition Documentation	28
5.10.2.1 EXTI_FallingEdge	29
5.10.2.2 EXTI_RisingEdge	29
5.10.2.3 EXTI_OnChange	29
5.11 EXTI Memory Addresses	29
5.11.1 Detailed Description	29
5.11.2 Macro Definition Documentation	29
5.11.2.1 EXTI_BASE_ADDRESS	30
5.12 EXTI Registers	30
5.12.1 Detailed Description	30
5.12.2 Macro Definition Documentation	30
5.12.2.1 MEXTI	30
5.13 EXTI Lines Status	30
5.13.1 Detailed Description	30
5.13.2 Macro Definition Documentation	30
5.13.2.1 ENABLE	31
5.13.2.2 DISABLE	31
5.14 EXTI Line Settings	31
5.14.1 Detailed Description	31
5.14.2 Macro Definition Documentation	31

5.14.2.1 EXTI_MAX_EXTI_NUM	31
5.15 SYSCFG Memory Addresses	32
5.15.1 Detailed Description	32
5.15.2 Macro Definition Documentation	32
5.15.2.1 SYSCFG_BASE_ADDR	32
5.16 SYSCFG Memory Registers	32
5.16.1 Detailed Description	32
5.16.2 Macro Definition Documentation	32
5.16.2.1 MSYSCFG	32
5.17 GPIO Modes	33
5.17.1 Detailed Description	33
5.17.2 Macro Definition Documentation	33
5.17.2.1 GPIOx_MODE_INPUT	33
5.17.2.2 GPIOx_MODE_OUTPUT	33
5.17.2.3 GPIOx_MODE_AF	33
5.17.2.4 GPIOx_MODE_ANALOG	34
5.18 GPIO Ports	34
5.18.1 Detailed Description	34
5.18.2 Macro Definition Documentation	34
5.18.2.1 GPIO_PORTA	34
5.18.2.2 GPIO_PORTB	34
5.18.2.3 GPIO_PORTC	35
5.19 GPIO Output Types	35
5.19.1 Detailed Description	35
5.19.2 Macro Definition Documentation	35
5.19.2.1 GPIOx_OPENDRAIN	35
5.19.2.2 GPIOx_PUSH_PULL	35
5.20 GPIO PIN Speed	36
5.20.1 Detailed Description	36
5.20.2 Macro Definition Documentation	36
5.20.2.1 GPIOx_LowSpeed	36
5.20.2.2 GPIOx_MediumSpeed	36
5.20.2.3 GPIOx_HighSpeed	36
5.20.2.4 GPIOx_VeryHighSpeed	37
5.21 GPIO Pull Types	37
5.21.1 Detailed Description	37
5.21.2 Macro Definition Documentation	37
5.21.2.1 GPIOx_NoPull	37
5.21.2.2 GPIOx_PullUp	37
5.21.2.3 GPIOx_PullDown	38
5.22 GPIO Output Values	38
5.22.1 Detailed Description	38

5.22.2 Macro Definition Documentation	38
5.22.2.1 GPIOx_HIGH	38
5.22.2.2 GPIOx_LOW	38
5.23 GPIO Output PINs	39
5.23.1 Detailed Description	39
5.23.2 Macro Definition Documentation	39
5.23.2.1 GPIOx_PIN0	39
5.23.2.2 GPIOx_PIN1	40
5.23.2.3 GPIOx_PIN2	40
5.23.2.4 GPIOx_PIN3	40
5.23.2.5 GPIOx_PIN4	40
5.23.2.6 GPIOx_PIN5	41
5.23.2.7 GPIOx_PIN6	41
5.23.2.8 GPIOx_PIN7	41
5.23.2.9 GPIOx_PIN8	41
5.23.2.10 GPIOx_PIN9	42
5.23.2.11 GPIOx_PIN10	42
5.23.2.12 GPIOx_PIN11	42
5.23.2.13 GPIOx_PIN12	42
5.23.2.14 GPIOx_PIN13	43
5.23.2.15 GPIOx_PIN14	43
5.23.2.16 GPIOx_PIN15	43
5.24 GPIO Alternate Functions	43
5.24.1 Detailed Description	44
5.24.2 Macro Definition Documentation	44
5.24.2.1 GPIOx_AF0	44
5.24.2.2 GPIOx_AF1	44
5.24.2.3 GPIOx_AF2	45
5.24.2.4 GPIOx_AF3	45
5.24.2.5 GPIOx_AF4	45
5.24.2.6 GPIOx_AF5	45
5.24.2.7 GPIOx_AF6	46
5.24.2.8 GPIOx_AF7	46
5.24.2.9 GPIOx_AF8	46
5.24.2.10 GPIOx_AF9	46
5.24.2.11 GPIOx_AF10	47
5.24.2.12 GPIOx_AF11	47
5.24.2.13 GPIOx_AF12	47
5.24.2.14 GPIOx_AF13	47
5.24.2.15 GPIOx_AF14	48
5.24.2.16 GPIOx_AF15	48
5.25 GPIO Addresses	48

5.25.1 Detailed Description	48
5.25.2 Macro Definition Documentation	48
5.25.2.1 GPIOA_BASE_ADDRESS	48
5.25.2.2 GPIOB_BASE_ADDRESS	49
5.25.2.3 GPIOC_BASE_ADDRESS	49
5.25.2.4 CLK_SOURCE	49
5.26 GPIO Registers	49
5.26.1 Detailed Description	49
5.26.2 Macro Definition Documentation	49
5.26.2.1 GPIOA	50
5.26.2.2 GPIOB	50
5.26.2.3 GPIOC	50
5.27 Interrupt priority grouping	50
5.27.1 Detailed Description	51
5.27.2 Macro Definition Documentation	51
5.27.2.1 _16GROUP_NoSub_Priorities	51
5.27.2.2 _8GROUP_2Sub_Priorities	51
5.27.2.3 _4GROUP_4Sub_Priorities	52
5.27.2.4 _2GROUP_8Sub_Priorities	52
5.27.2.5 NoGROUP_16Sub_Priorities	52
5.27.2.6 GROUP_4BITS	53
5.27.2.7 GROUP_3BITS	53
5.27.2.8 GROUP_2BITS	53
5.27.2.9 GROUP_1BITS	53
5.27.2.10 GROUP_0BITS	54
5.28 Group priorities	54
5.28.1 Detailed Description	55
5.28.2 Macro Definition Documentation	55
5.28.2.1 NO_GROUP_PRIORITY	55
5.28.2.2 GROUP_PRIORITY_0	55
5.28.2.3 GROUP_PRIORITY_1	55
5.28.2.4 GROUP_PRIORITY_2	56
5.28.2.5 GROUP_PRIORITY_3	56
5.28.2.6 GROUP_PRIORITY_4	56
5.28.2.7 GROUP_PRIORITY_5	56
5.28.2.8 GROUP_PRIORITY_6	57
5.28.2.9 GROUP_PRIORITY_7	57
5.28.2.10 GROUP_PRIORITY_8	57
5.28.2.11 GROUP_PRIORITY_9	57
5.28.2.12 GROUP_PRIORITY_10	58
5.28.2.13 GROUP_PRIORITY_11	58
5.28.2.14 GROUP_PRIORITY_12	58

5.28.2.15 GROUP_PRIORITY_13	58
5.28.2.16 GROUP_PRIORITY_14	59
5.28.2.17 GROUP_PRIORITY_15	59
5.29 Sub-Group priorities	59
5.29.1 Detailed Description	60
5.29.2 Macro Definition Documentation	60
5.29.2.1 NO_SUB_PRIORITY	60
5.29.2.2 SUB_PRIORITY_0	60
5.29.2.3 SUB_PRIORITY_1	61
5.29.2.4 SUB_PRIORITY_3	61
5.29.2.5 SUB_PRIORITY_2	61
5.29.2.6 SUB_PRIORITY_4	61
5.29.2.7 SUB_PRIORITY_5	62
5.29.2.8 SUB_PRIORITY_6	62
5.29.2.9 SUB_PRIORITY_7	62
5.29.2.10 SUB_PRIORITY_8	62
5.29.2.11 SUB_PRIORITY_9	63
5.29.2.12 SUB_PRIORITY_10	63
5.29.2.13 SUB_PRIORITY_11	63
5.29.2.14 SUB_PRIORITY_12	63
5.29.2.15 SUB_PRIORITY_13	64
5.29.2.16 SUB_PRIORITY_14	64
5.29.2.17 SUB_PRIORITY_15	64
5.30 NVIC Vector Table	64
5.30.1 Detailed Description	67
5.30.2 Macro Definition Documentation	67
5.30.2.1 WWDG	67
5.30.2.2 EXTI16	67
5.30.2.3 EXTI21	68
5.30.2.4 EXTI22	68
5.30.2.5 FLASH	68
5.30.2.6 RCC	68
5.30.2.7 EXTI0	69
5.30.2.8 EXTI1	69
5.30.2.9 EXTI2	69
5.30.2.10 EXTI3	69
5.30.2.11 EXTI4	70
5.30.2.12 DMA1_STREAM0	70
5.30.2.13 DMA1_STREAM1	70
5.30.2.14 DMA1_STREAM2	70
5.30.2.15 DMA1_STREAM3	71
5.30.2.16 DMA1_STREAM4	71

5.30.2.17 DMA1_STREAM5	71
5.30.2.18 DMA1_STREAM6	71
5.30.2.19 ADC	72
5.30.2.20 EXTI9	72
5.30.2.21 TIM1_BRK_TIM9	72
5.30.2.22 TIM1_UP_TIM10	72
5.30.2.23 TIM1_TRG_COM_TIM11	73
5.30.2.24 TIM1_CC	73
5.30.2.25 TIM2	73
5.30.2.26 TIM3	73
5.30.2.27 TIM4	74
5.30.2.28 I2C1_EV	74
5.30.2.29 I2C1_ER	74
5.30.2.30 I2C2_EV	74
5.30.2.31 I2C2_ER	75
5.30.2.32 SPI1	75
5.30.2.33 SPI2	75
5.30.2.34 USART1	75
5.30.2.35 USART2	76
5.30.2.36 EXTI15_10	76
5.30.2.37 EXTI17	76
5.30.2.38 EXTI18	76
5.30.2.39 DMA1_STREAM7	77
5.30.2.40 SDIO	77
5.30.2.41 TIM5	77
5.30.2.42 SPI3	77
5.30.2.43 DMA2_STREAM0	78
5.30.2.44 DMA2_STREAM1	78
5.30.2.45 DMA2_STREAM2	78
5.30.2.46 DMA2_STREAM5	78
5.30.2.47 DMA2_STREAM3	79
5.30.2.48 DMA2_STREAM4	79
5.30.2.49 OTG_FS	79
5.30.2.50 DMA2_STREAM6	79
5.30.2.51 DMA2_STREAM7	80
5.30.2.52 USART6	80
5.30.2.53 I2C3_EV	80
5.30.2.54 I2C3_ER	80
5.30.2.55 FPU	80
5.30.2.56 SPI4	81
5.31 NVIC Addresses	81
5.31.1 Detailed Description	81

5.31.2 Macro Definition Documentation	81
5.31.2.1 NVIC_BASE_ADDRESS	81
5.31.2.2 SCB_BASE_ADDRESS	81
5.32 NVIC Registers	82
5.32.1 Detailed Description	82
5.32.2 Macro Definition Documentation	82
5.32.2.1 MSCB	82
5.32.2.2 MNVIC	82
5.33 NVIC Settable Priorities	82
5.33.1 Detailed Description	83
5.33.2 Macro Definition Documentation	83
5.33.2.1 PEND_SV	83
5.33.2.2 SYSTICK	83
5.33.2.3 SV_CALL	83
5.33.2.4 MEMORY_MANAGE	84
5.33.2.5 BUS_FAULT	84
5.33.2.6 USAGE_FAULT	84
5.34 ID options	84
5.34.1 Detailed Description	84
5.34.2 Macro Definition Documentation	84
5.34.2.1 RCC_AHB1	85
5.34.2.2 RCC_AHB2	85
5.34.2.3 RCC_APB1	85
5.34.2.4 RCC_APB2	85
5.34.2.5 RCC_AHB1LPENR	86
5.35 ID options	86
5.35.1 Detailed Description	87
5.35.2 Macro Definition Documentation	87
5.35.2.1 AHB1ENR_DMA2EN	87
5.35.2.2 AHB1ENR_DMA1EN	88
5.35.2.3 AHB1ENR_CRCEN	88
5.35.2.4 AHB1ENR_GPIOHEN	88
5.35.2.5 AHB1ENR_GPIOEEN	88
5.35.2.6 AHB1ENR_GPIODEN	89
5.35.2.7 AHB1ENR_GPIOCEN	89
5.35.2.8 AHB1ENR_GPIOBEN	89
5.35.2.9 AHB1ENR_GPIOAEN	89
5.35.2.10 AHB2ENR_OTGFSEN	90
5.35.2.11 APB1ENR_PWREN	90
5.35.2.12 APB1ENR_I2C3EN	90
5.35.2.13 APB1ENR_I2C2EN	90
5.35.2.14 APB1ENR_I2C1EN	91

5.35.2.15 APB1ENR_USART2EN	91
5.35.2.16 APB1ENR_SPI3EN	91
5.35.2.17 APB1ENR_SPI2EN	91
5.35.2.18 APB1ENR_WWDGEN	92
5.35.2.19 APB1ENR_TIM5EN	92
5.35.2.20 APB1ENR_TIM4EN	92
5.35.2.21 APB1ENR_TIM3EN	92
5.35.2.22 APB1ENR_TIM2EN	93
5.35.2.23 APB2ENR_TIM11EN	93
5.35.2.24 APB2ENR_TIM10EN	93
5.35.2.25 APB2ENR_TIM9EN	93
5.35.2.26 APB2ENR_SYSCFGGEN	94
5.35.2.27 APB2ENR_SPI4EN	94
5.35.2.28 APB2ENR_SPI1EN	94
5.35.2.29 APB2ENR_SDIOEN	94
5.35.2.30 APB2ENR_ADC1EN	95
5.35.2.31 APB2ENR_USART6EN	95
5.35.2.32 APB2ENR_USART1EN	95
5.35.2.33 APB2ENR_TIM1EN	95
5.35.2.34 AHB1LPENR_FLITFLPEN	96
5.36 Systick Interval Mode Configuration	96
5.36.1 Detailed Description	96
5.36.2 Macro Definition Documentation	96
5.36.2.1 SINGLE_INTERVAL_MODE	96
5.36.2.2 PERIODIC_INTERVAL_MODE	96
5.37 Delay Units	97
5.37.1 Detailed Description	97
5.37.2 Macro Definition Documentation	97
5.37.2.1 MILLI_SEC	97
5.37.2.2 MICRO_SEC	97
5.37.2.3 SEC	97
5.38 Systick Addresses	98
5.38.1 Detailed Description	98
5.38.2 Macro Definition Documentation	98
5.38.2.1 SysTick_BASE_ADDRESS	98
5.39 Systick Registers	98
5.39.1 Detailed Description	98
5.39.2 Macro Definition Documentation	98
5.39.2.1 SysTick	98
5.40 Systick Register Bits Positions	99
5.40.1 Detailed Description	99
5.40.2 Macro Definition Documentation	99

5.40.2.1 COUNTFLAG	99
5.40.2.2 CLKSOURCE	99
5.40.2.3 TICKINT	99
5.40.2.4 COUNTER_ENABLE	100
5.41 Systick Clock Sources	100
5.41.1 Detailed Description	100
5.41.2 Macro Definition Documentation	100
5.41.2.1 AHB_DividedBy8	100
5.41.2.2 AHB	100
5.42 Systick Exception (Interrupt) Status	101
5.42.1 Detailed Description	101
5.42.2 Macro Definition Documentation	101
5.42.2.1 Dont AssertRequest	101
5.42.2.2 AssertRequest	101
6 Data Structure Documentation	103
6.1 ADC_MemoryMapType Struct Reference	103
6.1.1 Detailed Description	104
6.1.2 Field Documentation	104
6.1.2.1 SR	104
6.1.2.2 CR1	104
6.1.2.3 CR2	105
6.1.2.4 SMPR1	105
6.1.2.5 SMPR2	105
6.1.2.6 JOFR1	105
6.1.2.7 JOFR2	105
6.1.2.8 JOFR3	106
6.1.2.9 JOFR4	106
6.1.2.10 HTR	106
6.1.2.11 LTR	106
6.1.2.12 SQR1	106
6.1.2.13 SQR2	107
6.1.2.14 SQR3	107
6.1.2.15 JSQR	107
6.1.2.16 JDR1	107
6.1.2.17 JDR2	107
6.1.2.18 JDR3	108
6.1.2.19 JDR4	108
6.1.2.20 DR	108
6.1.2.21 CCR	108
6.2 DCM_MotorConfiguration Struct Reference	108
6.2.1 Detailed Description	109

6.2.2 Field Documentation	109
6.2.2.1 u8Port	109
6.2.2.2 u8Pin1	109
6.2.2.3 u8Pin2	110
6.2.2.4 u8Direction	110
6.3 EXTI_ConfigType Struct Reference	110
6.3.1 Detailed Description	110
6.3.2 Field Documentation	111
6.3.2.1 LineNum	111
6.3.2.2 PortNum	111
6.3.2.3 TriggerStatus	111
6.4 EXTI_Type Struct Reference	111
6.4.1 Detailed Description	112
6.4.2 Field Documentation	112
6.4.2.1 IMR	112
6.4.2.2 EMR	112
6.4.2.3 RTSR	113
6.4.2.4 FTSR	113
6.4.2.5 SWIER	113
6.4.2.6 PR	113
6.5 GPIOx_MemoryMapType Struct Reference	113
6.5.1 Detailed Description	114
6.5.2 Field Documentation	114
6.5.2.1 MODERx	114
6.5.2.2 OTYPERx	114
6.5.2.3 OSPEEDRx	115
6.5.2.4 PUPDRx	115
6.5.2.5 IDRx	115
6.5.2.6 ODRx	115
6.5.2.7 BSRRx	115
6.5.2.8 LCKRx	116
6.5.2.9 AFRLx	116
6.5.2.10 AFRHx	116
6.6 MADC_ConfigType Struct Reference	116
6.6.1 Detailed Description	116
6.6.2 Field Documentation	117
6.6.2.1 Port	117
6.7 MGPIOp_ConfigType Struct Reference	117
6.7.1 Detailed Description	117
6.7.2 Field Documentation	117
6.7.2.1 Port	118
6.7.2.2 Pin	118

6.7.2.3 Mode	118
6.7.2.4 OutputType	118
6.7.2.5 OutputSpeed	119
6.7.2.6 InputType	119
6.7.2.7 AF_Type	119
6.8 MSYSCFG_MemMap_t Struct Reference	119
6.8.1 Detailed Description	120
6.8.2 Field Documentation	120
6.8.2.1 MEMRMP	120
6.8.2.2 PMC	120
6.8.2.3 EXTICR	120
6.8.2.4 CMPCR	121
6.9 NVIC_MemoryMapType Struct Reference	121
6.9.1 Detailed Description	121
6.9.2 Field Documentation	121
6.9.2.1 ISERx	122
6.9.2.2 RESERVED0x	122
6.9.2.3 ICERx	122
6.9.2.4 RSERVED1x	123
6.9.2.5 ISPRx	123
6.9.2.6 RESERVED2x	123
6.9.2.7 ICPRx	124
6.9.2.8 RESERVED3x	124
6.9.2.9 IABRx	124
6.9.2.10 RESERVED4x	125
6.9.2.11 IPRx	125
6.10 RCC_MemoryMapType Struct Reference	125
6.10.1 Detailed Description	127
6.10.2 Field Documentation	127
6.10.2.1 CR	127
6.10.2.2 PLLCFG	127
6.10.2.3 CFGR	128
6.10.2.4 CIR	128
6.10.2.5 AHB1RSTR	128
6.10.2.6 AHB2RSTR	128
6.10.2.7 Reserved1	128
6.10.2.8 Reserved2	129
6.10.2.9 APB1RSTR	129
6.10.2.10 APB2RSTR	129
6.10.2.11 Reserved3	129
6.10.2.12 Reserved4	129
6.10.2.13 AHB1ENR	130

6.10.2.14 AHB2ENR	130
6.10.2.15 Reserved5	130
6.10.2.16 Reserved6	130
6.10.2.17 APB1ENR	130
6.10.2.18 APB2ENR	131
6.10.2.19 Reserved7	131
6.10.2.20 Reserved8	131
6.10.2.21 AHB1LPENR	131
6.10.2.22 AHB2LPENR	131
6.10.2.23 Reserved9	132
6.10.2.24 Reserved10	132
6.10.2.25 APB1LPENR	132
6.10.2.26 APB2LPENR	132
6.10.2.27 Reserved11	132
6.10.2.28 Reserved12	133
6.10.2.29 BDCR	133
6.10.2.30 CSR	133
6.10.2.31 Reserved13	133
6.10.2.32 Reserved14	133
6.10.2.33 SSCGR	134
6.10.2.34 PLLI2SCFGR	134
6.10.2.35 DCKCFGR	134
6.11 SCB_MemoryMapType Struct Reference	134
6.11.1 Detailed Description	135
6.11.2 Field Documentation	135
6.11.2.1 CPUID	136
6.11.2.2 ICSR	136
6.11.2.3 VTOR	136
6.11.2.4 AIRCR	136
6.11.2.5 SCR	137
6.11.2.6 CCR	137
6.11.2.7 SHPR1	137
6.11.2.8 SHPR2	138
6.11.2.9 SHPR3	138
6.11.2.10 SHCSR	138
6.11.2.11 CFSR	139
6.11.2.12 HFSR	139
6.11.2.13 RESERVED	139
6.11.2.14 MMFAR	140
6.11.2.15 BFAR	140
6.12 SPI_MemoryMapType Struct Reference	140
6.12.1 Detailed Description	140

6.12.2 Field Documentation	141
6.12.2.1 CR1	141
6.12.2.2 CR2	141
6.12.2.3 SR	141
6.12.2.4 DR	141
6.12.2.5 CRCPR	141
6.12.2.6 RXCRCR	142
6.12.2.7 TXCRCR	142
6.12.2.8 I2SCFGR	142
6.12.2.9 I2SPR	142
6.13 SysTick_Type Struct Reference	142
6.13.1 Detailed Description	143
6.13.2 Field Documentation	143
6.13.2.1 CTRL	143
6.13.2.2 LOAD	143
6.13.2.3 VAL	143
6.13.2.4 CALIB	144
6.14 USART_ClockInitTypeDef Struct Reference	144
6.14.1 Detailed Description	144
6.14.2 Field Documentation	144
6.14.2.1 ClockOutput	144
6.14.2.2 ClockPolarity	145
6.14.2.3 ClockPhase	145
6.14.2.4 LastBitClockPulse	145
6.15 USART_InitTypeDef Struct Reference	145
6.15.1 Detailed Description	146
6.15.2 Field Documentation	146
6.15.2.1 BaudRate	146
6.15.2.2 DataWidth	146
6.15.2.3 StopBits	146
6.15.2.4 Parity_Enable	146
6.15.2.5 Parity_Selection	147
6.15.2.6 TransferDirection	147
6.15.2.7 HardwareFlowControl	147
6.15.2.8 Oversampling	147
6.16 USART_MemoryMapTypeDef Struct Reference	147
6.16.1 Detailed Description	148
6.16.2 Field Documentation	148
6.16.2.1 SR_REG	148
6.16.2.2 DR_REG	148
6.16.2.3 BRR_REG	148
6.16.2.4 CR1_REG	149

6.16.2.5 CR2_REG	149
6.16.2.6 CR3_REG	149
6.16.2.7 GTPR_REG	149
6.17 UserInfo_t Struct Reference	149
6.17.1 Detailed Description	150
6.17.2 Field Documentation	150
6.17.2.1 strUsername	150
6.17.2.2 strPassword	150
7 File Documentation	151
7.1 D:/GProject/ADAS_Project/README.md File Reference	151
7.2 COTS/HAL/Bluetooth/Bluetooth_config.c File Reference	151
7.2.1 Variable Documentation	151
7.2.1.1 Bluetooth_Config	151
7.2.1.2 Bluetooth_CLK	152
7.2.1.3 Bluetooth_UART_ID	152
7.3 Bluetooth_config.c	152
7.4 COTS/HAL/Bluetooth/Bluetooth_config.h File Reference	153
7.5 Bluetooth_config.h	153
7.6 COTS/HAL/Bluetooth/Bluetooth_interface.h File Reference	153
7.6.1 Function Documentation	153
7.6.1.1 HBluetooth_vInit()	153
7.6.1.2 HBluetooth_vSendByte()	154
7.6.1.3 HBluetooth_u8ReceiveByte()	154
7.6.1.4 HBluetooth_vSendString()	154
7.6.1.5 HBluetooth_vReceiveString()	154
7.6.1.6 HBluetooth_u8CompStrings()	154
7.6.1.7 HBluetooth_vEnable()	155
7.6.1.8 HBluetooth_vDisable()	155
7.6.1.9 HBluetooth_ExchangeString()	155
7.6.1.10 HBluetooth_ExchangeByte()	155
7.7 Bluetooth_interface.h	156
7.8 COTS/HAL/Bluetooth/Bluetooth_private.h File Reference	156
7.9 Bluetooth_private.h	156
7.10 COTS/HAL/Bluetooth/Bluetooth_program.c File Reference	156
7.10.1 Function Documentation	157
7.10.1.1 HBluetooth_vInit()	157
7.10.1.2 HBluetooth_vSendByte()	157
7.10.1.3 HBluetooth_u8ReceiveByte()	158
7.10.1.4 HBluetooth_vSendString()	158
7.10.1.5 HBluetooth_vReceiveString()	158
7.10.1.6 HBluetooth_u8CompStrings()	159

7.10.1.7 HBluetooth_vEnable()	159
7.10.1.8 HBluetooth_vDisable()	159
7.10.1.9 HBluetooth_ExchangeString()	159
7.10.1.10 HBluetooth_ExchangeByte()	160
7.10.2 Variable Documentation	160
7.10.2.1 Bluetooth_Config	160
7.10.2.2 Bluetooth_CLK	160
7.10.2.3 Bluetooth_UART_ID	160
7.11 Bluetooth_program.c	161
7.12 COTS/HAL/CarControl/CarControl_config.h File Reference	163
7.13 CarControl_config.h	163
7.14 COTS/HAL/CarControl/CarControl_interface.h File Reference	163
7.14.1 Function Documentation	163
7.14.1.1 HCarControl_vInitCar()	163
7.14.1.2 HCarControl_vMoveForward()	163
7.14.1.3 HCarControl_vMoveBackward()	164
7.14.1.4 HCarControl_vTurnRight()	164
7.14.1.5 HCarControl_vTurnLeft()	164
7.14.1.6 HCarControl_vStopCar()	164
7.14.2 Variable Documentation	165
7.14.2.1 dcmMotor1	165
7.14.2.2 dcmMotor2	165
7.14.2.3 dcmMotor3	165
7.14.2.4 dcmMotor4	166
7.15 CarControl_interface.h	165
7.16 COTS/HAL/CarControl/CarControl_private.h File Reference	165
7.17 CarControl_private.h	165
7.18 COTS/HAL/CarControl/CarControl_program.c File Reference	165
7.18.1 Function Documentation	166
7.18.1.1 HCarControl_vInitCar()	166
7.18.1.2 HCarControl_vMoveForward()	166
7.18.1.3 HCarControl_vMoveBackward()	166
7.18.1.4 HCarControl_vTurnRight()	167
7.18.1.5 HCarControl_vTurnLeft()	167
7.18.1.6 HCarControl_vStopCar()	167
7.18.2 Variable Documentation	167
7.18.2.1 dcmMotor1	168
7.18.2.2 dcmMotor2	168
7.18.2.3 dcmMotor3	168
7.18.2.4 dcmMotor4	169
7.19 CarControl_program.c	169
7.20 COTS/HAL/DCMOTOR/DCM_config.h File Reference	170
7.20.1 Detailed Description	170
7.21 DCM_config.h	171
7.22 COTS/HAL/DCMOTOR/DCM_interface.h File Reference	171
7.22.1 Detailed Description	172
7.22.2 Function Documentation	172

7.22.2.1 HDCM_vInitMotor()	172
7.22.2.2 HDCM_vMoveForward()	172
7.22.2.3 HDCM_vMoveBackward()	173
7.22.2.4 HDCM_vStopMotor()	173
7.22.2.5 HDCM_vMotorSpeedCntrl()	173
7.23 DCM_interface.h	173
7.24 COTS/HAL/DCMOTOR/DCM_private.h File Reference	174
7.24.1 Detailed Description	174
7.25 DCM_private.h	175
7.26 COTS/HAL/DCMOTOR/DCM_program.c File Reference	175
7.26.1 Detailed Description	175
7.26.2 Function Documentation	176
7.26.2.1 HDCM_vInitMotor()	176
7.26.2.2 HDCM_vMoveForward()	176
7.26.2.3 HDCM_vMoveBackward()	177
7.26.2.4 HDCM_vStopMotor()	177
7.27 DCM_program.c	177
7.28 COTS/HAL/LCD/LCD_config.h File Reference	178
7.28.1 Macro Definition Documentation	179
7.28.1.1 HLCD_CTRL_PORT	179
7.28.1.2 HLCD_RS_PIN	179
7.28.1.3 HLCD_RW_PIN	180
7.28.1.4 HLCD_EN_PIN	180
7.28.1.5 HLCD_DATA_PORT	180
7.28.1.6 D0_PIN	180
7.28.1.7 D1_PIN	180
7.28.1.8 D2_PIN	180
7.28.1.9 D3_PIN	181
7.28.1.10 D4_PIN	181
7.28.1.11 D5_PIN	181
7.28.1.12 D6_PIN	181
7.28.1.13 D7_PIN	181
7.28.1.14 HLCD_FuctionSet_Cmd	181
7.28.1.15 HLCD_DisponOffCTRL_Cmd	182
7.28.1.16 HLCD_DispoFF_Cmd	182
7.28.1.17 HLCD_DispcursorWithBlinking_Cmd	182
7.28.1.18 HLCD_SetcursorBlinkingOFF_Cmd	182
7.28.1.19 HLCD_Dispclear_Cmd	182
7.28.1.20 HLCD_EntryModeSet_Cmd	182
7.28.1.21 HLCD_EntryModeSet_ShiftLeftOn_Cmd	183
7.28.1.22 HLCD_CharactersNums	183
7.28.1.23 FIRST_ROW_IDX	183

7.28.1.24 FIRST_ROW_START	183
7.28.1.25 SEC_ROW_START	183
7.28.1.26 SET_DDRAM_AC_MASK	183
7.28.1.27 SET_CGRAM_AC_MASK	184
7.28.1.28 NumOf_CGRAM_Patterns	184
7.28.1.29 FirstByteInCGRAM_Pattern	184
7.28.1.30 LastByteInCGRAM_Pattern	184
7.29 LCD_config.h	184
7.30 COTS/HAL/LCD/LCD_Interface.h File Reference	185
7.30.1 Macro Definition Documentation	186
7.30.1.1 MIN_IDX_OF_ROWS	186
7.30.1.2 MAX_IDX_OF_ROWS	186
7.30.1.3 MIN_IDX_OF_COL	187
7.30.1.4 MAX_IDX_OF_COL	187
7.30.1.5 LCD_LINE1	187
7.30.1.6 LCD_LINE2	187
7.30.1.7 LCD_Square1	187
7.30.1.8 LCD_Square2	187
7.30.1.9 LCD_Square3	188
7.30.1.10 LCD_Square4	188
7.30.1.11 LCD_Square5	188
7.30.1.12 LCD_Square6	188
7.30.1.13 LCD_Square7	188
7.30.1.14 LCD_Square8	188
7.30.1.15 LCD_Square9	189
7.30.1.16 LCD_Square10	189
7.30.1.17 LCD_Square11	189
7.30.1.18 LCD_Square12	189
7.30.1.19 LCD_Square13	189
7.30.1.20 LCD_Square14	189
7.30.1.21 LCD_Square15	190
7.30.1.22 LCD_Square16	190
7.30.1.23 CGRAM_AddressOfPattern0	190
7.30.1.24 CGRAM_AddressOfPattern1	190
7.30.1.25 CGRAM_AddressOfPattern2	190
7.30.1.26 CGRAM_AddressOfPattern3	190
7.30.1.27 CGRAM_AddressOfPattern4	191
7.30.1.28 CGRAM_AddressOfPattern5	191
7.30.1.29 CGRAM_AddressOfPattern7	191
7.30.1.30 CGRAM_AddressOfPattern8	191
7.30.2 Function Documentation	191
7.30.2.1 LCD_vInit()	192

7.30.2.2 LCD_vSendCommand()	193
7.30.2.3 LCD_vSendData()	194
7.30.2.4 LCD_vClear()	194
7.30.2.5 LCD_vDispString()	194
7.30.2.6 LCD_vDispNumber()	194
7.30.2.7 LCD_vSaveCustomChar()	194
7.30.2.8 LCD_vSaveDispChar()	195
7.30.2.9 LCD_vGoTo()	195
7.30.2.10 LCD_vSetShiftLeftOn()	195
7.30.2.11 LCD_vSetDispOFF()	195
7.30.2.12 LCD_vDispCursorWithBlinking()	196
7.30.2.13 LCD_vSetCursorBlinkingOFF()	196
7.30.2.14 LCD_vDispShiftLeftString()	196
7.30.2.15 LCD_vClearChar()	196
7.31 LCD_interface.h	197
7.32 COTS/HAL/LCD/LCD_private.h File Reference	198
7.32.1 Macro Definition Documentation	198
7.32.1.1 FuctionSetCmd	199
7.32.1.2 DispOnOffCTRLCmd	199
7.32.1.3 DispCursorWithBlinkingCmd	199
7.32.1.4 SetCursorBlinkingOFFCmd	199
7.32.1.5 DispOffCmd	199
7.32.1.6 DispClearCmd	199
7.32.1.7 EntryModeSet_Cmd	200
7.32.1.8 EntryModeSet_ShiftLeftOn_Cmd	200
7.33 LCD_private.h	200
7.34 COTS/HAL/LCD/LCD_program.c File Reference	201
7.34.1 Function Documentation	201
7.34.1.1 LCD_vInit()	201
7.34.1.2 LCD_vSendCommand()	203
7.34.1.3 LCD_vSendData()	204
7.34.1.4 LCD_vClear()	204
7.34.1.5 LCD_vDispString()	205
7.34.1.6 LCD_vDispNumber()	205
7.34.1.7 LCD_vSaveCustomChar()	206
7.34.1.8 LCD_vGoTo()	206
7.34.1.9 LCD_vSetShiftLeftOn()	207
7.34.1.10 LCD_vSetDispOFF()	207
7.34.1.11 LCD_vDispCursorWithBlinking()	207
7.34.1.12 LCD_vSetCursorBlinkingOFF()	208
7.34.1.13 LCD_vDispShiftLeftString()	208
7.34.1.14 LCD_vClearChar()	208

7.35 LCD_program.c	209
7.36 COTS/HAL/LDR/LDR_config.c File Reference	215
7.36.1 Variable Documentation	215
7.36.1.1 LDR_Vo	215
7.36.1.2 LDR_CHANNEL	216
7.37 LDR_config.c	216
7.38 COTS/HAL/LDR/LDR_config.h File Reference	216
7.38.1 Macro Definition Documentation	217
7.38.1.1 LDR_Vo_PORT	217
7.38.1.2 LDR_Vo_PIN	217
7.39 LDR_config.h	217
7.40 COTS/HAL/LDR/LDR_interface.h File Reference	217
7.40.1 Macro Definition Documentation	218
7.40.1.1 DARK	218
7.40.1.2 VERY_DIM	218
7.40.1.3 MODERATE	218
7.40.1.4 OVERCAST	218
7.40.1.5 VERY_BRIGHT	218
7.40.2 Function Documentation	218
7.40.2.1 HLDR_vInit()	219
7.40.2.2 HLDR_u16DigitalOutputValue()	219
7.41 LDR_interface.h	219
7.42 COTS/HAL/LDR/LDR_private.h File Reference	220
7.43 LDR_private.h	220
7.44 COTS/HAL/LDR/LDR_program.c File Reference	220
7.44.1 Function Documentation	220
7.44.1.1 HLDR_vInit()	221
7.44.1.2 HLDR_u16DigitalOutputValue()	221
7.44.2 Variable Documentation	221
7.44.2.1 LDR_Vo	221
7.44.2.2 LDR_CHANNEL	221
7.45 LDR_program.c	222
7.46 COTS/LIB/LSTD_BITMATH.h File Reference	222
7.46.1 Detailed Description	223
7.47 LSTD_BITMATH.h	223
7.48 COTS/LIB/LSTD_COMPILER.h File Reference	223
7.48.1 Detailed Description	224
7.49 LSTD_COMPILER.h	224
7.50 COTS/LIB/LSTD MCU UTILITIES.h File Reference	225
7.50.1 Detailed Description	225
7.51 LSTD MCU UTILITIES.h	225
7.52 COTS/LIB/LSTD TYPES.h File Reference	225

7.52.1 Detailed Description	226
7.53 LSTD_TYPES.h	226
7.54 COTS/LIB/LSTD_VALUES.h File Reference	226
7.54.1 Detailed Description	227
7.55 LSTD_VALUES.h	227
7.56 COTS/MCAL/ADC/ADC_config.h File Reference	228
7.56.1 Macro Definition Documentation	229
7.56.1.1 OVERRUN_INT	229
7.56.1.2 RESOLUTION	229
7.56.1.3 ANALOG_WDT	230
7.56.1.4 INJ_ANALOG_WDT	230
7.56.1.5 DISC_CHANNEL_NUM	230
7.56.1.6 INJ_DISC_MODE	230
7.56.1.7 DISC_MODE	230
7.56.1.8 INJ_AUTO_MODE	230
7.56.1.9 AWDSGL_MODE	231
7.56.1.10 SCAN_MODE	231
7.56.1.11 JEOCIE_MODE	231
7.56.1.12 AWDIE_MODE	231
7.56.1.13 EOCIE_MODE	231
7.56.1.14 AWD_CHANNEL	231
7.56.1.15 REG_START_CONV	232
7.56.1.16 REG_EXTERNAL_TRIGGER	232
7.56.1.17 REG_EXT_EVENT_SRC	232
7.56.1.18 INJ_START_CONV	232
7.56.1.19 INJ_EXTERNAL_TRIGGER	232
7.56.1.20 INJ_EXT_EVENT_SRC	232
7.56.1.21 DATA_ALIGN	233
7.56.1.22 EOC_SELECTION	233
7.56.1.23 DMA_MODE	233
7.56.1.24 DMA_DIS_SELECTION	233
7.56.1.25 CONT_CONV	233
7.56.1.26 ADC_ON	233
7.56.1.27 SMP10	234
7.56.1.28 SMP11	234
7.56.1.29 SMP12	234
7.56.1.30 SMP13	234
7.56.1.31 SMP14	234
7.56.1.32 SMP15	234
7.56.1.33 SMP16	235
7.56.1.34 SMP17	235
7.56.1.35 SMP18	235

7.56.1.36 SMP0	235
7.56.1.37 SMP1	235
7.56.1.38 SMP2	235
7.56.1.39 SMP3	236
7.56.1.40 SMP4	236
7.56.1.41 SMP5	236
7.56.1.42 SMP6	236
7.56.1.43 SMP7	236
7.56.1.44 SMP8	236
7.56.1.45 SMP9	237
7.56.1.46 REG_SQ_LENGTH	237
7.56.1.47 TEMP_SENSOR_AND_VREF	237
7.56.1.48 V_BATTERY	237
7.56.1.49 ADC_PRESCALER	237
7.56.1.50 SQ1_BIT_MANIPULATION	237
7.57 ADC_config.h	238
7.58 COTS/MCAL/ADC/ADC_interface.h File Reference	245
7.58.1 Macro Definition Documentation	245
7.58.1.1 CHANNEL0	246
7.58.1.2 CHANNEL1	246
7.58.1.3 CHANNEL2	246
7.58.1.4 CHANNEL3	246
7.58.1.5 CHANNEL4	246
7.58.1.6 CHANNEL5	246
7.58.1.7 CHANNEL6	247
7.58.1.8 CHANNEL7	247
7.58.1.9 CHANNEL8	247
7.58.1.10 CHANNEL9	247
7.58.1.11 CHANNEL10	247
7.58.1.12 CHANNEL11	247
7.58.1.13 CHANNEL12	248
7.58.1.14 CHANNEL13	248
7.58.1.15 CHANNEL14	248
7.58.1.16 CHANNEL15	248
7.58.1.17 CHANNEL16	248
7.58.1.18 CHANNEL17	248
7.58.1.19 CHANNEL18	249
7.58.2 Function Documentation	249
7.58.2.1 MADC_vInit()	249
7.58.2.2 MADC_u16ConvertToDigital()	253
7.58.2.3 MADC_vDisable()	253
7.58.2.4 MADC_vEnable()	253

7.58.2.5 MADC_vRegINT_Disable()	253
7.58.2.6 MADC_vRegINTEnable()	254
7.58.2.7 MADC_vSelectMode()	254
7.58.2.8 MADC_vSelectChannel()	254
7.58.2.9 MADC_vStartConversion()	254
7.58.2.10 MADC_vSelectChannelsOrder()	254
7.58.2.11 MADC_vSelectSampleTime()	254
7.58.2.12 MADC_vSelectResolution()	255
7.58.2.13 MADC_u16GetADCData()	255
7.58.2.14 MADC_vInitStruct()	255
7.58.2.15 MADC_vSetCallBack()	255
7.59 ADC_interface.h	255
7.60 COTS/MCAL/ADC/ADC_private.h File Reference	257
7.60.1 Macro Definition Documentation	262
7.60.1.1 ADC_BASE_ADDRESS	262
7.60.1.2 ADC	262
7.60.1.3 OVR	263
7.60.1.4 STRT	263
7.60.1.5 JSTRT	263
7.60.1.6 JEOC	263
7.60.1.7 EOC	263
7.60.1.8 AWD	263
7.60.1.9 OVRIE	264
7.60.1.10 RES0	264
7.60.1.11 RES1	264
7.60.1.12 AWDEN	264
7.60.1.13 JAWDEN	264
7.60.1.14 DISCNUM0	264
7.60.1.15 DISCNUM1	265
7.60.1.16 DISCNUM2	265
7.60.1.17 JDISCEN	265
7.60.1.18 DISCEN	265
7.60.1.19 JAUTO	265
7.60.1.20 AWDSGL	265
7.60.1.21 SCAN	266
7.60.1.22 JEOCIE	266
7.60.1.23 AWDIE	266
7.60.1.24 EOCIE	266
7.60.1.25 AWDCH0	266
7.60.1.26 AWDCH1	266
7.60.1.27 AWDCH2	267
7.60.1.28 AWDCH3	267

7.60.1.29 AWDCH4	267
7.60.1.30 SWSTART	267
7.60.1.31 EXTENO	267
7.60.1.32 EXTEN1	267
7.60.1.33 EXTSEL0	268
7.60.1.34 EXTSEL1	268
7.60.1.35 EXTSEL2	268
7.60.1.36 EXTSEL3	268
7.60.1.37 JSWSTART	268
7.60.1.38 JEXTENO	268
7.60.1.39 JEXTEN1	269
7.60.1.40 JEXTSEL0	269
7.60.1.41 JEXTSEL1	269
7.60.1.42 JEXTSEL2	269
7.60.1.43 JEXTSEL3	269
7.60.1.44 ALIGN	269
7.60.1.45 EOCS	270
7.60.1.46 DDS	270
7.60.1.47 DMA	270
7.60.1.48 CONT	270
7.60.1.49 ADON	270
7.60.1.50 SMP10_b0	270
7.60.1.51 SMP10_b1	271
7.60.1.52 SMP10_b2	271
7.60.1.53 SMP11_b0	271
7.60.1.54 SMP11_b1	271
7.60.1.55 SMP11_b2	271
7.60.1.56 SMP12_b0	271
7.60.1.57 SMP12_b1	272
7.60.1.58 SMP12_b2	272
7.60.1.59 SMP13_b0	272
7.60.1.60 SMP13_b1	272
7.60.1.61 SMP13_b2	272
7.60.1.62 SMP14_b0	272
7.60.1.63 SMP14_b1	273
7.60.1.64 SMP14_b2	273
7.60.1.65 SMP15_b0 [1/2]	273
7.60.1.66 SMP15_b1 [1/2]	273
7.60.1.67 SMP15_b2 [1/2]	273
7.60.1.68 SMP15_b0 [2/2]	273
7.60.1.69 SMP15_b1 [2/2]	274
7.60.1.70 SMP15_b2 [2/2]	274

7.60.1.71 SMP16_b0	274
7.60.1.72 SMP16_b1	274
7.60.1.73 SMP16_b2	274
7.60.1.74 SMP17_b0	274
7.60.1.75 SMP17_b1	275
7.60.1.76 SMP17_b2	275
7.60.1.77 SMP18_b0	275
7.60.1.78 SMP18_b1	275
7.60.1.79 SMP18_b2	275
7.60.1.80 SMP0_b0	275
7.60.1.81 SMP0_b1	276
7.60.1.82 SMP0_b2	276
7.60.1.83 SMP1_b0	276
7.60.1.84 SMP1_b1	276
7.60.1.85 SMP1_b2	276
7.60.1.86 SMP2_b0	276
7.60.1.87 SMP2_b1	277
7.60.1.88 SMP2_b2	277
7.60.1.89 SMP3_b0	277
7.60.1.90 SMP3_b1	277
7.60.1.91 SMP3_b2	277
7.60.1.92 SMP4_b0	277
7.60.1.93 SMP4_b1	278
7.60.1.94 SMP4_b2	278
7.60.1.95 SMP5_b0 [1/2]	278
7.60.1.96 SMP5_b1 [1/2]	278
7.60.1.97 SMP5_b2 [1/2]	278
7.60.1.98 SMP5_b0 [2/2]	278
7.60.1.99 SMP5_b1 [2/2]	279
7.60.1.100 SMP5_b2 [2/2]	279
7.60.1.101 SMP6_b0	279
7.60.1.102 SMP6_b1	279
7.60.1.103 SMP6_b2	279
7.60.1.104 SMP7_b0	279
7.60.1.105 SMP7_b1	280
7.60.1.106 SMP7_b2	280
7.60.1.107 SMP8_b0	280
7.60.1.108 SMP8_b1	280
7.60.1.109 SMP8_b2	280
7.60.1.110 SMP9_b0	280
7.60.1.111 SMP9_b1	281
7.60.1.112 SMP9_b2	281

7.60.1.113 L0	281
7.60.1.114 L1	281
7.60.1.115 L2	281
7.60.1.116 L3	281
7.60.1.117 SQ16_b0	282
7.60.1.118 SQ16_b1	282
7.60.1.119 SQ16_b2	282
7.60.1.120 SQ16_b3	282
7.60.1.121 SQ16_b4	282
7.60.1.122 SQ15_b0	282
7.60.1.123 SQ15_b1	283
7.60.1.124 SQ15_b2	283
7.60.1.125 SQ15_b3	283
7.60.1.126 SQ15_b4	283
7.60.1.127 SQ14_b0	283
7.60.1.128 SQ14_b1	283
7.60.1.129 SQ14_b2	284
7.60.1.130 SQ14_b3	284
7.60.1.131 SQ14_b4	284
7.60.1.132 SQ13_b0	284
7.60.1.133 SQ13_b1	284
7.60.1.134 SQ13_b2	284
7.60.1.135 SQ13_b3	285
7.60.1.136 SQ13_b4	285
7.60.1.137 SQ12_b0	285
7.60.1.138 SQ12_b1	285
7.60.1.139 SQ12_b2	285
7.60.1.140 SQ12_b3	285
7.60.1.141 SQ12_b4	286
7.60.1.142 SQ11_b0	286
7.60.1.143 SQ11_b1	286
7.60.1.144 SQ11_b2	286
7.60.1.145 SQ11_b3	286
7.60.1.146 SQ11_b4	286
7.60.1.147 SQ10_b0	287
7.60.1.148 SQ10_b1	287
7.60.1.149 SQ10_b2	287
7.60.1.150 SQ10_b3	287
7.60.1.151 SQ10_b4	287
7.60.1.152 SQ9_b0	287
7.60.1.153 SQ9_b1	288
7.60.1.154 SQ9_b2	288

7.60.1.155 SQ9_b3	288
7.60.1.156 SQ9_b4	288
7.60.1.157 SQ8_b0	288
7.60.1.158 SQ8_b1	288
7.60.1.159 SQ8_b2	289
7.60.1.160 SQ8_b3	289
7.60.1.161 SQ8_b4	289
7.60.1.162 SQ7_b0	289
7.60.1.163 SQ7_b1	289
7.60.1.164 SQ7_b2	289
7.60.1.165 SQ7_b3	290
7.60.1.166 SQ7_b4	290
7.60.1.167 SQ6_b0	290
7.60.1.168 SQ6_b1	290
7.60.1.169 SQ6_b2	290
7.60.1.170 SQ6_b3	290
7.60.1.171 SQ6_b4	291
7.60.1.172 SQ5_b0	291
7.60.1.173 SQ5_b1	291
7.60.1.174 SQ5_b2	291
7.60.1.175 SQ5_b3	291
7.60.1.176 SQ5_b4	291
7.60.1.177 SQ4_b0	292
7.60.1.178 SQ4_b1	292
7.60.1.179 SQ4_b2	292
7.60.1.180 SQ4_b3	292
7.60.1.181 SQ4_b4	292
7.60.1.182 SQ3_b0	292
7.60.1.183 SQ3_b1	293
7.60.1.184 SQ3_b2	293
7.60.1.185 SQ3_b3	293
7.60.1.186 SQ3_b4	293
7.60.1.187 SQ2_b0	293
7.60.1.188 SQ2_b1	293
7.60.1.189 SQ2_b2	294
7.60.1.190 SQ2_b3	294
7.60.1.191 SQ2_b4	294
7.60.1.192 SQ1_b0	294
7.60.1.193 SQ1_b1	294
7.60.1.194 SQ1_b2	294
7.60.1.195 SQ1_b3	295
7.60.1.196 SQ1_b4	295

7.60.1.197 TSVREFE	295
7.60.1.198 VBATE	295
7.60.1.199 ADCPRE0	295
7.60.1.200 ADCPRE1	295
7.60.1.201 ENABLE	296
7.60.1.202 DISABLE	296
7.60.1.203 _12_BITS	296
7.60.1.204 _10_BITS	296
7.60.1.205 _8_BITS	296
7.60.1.206 _6_BITS	296
7.60.1.207 _1_CHANNEL	297
7.60.1.208 _2_CHANNELS	297
7.60.1.209 _3_CHANNELS	297
7.60.1.210 _4_CHANNELS	297
7.60.1.211 _5_CHANNELS	297
7.60.1.212 _6_CHANNELS	297
7.60.1.213 _7_CHANNELS	298
7.60.1.214 _8_CHANNELS	298
7.60.1.215 ZERO	298
7.60.1.216 ONE	298
7.60.1.217 THREE	298
7.60.1.218 FOUR	298
7.60.1.219 FIVE	299
7.60.1.220 SIX	299
7.60.1.221 SEVEN	299
7.60.1.222 EIGHT	299
7.60.1.223 NINE	299
7.60.1.224 TEN	299
7.60.1.225 ELEVEN	300
7.60.1.226 TWELVE	300
7.60.1.227 THIRTEEN	300
7.60.1.228 FOURTEEN	300
7.60.1.229 FIFTEEN	300
7.60.1.230 SIXTEEN	300
7.60.1.231 SEVENTEEN	301
7.60.1.232 EIGHTEEN	301
7.60.1.233 T1_CC1	301
7.60.1.234 T1_CC2	301
7.60.1.235 T1_CC3	301
7.60.1.236 T2_CC2	301
7.60.1.237 T2_CC3	302
7.60.1.238 T2_CC4	302

7.60.1.239 REG_T2_TRGO	302
7.60.1.240 T3_CC1	302
7.60.1.241 T3_TRGO	302
7.60.1.242 T4_CC4	302
7.60.1.243 T5_CC1	303
7.60.1.244 T5_CC2	303
7.60.1.245 T5_CC3	303
7.60.1.246 EXTI_LINE11	303
7.60.1.247 RISING	303
7.60.1.248 FALLING	303
7.60.1.249 ON_CHANGE	304
7.60.1.250 T1_CC4	304
7.60.1.251 T1_TRGO	304
7.60.1.252 T2_CC1	304
7.60.1.253 INJ_T2_TRGO	304
7.60.1.254 T3_CC2	304
7.60.1.255 T3_CC4	305
7.60.1.256 T4_CC1	305
7.60.1.257 T4_CC2	305
7.60.1.258 T4_CC3	305
7.60.1.259 T4_TRGO	305
7.60.1.260 T5_CC4	305
7.60.1.261 T5_TRGO	306
7.60.1.262 EXTI_LINE15	306
7.60.1.263 RIGHT	306
7.60.1.264 LEFT	306
7.60.1.265 _3_CYCLE	306
7.60.1.266 _15_CYCLE	306
7.60.1.267 _28_CYCLE	307
7.60.1.268 _56_CYCLE	307
7.60.1.269 _84_CYCLE	307
7.60.1.270 _112_CYCLE	307
7.60.1.271 _144_CYCLE	307
7.60.1.272 _480_CYCLE	307
7.60.1.273 _1_CONV	308
7.60.1.274 _2_CONV	308
7.60.1.275 _3_CONV	308
7.60.1.276 _4_CONV	308
7.60.1.277 _5_CONV	308
7.60.1.278 _6_CONV	308
7.60.1.279 _7_CONV	309
7.60.1.280 _8_CONV	309

7.60.1.281 _9_CONV	309
7.60.1.282 _10_CONV	309
7.60.1.283 _11_CONV	309
7.60.1.284 _15_CONV	309
7.60.1.285 _16_CONV	310
7.60.1.286 DIV_BY_2	310
7.60.1.287 DIV_BY_4	310
7.60.1.288 DIV_BY_6	310
7.60.1.289 DIV_BY_8	310
7.61 ADC_private.h	311
7.62 COTS/MCAL/ADC/ADC_program.c File Reference	316
7.62.1 Function Documentation	316
7.62.1.1 MADC_vInit()	317
7.62.1.2 MADC_vSelectChannel()	320
7.62.1.3 MADC_u16ConvertToDigital()	321
7.62.1.4 MADC_vDisable()	321
7.62.1.5 MADC_vEnable()	321
7.62.1.6 MADC_vRegINT_Disable()	322
7.62.1.7 MADC_vRegINTEnable()	322
7.62.1.8 MADC_vSetCallBack()	322
7.62.1.9 ADC_IRQHandler()	322
7.62.2 Variable Documentation	323
7.62.2.1 MADC_ypPointerToFunction	323
7.63 ADC_program.c	323
7.64 COTS/MCAL/EXTI/EXTI_config.h File Reference	328
7.64.1 Macro Definition Documentation	329
7.64.1.1 EXTI_LINE0_EN	329
7.64.1.2 EXTI_LINE0_TRIGGER	329
7.64.1.3 EXTI_LINE1_EN	330
7.64.1.4 EXTI_LINE1_TRIGGER	330
7.64.1.5 EXTI_LINE2_EN	330
7.64.1.6 EXTI_LINE2_TRIGGER	330
7.64.1.7 EXTI_LINE3_EN	330
7.64.1.8 EXTI_LINE3_TRIGGER	330
7.64.1.9 EXTI_LINE4_EN	331
7.64.1.10 EXTI_LINE4_TRIGGER	331
7.64.1.11 EXTI_LINE5_EN	331
7.64.1.12 EXTI_LINE5_TRIGGER	331
7.64.1.13 EXTI_LINE6_EN	331
7.64.1.14 EXTI_LINE6_TRIGGER	331
7.64.1.15 EXTI_LINE7_EN	332
7.64.1.16 EXTI_LINE7_TRIGGER	332

7.64.1.17 EXTI_LINE8_EN	332
7.64.1.18 EXTI_LINE8_TRIGGER	332
7.64.1.19 EXTI_LINE9_EN	332
7.64.1.20 EXTI_LINE9_TRIGGER	332
7.64.1.21 EXTI_LINE10_EN	333
7.64.1.22 EXTI_LINE10_TRIGGER	333
7.64.1.23 EXTI_LINE11_EN	333
7.64.1.24 EXTI_LINE11_TRIGGER	333
7.64.1.25 EXTI_LINE12_EN	333
7.64.1.26 EXTI_LINE12_TRIGGER	333
7.64.1.27 EXTI_LINE13_EN	334
7.64.1.28 EXTI_LINE13_TRIGGER	334
7.64.1.29 EXTI_LINE14_EN	334
7.64.1.30 EXTI_LINE14_TRIGGER	334
7.64.1.31 EXTI_LINE15_EN	334
7.64.1.32 EXTI_LINE15_TRIGGER	334
7.65 EXTI_config.h	335
7.66 COTS/MCAL/EXTI/EXTI_interface.h File Reference	337
7.66.1 Detailed Description	339
7.66.2 Function Documentation	339
7.66.2.1 MEXTI_vInit()	339
7.66.2.2 MEXTI_vInit_WithStruct()	342
7.66.2.3 MEXTI_vEnableLine()	342
7.66.2.4 MEXTI_vDisableLine()	342
7.66.2.5 MEXTI_vSWITrigger()	343
7.66.2.6 MEXTI_vSetTrigger()	343
7.66.2.7 MEXTI_vSetCallback()	343
7.66.2.8 MSYSCFG_vSetEXTIPort()	343
7.67 EXTI_interface.h	344
7.68 COTS/MCAL/EXTI/EXTI_private.h File Reference	345
7.68.1 Detailed Description	345
7.68.2 Macro Definition Documentation	345
7.68.2.1 EXTI IRQs	345
7.69 EXTI_private.h	346
7.70 COTS/MCAL/EXTI/EXTI_program.c File Reference	346
7.70.1 Detailed Description	347
7.70.2 Function Documentation	347
7.70.2.1 MSYSCFG_vSetEXTIPort()	347
7.70.2.2 MEXTI_vInit()	348
7.70.2.3 MEXTI_vInit_WithStruct()	350
7.70.2.4 MEXTI_vEnableLine()	351
7.70.2.5 MEXTI_vDisableLine()	351

7.70.2.6 MEXTI_vSWITrigger()	351
7.70.2.7 MEXTI_vSetTrigger()	352
7.70.2.8 MEXTI_vSetCallback()	352
7.70.2.9 EXTI0_IRQHandler()	353
7.70.2.10 EXTI1_IRQHandler()	353
7.70.2.11 EXTI2_IRQHandler()	353
7.70.2.12 EXTI3_IRQHandler()	354
7.70.2.13 EXTI4_IRQHandler()	354
7.70.2.14 EXTI9_5_IRQHandler()	354
7.70.2.15 EXTI15_10_IRQHandler()	354
7.71 EXTI_program.c	355
7.72 COTS/MCAL/EXTI/SYSCFG_private.h File Reference	360
7.72.1 Detailed Description	361
7.73 SYSCFG_private.h	361
7.74 COTS/MCAL/GPIO/GPIO_config.h File Reference	361
7.74.1 Detailed Description	362
7.74.2 Macro Definition Documentation	362
7.74.2.1 GPIOA_PIN_POS	362
7.74.2.2 GPIOB_PIN_POS	362
7.74.2.3 LCKK_BIT_POS	363
7.74.2.4 PORTA_BIT_MANIPULATION	363
7.74.2.5 PORTB_BIT_MANIPULATION	363
7.75 GPIO_config.h	363
7.76 COTS/MCAL/GPIO/GPIO_interface.h File Reference	363
7.76.1 Detailed Description	366
7.76.2 Macro Definition Documentation	367
7.76.2.1 GPIOA_LOW_NIBBLE_HIGH	367
7.76.2.2 GPIOA_LOW_NIBBLE_LOW	367
7.76.3 Function Documentation	367
7.76.3.1 MGPIOp_vLockedPins()	367
7.76.3.2 MGPIOp_vSetPinMode()	368
7.76.3.3 MGPIOp_vSetPinOutputType()	368
7.76.3.4 MGPIOp_vSetPinOutputSpeed()	369
7.76.3.5 MGPIOp_vSetPinInputPullType()	369
7.76.3.6 MGPIOp_u8GetPinValue()	369
7.76.3.7 MGPIOp_vSetPinValue()	370
7.76.3.8 MGPIOp_vSetResetAtomic()	370
7.76.3.9 MGPIOp_vSetAlternateFunctionON()	370
7.76.3.10 MGPIOp_vSetPortConfigLock()	371
7.76.3.11 MGPIOp_vInit()	371
7.76.3.12 MGPIOp_vTogglePinValue()	371
7.76.3.13 GPIO_vSetPortValue()	372

7.77 GPIO_interface.h	372
7.78 COTS/MCAL.GPIO.GPIO_private.h File Reference	374
7.78.1 Detailed Description	374
7.79 GPIO_private.h	375
7.80 COTS/MCAL.GPIO.GPIO_program.c File Reference	375
7.80.1 Detailed Description	376
7.80.2 Function Documentation	376
7.80.2.1 MGPIOX_vLockedPins()	376
7.80.2.2 MGPIOX_vSetPinMode()	377
7.80.2.3 MGPIOX_vSetPinOutputType()	378
7.80.2.4 MGPIOX_vSetPinOutputSpeed()	378
7.80.2.5 MGPIOX_vSetPinInputPullType()	379
7.80.2.6 MGPIOX_u8GetPinValue()	379
7.80.2.7 MGPIOX_vSetPinValue()	380
7.80.2.8 MGPIOX_vSetResetAtomic()	381
7.80.2.9 MGPIOX_vSetAlternateFunctionON()	381
7.80.2.10 MGPIOX_vInit()	383
7.80.2.11 MGPIOX_vTogglePinValue()	383
7.80.2.12 GPIO_vSetPortValue()	384
7.81 GPIO_program.c	384
7.82 COTS/MCAL/NVIC/NVIC_config.h File Reference	390
7.83 NVIC_config.h	390
7.84 COTS/MCAL/NVIC/NVIC_interface.h File Reference	390
7.84.1 Detailed Description	394
7.84.2 Function Documentation	395
7.84.2.1 MNVIC_vEnablePeriphral()	395
7.84.2.2 MNVIC_vDisablePeriphral()	395
7.84.2.3 MNVIC_vSetPendingFlag()	396
7.84.2.4 MNVIC_vClearPendingFlag()	396
7.84.2.5 MNVIC_u8GetActive()	396
7.84.2.6 MNVIC_vSetPriorityConfig()	397
7.84.2.7 MNVIC_vSetPriority()	397
7.84.2.8 NVIC_GetPriority()	398
7.85 NVIC_interface.h	398
7.86 COTS/MCAL/NVIC/NVIC_private.h File Reference	399
7.86.1 Detailed Description	400
7.86.2 Macro Definition Documentation	400
7.86.2.1 MNVIC_STIR	401
7.86.2.2 VECTKEY_PASSWORD	401
7.87 NVIC_private.h	401
7.88 COTS/MCAL/NVIC/NVIC_program.c File Reference	402
7.88.1 Detailed Description	402

7.88.2 Macro Definition Documentation	403
7.88.2.1 REGISTER_SIZE	403
7.88.3 Function Documentation	403
7.88.3.1 MNVIC_vEnablePeriphral()	403
7.88.3.2 MNVIC_vDisablePeriphral()	403
7.88.3.3 MNVIC_vSetPendingFlag()	403
7.88.3.4 MNVIC_vClearPendingFlag()	404
7.88.3.5 MNVIC_u8GetActive()	404
7.88.3.6 MNVIC_vSetPriorityConfig()	404
7.88.3.7 MNVIC_vSetPriority()	405
7.88.3.8 NVIC_GetPriority()	405
7.89 NVIC_program.c	406
7.90 COTS/MCAL/RCC/MRCC_config.h File Reference	408
7.90.1 Detailed Description	409
7.90.2 Macro Definition Documentation	410
7.90.2.1 PLLI2S	410
7.90.2.2 PLL	410
7.90.2.3 CSS	410
7.90.2.4 HSEBYP	410
7.90.2.5 HSE_EN	410
7.90.2.6 HSI_EN	411
7.90.2.7 PLLQ	411
7.90.2.8 PLLSRC	411
7.90.2.9 PLLP	411
7.90.2.10 PLLN	411
7.90.2.11 PLLM	411
7.90.2.12 MCO2	412
7.90.2.13 MCO2PRE	412
7.90.2.14 MCO1PRE	412
7.90.2.15 I2SSRC	412
7.90.2.16 MCO1	412
7.90.2.17 RTCPRE	412
7.90.2.18 PPREG	413
7.90.2.19 PPREF	413
7.90.2.20 HPREG	413
7.90.2.21 SWS	413
7.90.2.22 SW	413
7.90.2.23 DMA2CLK	413
7.90.2.24 DMA1CLK	414
7.90.2.25 CRCCLK	414
7.90.2.26 GPIOHCLK	414
7.90.2.27 GPIOECLK	414

7.90.2.28 GPIODCLK	414
7.90.2.29 GPIOCCLK	414
7.90.2.30 GPIOBCLK	415
7.90.2.31 GPIOACLK	415
7.90.2.32 OTGFS	415
7.90.2.33 PWREN	415
7.90.2.34 I2C3EN	415
7.90.2.35 I2C2EN	415
7.90.2.36 I2C1EN	416
7.90.2.37 USART2EN	416
7.90.2.38 SPI3EN	416
7.90.2.39 SPI2EN	416
7.90.2.40 WWDGEN	416
7.90.2.41 TIM5EN	416
7.90.2.42 TIM4EN	417
7.90.2.43 TIM3EN	417
7.90.2.44 TIM2EN	417
7.90.2.45 TIM1EN	417
7.90.2.46 TIM10EN	417
7.90.2.47 TIM9EN	417
7.90.2.48 SYSCFGEN	418
7.90.2.49 SPI4EN	418
7.90.2.50 SPI1EN	418
7.90.2.51 SDIOEN	418
7.90.2.52 ADC1EN	418
7.90.2.53 USART6EN	418
7.90.2.54 USART1EN	419
7.90.2.55 TIM1EN	419
7.91 MRCC_config.h	419
7.92 COTS/MCAL/RCC/MRCC_interface.h File Reference	427
7.92.1 Detailed Description	429
7.92.2 Function Documentation	429
7.92.2.1 MRCC_vInit()	429
7.92.2.2 MRCC_vEnablePeriphralCLK()	434
7.92.2.3 MRCC_vDisablePeriphralCLK()	434
7.93 MRCC_interface.h	434
7.94 COTS/MCAL/RCC/MRCC_private.h File Reference	435
7.94.1 Detailed Description	440
7.94.2 Macro Definition Documentation	440
7.94.2.1 RCC_BASE_ADDRESS	440
7.94.2.2 RCC	441
7.94.2.3 RCC_CR_PLLI2SRDY	441

7.94.2.4 RCC_CR_PLLI2SON	441
7.94.2.5 RCC_CR_PLLRDY	441
7.94.2.6 RCC_CR_PLLON	441
7.94.2.7 RCC_CR_CSSON	441
7.94.2.8 RCC_CR_HSEBYP	442
7.94.2.9 RCC_CR_HSERDY	442
7.94.2.10 RCC_CR_HSEON	442
7.94.2.11 RCC_CR_HSIRDY	442
7.94.2.12 RCC_CR_HSION	442
7.94.2.13 RCC_PLLCFGR_PLLQ_b0	442
7.94.2.14 RCC_PLLCFGR_PLLQ_b1	443
7.94.2.15 RCC_PLLCFGR_PLLQ_b2	443
7.94.2.16 RCC_PLLCFGR_PLLQ_b3	443
7.94.2.17 RCC_PLLCFGR_PLLSRC	443
7.94.2.18 RCC_PLLCFGR_PLLP_b0	443
7.94.2.19 RCC_PLLCFGR_PLLP_b1	443
7.94.2.20 RCC_PLLCFGR_PLLN_b0	444
7.94.2.21 RCC_PLLCFGR_PLLN_b1	444
7.94.2.22 RCC_PLLCFGR_PLLN_b2	444
7.94.2.23 RCC_PLLCFGR_PLLN_b3	444
7.94.2.24 RCC_PLLCFGR_PLLN_b4	444
7.94.2.25 RCC_PLLCFGR_PLLN_b5	444
7.94.2.26 RCC_PLLCFGR_PLLN_b6	445
7.94.2.27 RCC_PLLCFGR_PLLN_b7	445
7.94.2.28 RCC_PLLCFGR_PLLN_b8	445
7.94.2.29 RCC_PLLCFGR_PLLM_b0	445
7.94.2.30 RCC_PLLCFGR_PLLM_b1	445
7.94.2.31 RCC_PLLCFGR_PLLM_b2	445
7.94.2.32 RCC_PLLCFGR_PLLM_b3	446
7.94.2.33 RCC_PLLCFGR_PLLM_b4	446
7.94.2.34 RCC_PLLCFGR_PLLM_b5	446
7.94.2.35 RCC_CFGR_MOC2_b0	446
7.94.2.36 RCC_CFGR_MOC2_b1	446
7.94.2.37 RCC_CFGR_MOC2PRE_b0	446
7.94.2.38 RCC_CFGR_MOC2PRE_b1	447
7.94.2.39 RCC_CFGR_MOC2PRE_b2	447
7.94.2.40 RCC_CFGR_MOC1PRE_b0	447
7.94.2.41 RCC_CFGR_MOC1PRE_b1	447
7.94.2.42 RCC_CFGR_MOC1PRE_b2	447
7.94.2.43 RCC_CFGR_I2SSRC	447
7.94.2.44 RCC_CFGR_MOC1_b0	448
7.94.2.45 RCC_CFGR_MOC1_b1	448

7.94.2.46 RCC_CFGR_RTCPRE_b0	448
7.94.2.47 RCC_CFGR_RTCPRE_b1	448
7.94.2.48 RCC_CFGR_RTCPRE_b2	448
7.94.2.49 RCC_CFGR_RTCPRE_b3	448
7.94.2.50 RCC_CFGR_RTCPRE_b4	449
7.94.2.51 RCC_CFGR_PPREG2_b0	449
7.94.2.52 RCC_CFGR_PPREG2_b1	449
7.94.2.53 RCC_CFGR_PPREG2_b2	449
7.94.2.54 RCC_CFGR_PPREG1_b0	449
7.94.2.55 RCC_CFGR_PPREG1_b1	449
7.94.2.56 RCC_CFGR_PPREG1_b2	450
7.94.2.57 RCC_CFGR_HPRE_b0	450
7.94.2.58 RCC_CFGR_HPRE_b1	450
7.94.2.59 RCC_CFGR_HPRE_b2	450
7.94.2.60 RCC_CFGR_HPRE_b3	450
7.94.2.61 RCC_CFGR_SWS_b0	450
7.94.2.62 RCC_CFGR_SWS_b1	451
7.94.2.63 RCC_CFGR_SW_b0	451
7.94.2.64 RCC_CFGR_SW_b1	451
7.94.2.65 RCC_AHB1ENR_DMA2EN	451
7.94.2.66 RCC_AHB1ENR_DMA1EN	451
7.94.2.67 RCC_AHB1ENR_CRCEN	451
7.94.2.68 RCC_AHB1ENR_GPIOHEN	452
7.94.2.69 RCC_AHB1ENR_GPIOEEN	452
7.94.2.70 RCC_AHB1ENR_GPIODEN	452
7.94.2.71 RCC_AHB1ENR_GPIOCEN	452
7.94.2.72 RCC_AHB1ENR_GPIOBEN	452
7.94.2.73 RCC_AHB1ENR_GPIOAEN	452
7.94.2.74 RCC_AHB2ENR_OTGFSEN	453
7.94.2.75 RCC_APB1ENR_PWREN	453
7.94.2.76 RCC_APB1ENR_I2C3EN	453
7.94.2.77 RCC_APB1ENR_I2C2EN	453
7.94.2.78 RCC_APB1ENR_I2C1EN	453
7.94.2.79 RCC_APB1ENR_USART2EN	453
7.94.2.80 RCC_APB1ENR_SPI3EN	454
7.94.2.81 RCC_APB1ENR_SPI2EN	454
7.94.2.82 RCC_APB1ENR_WWDGEN	454
7.94.2.83 RCC_APB1ENR_TIM5EN	454
7.94.2.84 RCC_APB1ENR_TIM4EN	454
7.94.2.85 RCC_APB1ENR_TIM3EN	454
7.94.2.86 RCC_APB1ENR_TIM2EN	455
7.94.2.87 RCC_APB2ENR_TIM11EN	455

7.94.2.88 RCC_APB2ENR_TIM10EN	455
7.94.2.89 RCC_APB2ENR_TIM9EN	455
7.94.2.90 RCC_APB2ENR_SYSCFGEN	455
7.94.2.91 RCC_APB2ENR_SPI4EN	455
7.94.2.92 RCC_APB2ENR_SPI1EN	456
7.94.2.93 RCC_APB2ENR_SDIOEN	456
7.94.2.94 RCC_APB2ENR_ADC1EN	456
7.94.2.95 RCC_APB2ENR_USART6EN	456
7.94.2.96 RCC_APB2ENR_USART1EN	456
7.94.2.97 RCC_APB2ENR_TIM1EN	456
7.94.2.98 RCC_AHB1LPENR_FLITFLPEN	457
7.94.2.99 ENABLE	457
7.94.2.100 DISABLE	457
7.94.2.101 BYBASED	457
7.94.2.102 NOTBYBASED	457
7.94.2.103 SYSCLK	457
7.94.2.104 PLLI2SCLK	458
7.94.2.105 HSE	458
7.94.2.106 PLLCLK [1/2]	458
7.94.2.107 NoDivision	458
7.94.2.108 DivisionBy2	458
7.94.2.109 DivisionBy3	458
7.94.2.110 DivisionBy4	459
7.94.2.111 DivisionBy5	459
7.94.2.112 I2S_CKIN	459
7.94.2.113 HSI	459
7.94.2.114 LSE	459
7.94.2.115 PLLCLK [2/2]	459
7.94.2.116 SYSCLKby2	460
7.94.2.117 SYSCLKby4	460
7.94.2.118 SYSCLKby8	460
7.94.2.119 SYSCLKby16	460
7.94.2.120 SYSCLKby64	460
7.94.2.121 SYSCLKby128	460
7.94.2.122 SYSCLKby256	461
7.94.2.123 SYSCLKby512	461
7.94.2.124 NoCLK0	461
7.94.2.125 NoCLK1	461
7.94.2.126 HSEby2	461
7.94.2.127 HSEby3	461
7.94.2.128 HSEby4	462
7.94.2.129 HSEby5	462

7.94.2.130 HSEby6	462
7.94.2.131 HSEby7	462
7.94.2.132 HSEby8	462
7.94.2.133 HSEby9	462
7.94.2.134 HSEby10	463
7.94.2.135 HSEby11	463
7.94.2.136 HSEby12	463
7.94.2.137 HSEby13	463
7.94.2.138 HSEby14	463
7.94.2.139 HSEby15	463
7.94.2.140 HSEby16	464
7.94.2.141 HSEby17	464
7.94.2.142 HSEby18	464
7.94.2.143 HSEby19	464
7.94.2.144 HSEby20	464
7.94.2.145 HSEby21	464
7.94.2.146 HSEby22	465
7.94.2.147 HSEby23	465
7.94.2.148 HSEby24	465
7.94.2.149 HSEby25	465
7.94.2.150 HSEby26	465
7.94.2.151 HSEby27	465
7.94.2.152 HSEby28	466
7.94.2.153 HSEby29	466
7.94.2.154 HSEby30	466
7.94.2.155 HSEby31	466
7.94.2.156 AHBby2	466
7.94.2.157 AHBby4	466
7.94.2.158 AHBby8	467
7.94.2.159 AHBby16	467
7.94.2.160 Equal_0	467
7.94.2.161 Equal_1	467
7.94.2.162 Equal_2	467
7.94.2.163 Equal_3	467
7.94.2.164 Equal_4	468
7.94.2.165 Equal_5	468
7.94.2.166 Equal_6	468
7.94.2.167 Equal_7	468
7.94.2.168 Equal_8	468
7.94.2.169 Equal_9	468
7.94.2.170 Equal_10	469
7.94.2.171 Equal_11	469

7.94.2.172 Equal_12	469
7.94.2.173 Equal_13	469
7.94.2.174 Equal_14	469
7.94.2.175 Equal_15	469
7.94.2.176 Equal_16	470
7.94.2.177 Equal_17	470
7.94.2.178 Equal_18	470
7.94.2.179 Equal_19	470
7.94.2.180 Equal_20	470
7.94.2.181 Equal_21	470
7.94.2.182 Equal_22	471
7.94.2.183 Equal_23	471
7.94.2.184 Equal_24	471
7.94.2.185 Equal_25	471
7.94.2.186 Equal_26	471
7.94.2.187 Equal_27	471
7.94.2.188 Equal_28	472
7.94.2.189 Equal_29	472
7.94.2.190 Equal_30	472
7.94.2.191 Equal_31	472
7.94.2.192 Equal_32	472
7.94.2.193 Equal_33	472
7.94.2.194 Equal_34	473
7.94.2.195 Equal_35	473
7.94.2.196 Equal_36	473
7.94.2.197 Equal_37	473
7.94.2.198 Equal_38	473
7.94.2.199 Equal_39	473
7.94.2.200 Equal_40	474
7.94.2.201 Equal_41	474
7.94.2.202 Equal_42	474
7.94.2.203 Equal_43	474
7.94.2.204 Equal_44	474
7.94.2.205 Equal_45	474
7.94.2.206 Equal_46	475
7.94.2.207 Equal_47	475
7.94.2.208 Equal_48	475
7.94.2.209 Equal_49	475
7.94.2.210 Equal_50	475
7.94.2.211 Equal_51	475
7.94.2.212 Equal_52	476
7.94.2.213 Equal_53	476

7.94.2.214 Equal_54	476
7.94.2.215 Equal_55	476
7.94.2.216 Equal_56	476
7.94.2.217 Equal_57	476
7.94.2.218 Equal_58	477
7.94.2.219 Equal_59	477
7.94.2.220 Equal_60	477
7.94.2.221 Equal_61	477
7.94.2.222 Equal_62	477
7.94.2.223 Equal_63	477
7.95 MRCC_private.h	478
7.96 COTS/MCAL/RCC/MRCC_program.c File Reference	482
7.96.1 Detailed Description	482
7.96.2 Function Documentation	483
7.96.2.1 MRCC_vInit()	483
7.96.2.2 MRCC_vEnablePeriphralCLK()	487
7.96.2.3 MRCC_vDisablePeriphralCLK()	488
7.97 MRCC_program.c	489
7.98 COTS/MCAL/SPI/SPI_config.h File Reference	495
7.98.1 Macro Definition Documentation	495
7.98.1.1 CPHA_MODE	496
7.98.1.2 CPOL_MODE	496
7.98.1.3 MSTR_MODE	496
7.98.1.4 BR_MODE	496
7.98.1.5 SPE_MODE	496
7.98.1.6 LSBFIRST_MODE	496
7.98.1.7 SSM_MODE	497
7.98.1.8 RXONLY_MODE	497
7.98.1.9 DFF_MODE	497
7.98.1.10 CRCNEXT_MODE	497
7.98.1.11 CRCEN_MODE	497
7.98.1.12 BIDIOE_MODE	497
7.98.1.13 BIDMODE_MODE	498
7.98.1.14 HALF_DUPLEX_MODE	498
7.98.1.15 SIMPLEX_MODE	498
7.98.1.16 RXDMAEN_MODE	498
7.98.1.17 TXDMAEN_MODE	498
7.98.1.18 MULTI_MSTR_MODE	498
7.98.1.19 FRF_MODE	499
7.98.1.20 ERRIE_MODE	499
7.98.1.21 RXNEIE_MODE [1/2]	499
7.98.1.22 TXNEIE_MODE [1/2]	499

7.98.1.23 FRE_MODE	499
7.98.1.24 BSY_MODE	499
7.98.1.25 OVR_MODE	500
7.98.1.26 MODF_MODE	500
7.98.1.27 UDR_MODE	500
7.98.1.28 CRCERR_MODE	500
7.98.1.29 RXNEIE_MODE [2/2]	500
7.98.1.30 TXNEIE_MODE [2/2]	500
7.99 SPI_config.h	501
7.100 COTS/MCAL/SPI/SPI_interface.h File Reference	504
7.100.1 Macro Definition Documentation	505
7.100.1.1 SPI1_BASE_ADDRESS	505
7.100.1.2 SPI2_BASE_ADDRESS	505
7.100.1.3 SPI3_BASE_ADDRESS	505
7.100.1.4 SPI4_BASE_ADDRESS	505
7.100.1.5 SPI1	505
7.100.1.6 SPI2	506
7.100.1.7 SPI3	506
7.100.1.8 SPIx_MSTR	506
7.100.1.9 SPIx_SLAVE	506
7.100.1.10 SPIx_SIMPLEX	506
7.100.1.11 SPIx_HALF_DUPLEX	506
7.100.1.12 SPIx_FULL_DUPLEX	507
7.100.2 Function Documentation	507
7.100.2.1 MSPI_vInit()	507
7.100.2.2 MSPI_u16Transceive()	511
7.100.2.3 MSPI_vMasterTransmit()	511
7.100.2.4 MSPI_u16MasterRecieve()	512
7.100.2.5 MSPI_vSlaveTransmit()	512
7.100.2.6 MSPI_u16SlaveRecieve()	512
7.100.2.7 MSPI_vDISABLE()	513
7.100.2.8 MSPI1_vSetCallBack()	514
7.100.2.9 MSPI2_vSetCallBack()	515
7.100.2.10 MSPI3_vSetCallBack()	515
7.101 SPI_interface.h	515
7.102 COTS/MCAL/SPI/SPI_private.h File Reference	516
7.102.1 Macro Definition Documentation	518
7.102.1.1 CPHA	518
7.102.1.2 CPOL	518
7.102.1.3 MSTR	518
7.102.1.4 SPE	518
7.102.1.5 LSBFIRST	518

7.102.1.6 SSI	519
7.102.1.7 SSM	519
7.102.1.8 RXONLY	519
7.102.1.9 DFF	519
7.102.1.10 CRCNEXT	519
7.102.1.11 CRCEN	519
7.102.1.12 BIDIOE	520
7.102.1.13 BIDI MODE	520
7.102.1.14 RXDMAEN	520
7.102.1.15 TXDMAEN	520
7.102.1.16 SSOE	520
7.102.1.17 FRF	520
7.102.1.18 ERRIE	521
7.102.1.19 RXNEIE	521
7.102.1.20 TXEIE	521
7.102.1.21 RXNE	521
7.102.1.22 TXE	521
7.102.1.23 CHSIDE	521
7.102.1.24 UDR	522
7.102.1.25 CRCERR	522
7.102.1.26 MODF	522
7.102.1.27 OVR	522
7.102.1.28 BSY	522
7.102.1.29 FRE	522
7.102.1.30 FIRST_CLK_Captured	523
7.102.1.31 SECOND_CLK_Captured	523
7.102.1.32 CLK_IdleAt0	523
7.102.1.33 CLK_IdleAt1	523
7.102.1.34 SLAVE	523
7.102.1.35 MASTER	523
7.102.1.36 CLK_By2	524
7.102.1.37 CLK_By4	524
7.102.1.38 CLK_By8	524
7.102.1.39 CLK_By16	524
7.102.1.40 CLK_By32	524
7.102.1.41 CLK_By64	524
7.102.1.42 CLK_By128	525
7.102.1.43 CLK_By256	525
7.102.1.44 ENABLE	525
7.102.1.45 DISABLE	525
7.102.1.46 MSB_First	525
7.102.1.47 LSB_First	525

7.102.1.48 FULL_DUPLEX	526
7.102.1.49 RX_ONLY	526
7.102.1.50 EIGHT_BITS	526
7.102.1.51 SIXTEEN_BITS	526
7.102.1.52 CRC_PHASE	526
7.102.1.53 NO_CRC_PHASE	526
7.102.1.54 OneLine	527
7.102.1.55 TwoLines	527
7.102.1.56 MOTOROLA	527
7.102.1.57 TI	527
7.102.1.58 EMPTY	527
7.102.1.59 NOTEMPTY	527
7.102.1.60 LEFT	528
7.102.1.61 RIGHT	528
7.102.1.62 OCCURRED	528
7.102.1.63 NOT_OCCURRED	528
7.102.1.64 MATCHED	528
7.102.1.65 NOT_MATCHED	528
7.102.1.66 FAULT_OCCURRED	529
7.102.1.67 NO_FAULT_OCCURRED	529
7.102.1.68 BUSY	529
7.102.1.69 NOT_BUSY	529
7.102.1.70 ERROR	529
7.102.1.71 NO_ERROR	529
7.102.1.72 HALF_DUPLEX_TX	530
7.102.1.73 HALF_DUPLEX_RX	530
7.102.1.74 SIMPLEX_TX	530
7.102.1.75 SIMPLEX_RX	530
7.102.1.76 SR_RESET	530
7.103 SPI_private.h	531
7.104 COTS/MCAL/SPI/SPI_program.c File Reference	532
7.104.1 Function Documentation	533
7.104.1.1 MSPI_vInit()	533
7.104.1.2 MSPI_u16Transcieve()	537
7.104.1.3 MSPI_vMasterTransmit()	537
7.104.1.4 MSPI_u16MasterRecieve()	538
7.104.1.5 MSPI_vSlaveTransmit()	538
7.104.1.6 MSPI_u16SlaveRecieve()	538
7.104.1.7 MSPI_vDISABLE()	539
7.104.1.8 MSPI1_vSetCallBack()	540
7.104.1.9 MSPI2_vSetCallBack()	541
7.104.1.10 MSPI3_vSetCallBack()	541

7.104.1.11 SPI1_IRQHandler()	541
7.104.1.12 SPI2_IRQHandler()	541
7.104.1.13 SPI3_IRQHandler()	542
7.104.2 Variable Documentation	542
7.104.2.1 MSPI1_CallBack	542
7.104.2.2 MSPI2_CallBack	542
7.104.2.3 MSPI3_CallBack	542
7.105 SPI_program.c	543
7.106 COTS/MCAL/SysTick/SysTick_config.h File Reference	550
7.106.1 Macro Definition Documentation	550
7.106.1.1 Exception_Request	550
7.106.1.2 MAX_TICKS	551
7.107 SysTick_config.h	551
7.108 COTS/MCAL/SysTick/SysTick_interface.h File Reference	551
7.108.1 Detailed Description	552
7.108.2 Macro Definition Documentation	553
7.108.2.1 BUSY_TICK_TIME	553
7.108.2.2 SINGLE_INTERVAL_TICK_TIME	553
7.108.2.3 PERIODIC_INTERVAL_TICK_TIME	553
7.108.3 Function Documentation	553
7.108.3.1 MSysTick_vInit()	554
7.108.3.2 MSysTick_vSetBusyWait()	554
7.108.3.3 MSysTick_vDelay()	554
7.108.3.4 MSysTick_vDelayMicroSec()	555
7.108.3.5 MSysTick_vDelayMilliSec()	555
7.108.3.6 MSysTick_vDelaySec()	555
7.108.3.7 MSysTick_vSetSingleInterval()	556
7.108.3.8 MSysTick_vSetPeriodicInterval()	556
7.108.3.9 MSysTick_vStopInterval()	556
7.108.3.10 MSysTick_u32GetElapsedTime()	557
7.108.3.11 MSysTick_u32GetRemainingTime()	557
7.108.3.12 MSysTick_vEnable()	558
7.108.3.13 MSysTick_vDisable()	558
7.108.3.14 MSysTick_vEnableException()	558
7.109 SysTick_interface.h	559
7.110 COTS/MCAL/SysTick/SysTick_private.h File Reference	559
7.110.1 Detailed Description	560
7.111 SysTick_private.h	560
7.112 COTS/MCAL/SysTick/SysTick_program.c File Reference	561
7.112.1 Detailed Description	561
7.112.2 Function Documentation	562
7.112.2.1 MSysTick_vEnableException()	562

7.112.2.2 MSysTick_vInit()	562
7.112.2.3 MSysTick_vSetBusyWait()	563
7.112.2.4 MSysTick_vDelay()	563
7.112.2.5 MSysTick_vDelayMicroSec()	564
7.112.2.6 MSysTick_vDelayMilliSec()	564
7.112.2.7 MSysTick_vDelaySec()	565
7.112.2.8 MSysTick_vSetSingleInterval()	566
7.112.2.9 MSysTick_vSetPeriodicInterval()	566
7.112.2.10 MSysTick_vStopInterval()	567
7.112.2.11 MSysTick_u32GetElapsedTime()	567
7.112.2.12 MSysTick_u32GetRemainingTime()	568
7.112.2.13 MSysTick_vEnable()	568
7.112.2.14 MSysTick_vDisable()	568
7.112.2.15 SysTick_Handler()	569
7.113 SysTick_program.c	569
7.114 COTS/MCAL/UART/UART_config.h File Reference	574
7.114.1 Macro Definition Documentation	574
7.114.1.1 __PCLK__	575
7.114.1.2 THRESHOLD_VALUE	575
7.114.1.3 USART1_PORT	575
7.114.1.4 TX1_PIN	575
7.114.1.5 RX1_PIN	575
7.114.1.6 USART2_PORT	575
7.114.1.7 TX2_PIN	576
7.114.1.8 RX2_PIN	576
7.114.1.9 USART6_PORT	576
7.114.1.10 TX6_PIN	576
7.114.1.11 RX6_PIN	576
7.115 UART_config.h	577
7.116 COTS/MCAL/UART/UART_interface.h File Reference	578
7.116.1 Macro Definition Documentation	579
7.116.1.1 USART1_BASE_ADDRESS	579
7.116.1.2 USART2_BASE_ADDRESS	579
7.116.1.3 USART6_BASE_ADDRESS	579
7.116.1.4 USART1_REG	580
7.116.1.5 USART2_REG	580
7.116.1.6 USART6_REG	580
7.116.1.7 OVER_SAMPLING_16	580
7.116.1.8 OVER_SAMPLING_8	580
7.116.1.9 TX_ONLY	580
7.116.1.10 RX_ONLY	581
7.116.1.11 TX_RX	581

7.116.1.12 EVEN_PARITY	581
7.116.1.13 ODD_PARITY	581
7.116.1.14 MODE_8BIT	581
7.116.1.15 MODE_9BIT	581
7.116.1.16 STOP_BIT_1	582
7.116.1.17 STOP_BIT_0_5	582
7.116.1.18 STOP_BIT_2	582
7.116.1.19 STOP_BIT_1_5	582
7.116.1.20 ENABLE	582
7.116.1.21 DISABLE	582
7.116.1.22 __BAUDRATE__	583
7.116.2 Function Documentation	583
7.116.2.1 MUSART_vInit()	583
7.116.2.2 MUSART_vEnable()	585
7.116.2.3 MUSART_vDisable()	585
7.116.2.4 MUSART_vTransmitByte()	586
7.116.2.5 MUSART_vTransmitString()	586
7.116.2.6 MUSART_u8ReceiveByteSynchNonBlocking()	587
7.116.2.7 MUSART_u8ReceiveByteSynchBlocking()	587
7.116.2.8 MUSART_ptrReceiveStringSynchNonBlocking()	588
7.116.2.9 MUSART_vRecieveString()	588
7.116.2.10 MUSART_u8CompareString()	589
7.116.2.11 MUSART_u8ReadDataRegister()	589
7.116.2.12 MUSART_vClearFlags()	589
7.116.2.13 MUSART_vRxIntSetStatus()	590
7.116.2.14 MUSART1_vSetCallBack()	590
7.116.2.15 MUSART2_vSetCallBack()	590
7.116.2.16 MUSART6_vSetCallBack()	590
7.117 UART_interface.h	591
7.118 COTS/MCAL/UART/UART_private.h File Reference	593
7.118.1 Macro Definition Documentation	594
7.118.1.1 UART_DIV_SAMPLING16	594
7.118.1.2 UART_DIVMANT_SAMPLING16	594
7.118.1.3 UART_DIVFRAQ_SAMPLING16	594
7.118.1.4 UART_BRR_SAMPLING16	595
7.118.1.5 UART_DIV_SAMPLING8	595
7.118.1.6 UART_DIVMANT_SAMPLING8	595
7.118.1.7 UART_DIVFRAQ_SAMPLING8	595
7.118.1.8 UART_BRR_SAMPLING8	596
7.118.1.9 MUSART_SR_PE_BIT	596
7.118.1.10 MUSART_SR_FE_BIT	596
7.118.1.11 MUSART_SR_NE_BIT	596

7.118.1.12 MUSART_SR_ORE_BIT	596
7.118.1.13 MUSART_SR_IDLE_BIT	597
7.118.1.14 MUSART_SR_RXNE_BIT	597
7.118.1.15 MUSART_SR_TC_BIT	597
7.118.1.16 MUSART_SR_TXE_BIT	597
7.118.1.17 MUSART_SR_LBD_BIT	597
7.118.1.18 MUSART_SR_CTS_BIT	597
7.118.1.19 MUSART_CR1_SBK_BIT	598
7.118.1.20 MUSART_CR1_RWU_BIT	598
7.118.1.21 MUSART_CR1_RE_BIT	598
7.118.1.22 MUSART_CR1_TE_BIT	598
7.118.1.23 MUSART_CR1_IDLEIE_BIT	598
7.118.1.24 MUSART_CR1_RXNEIE_BIT	598
7.118.1.25 MUSART_CR1_TCIE_BIT	599
7.118.1.26 MUSART_CR1_TXEIE_BIT	599
7.118.1.27 MUSART_CR1_PEIE_BIT	599
7.118.1.28 MUSART_CR1_PS_BIT	599
7.118.1.29 MUSART_CR1_PCE_BIT	599
7.118.1.30 MUSART_CR1_WAKE_BIT	599
7.118.1.31 MUSART_CR1_M_BIT	600
7.118.1.32 MUSART_CR1_UE_BIT	600
7.118.1.33 MUSART_CR1_OVER8_BIT	600
7.118.1.34 MUSART_CR2_ADD0_BIT	600
7.118.1.35 MUSART_CR2_ADD1_BIT	600
7.118.1.36 MUSART_CR2_ADD2_BIT	600
7.118.1.37 MUSART_CR2_ADD3_BIT	601
7.118.1.38 MUSART_CR2_LBDEL_BIT	601
7.118.1.39 MUSART_CR2_LBDIE_BIT	601
7.118.1.40 MUSART_CR2_LBCL_BIT	601
7.118.1.41 MUSART_CR2_CPHA_BIT	601
7.118.1.42 MUSART_CR2_CPOL_BIT	601
7.118.1.43 MUSART_CR2_CLKEN_BIT	602
7.118.1.44 MUSART_CR2_STOP_BIT	602
7.118.1.45 MUSART_CR2_STOP0_BIT	602
7.118.1.46 MUSART_CR2_STOP1_BIT	602
7.118.1.47 MUSART_CR2_LINEN_BIT	602
7.118.1.48 MUSART_CR3_CTSIE_BIT	602
7.118.1.49 MUSART_CR3_CTSE_BIT	603
7.118.1.50 MUSART_CR3_RTSE_BIT	603
7.118.1.51 MUSART_CR3_DMAT_BIT	603
7.118.1.52 MUSART_CR3_DMAR_BIT	603
7.118.1.53 MUSART_CR3_SCEN_BIT	603

7.118.1.54 MUSART_CR3_NACK_BIT	603
7.118.1.55 MUSART_CR3_HDSEL_BIT	604
7.118.1.56 MUSART_CR3_IRLP_BIT	604
7.118.1.57 MUSART_CR3_IREN_BIT	604
7.118.1.58 MUSART_CR3_EIE_BIT	604
7.119 UART_private.h	604
7.120 COTS/MCAL/UART/UART_program.c File Reference	606
7.120.1 Function Documentation	607
7.120.1.1 MUSART_vInit()	607
7.120.1.2 MUSART_vEnable()	609
7.120.1.3 MUSART_vDisable()	609
7.120.1.4 MUSART_vTransmitByte()	610
7.120.1.5 MUSART_vTransmitString()	610
7.120.1.6 MUSART_u8ReceiveByteSynchNonBlocking()	611
7.120.1.7 MUSART_u8ReceiveByteSynchBlocking()	611
7.120.1.8 MUSART_ptrReceiveStringSynchNonBlocking()	612
7.120.1.9 MUSART_vRecieveString()	612
7.120.1.10 MUSART_u8CompareString()	613
7.120.1.11 MUSART_u8ReadDataRegister()	613
7.120.1.12 MUSART_vClearFlags()	613
7.120.1.13 MUSART_vRxIntSetStatus()	614
7.120.1.14 MUSART1_vSetCallBack()	614
7.120.1.15 MUSART2_vSetCallBack()	614
7.120.1.16 MUSART6_vSetCallBack()	614
7.120.1.17 USART1_IRQHandler()	615
7.120.1.18 USART2_IRQHandler()	615
7.120.1.19 USART6_IRQHandler()	615
7.120.2 Variable Documentation	615
7.120.2.1 G_u8String	616
7.120.2.2 MUSART1_CallBack	616
7.120.2.3 MUSART2_CallBack	616
7.120.2.4 MUSART6_CallBack	616
7.121 UART_program.c	617
7.122 COTS/Services/UsersLogin/UsersLogin_config.h File Reference	622
7.123 UsersLogin_config.h	622
7.124 COTS/Services/UsersLogin/UsersLogin_interface.h File Reference	622
7.124.1 Function Documentation	622
7.124.1.1 UsersLogin_vUserLoginProcess()	623
7.125 UsersLogin_interface.h	623
7.126 COTS/Services/UsersLogin/UsersLogin_private.h File Reference	623
7.126.1 Macro Definition Documentation	623
7.126.1.1 MAX_INFO_LENGTH	624

7.127 UsersLogin_private.h	624
7.128 COTS/Services/UsersLogin/UsersLogin_program.c File Reference	624
7.128.1 Function Documentation	624
7.128.1.1 UsersLogin_vUserLoginProcess()	625
7.129 UsersLogin_program.c	625
Index	627

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 Health Care

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	9
DC Motor Rotation Directions	10
Bit Manipulation Math Macros	10
Bit Group Manipulation Math Macros	12
Compiler standard macros	14
Utilities macros	16
Standard types	17
Standard values	20
Interrupt line IDs	24
Interrupt trigger status	28
EXTI Memory Addresses	29
EXTI Registers	30
EXTI Lines Status	30
EXTI Line Settings	31
SYSCFG Memory Addresses	32
SYSCFG Memory Registers	32
GPIO Modes	33
GPIO Ports	34
GPIO Output Types	35
GPIO PIN Speed	36
GPIO Pull Types	37
GPIO Output Values	38
GPIO Output PINs	39
GPIO Alternate Functions	43
GPIO Addresses	48
GPIO Registers	49
Interrupt priority grouping	50
Group priorities	54
Sub-Group priorities	59
NVIC Vector Table	64
NVIC Addresses	81
NVIC Registers	82
NVIC Settable Priorities	82
ID options	84
ID options	86

Systick Interval Mode Configuration	96
Delay Units	97
Systick Addresses	98
Systick Registers	98
Systick Register Bits Positions	99
Systick Clock Sources	100
Systick Exception (Interrupt) Status	101

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	103
DCM_MotorConfiguration	DC Motor configuration structure for motor initialization	108
EXTI_ConfigType	Interrupt configuration structure to initialize the interrupt with	110
EXTI_Type	EXTI Configuration structure for interrupt initialization process	111
GPIOx_MemoryMapType	GPIO declaration structure for its registers	113
MADC_ConfigType		116
MGPIOx_ConfigType	MDIO Configuration structure for a specific PIN initialization	117
MSYSCFG_MemMap_t	System configuration structure to initialize the SYSCFG module with	119
NVIC_MemoryMapType	NVIC configuration structure for NVIC memory map	121
RCC_MemoryMapType	RCC declaration structure for its registers	125
SCB_MemoryMapType	System control block memory map structure	134
SPI_MemoryMapType		140
SysTick_Type	Systick memory map structure declaration for the Systick's registers	142
USART_ClockInitTypeDef		144
USART_InitType		145
USART_MemoryMapType		147
UserInfo_t		149

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

COTS/HAL/Bluetooth/ Bluetooth_config.c	151
COTS/HAL/Bluetooth/ Bluetooth_config.h	153
COTS/HAL/Bluetooth/ Bluetooth_interface.h	153
COTS/HAL/Bluetooth/ Bluetooth_private.h	156
COTS/HAL/Bluetooth/ Bluetooth_program.c	156
COTS/HAL/CarControl/ CarControl_config.h	163
COTS/HAL/CarControl/ CarControl_interface.h	163
COTS/HAL/CarControl/ CarControl_private.h	165
COTS/HAL/CarControl/ CarControl_program.c	165
COTS/HAL/DCMOTOR/ DCM_config.h This file contains the configuration information for the DC Motor module	170
COTS/HAL/DCMOTOR/ DCM_interface.h This file contains the interfacing information for the DC Motor module	171
COTS/HAL/DCMOTOR/ DCM_private.h This file contains the private information for the DC Motor module	174
COTS/HAL/DCMOTOR/ DCM_program.c This file contains the source code of the interfacing information for the DC Motor module	175
COTS/HAL/LCD/ LCD_config.h	178
COTS/HAL/LCD/ LCD_interface.h	185
COTS/HAL/LCD/ LCD_private.h	198
COTS/HAL/LCD/ LCD_program.c	201
COTS/HAL/LDR/ LDR_config.c	215
COTS/HAL/LDR/ LDR_config.h	216
COTS/HAL/LDR/ LDR_interface.h	217
COTS/HAL/LDR/ LDR_private.h	220
COTS/HAL/LDR/ LDR_program.c	220
COTS/LIB/LSTD_BITMATH.h This file contains the bit math manipulation macro-functions	222
COTS/LIB/LSTD_COMPILER.h This file contains the compiler standard macros	223
COTS/LIB/LSTD MCU UTILITIES.h This file contains the MCU utility macro-functions	225
COTS/LIB/LSTD TYPES.h This file contains the standard types	225
COTS/LIB/LSTD VALUES.h This file contains the standard values	226

COTS/MCAL/ADC/ ADC_config.h	228
COTS/MCAL/ADC/ ADC_interface.h	245
COTS/MCAL/ADC/ ADC_private.h	257
COTS/MCAL/ADC/ ADC_program.c	316
COTS/MCAL/EXTI/ EXTI_config.h	328
COTS/MCAL/EXTI/ EXTI_interface.h This file contains the interfacing information for the EXTI module	337
COTS/MCAL/EXTI/ EXTI_private.h This file contains the private information related to the EXTI module	345
COTS/MCAL/EXTI/ EXTI_program.c This file contains the source code of the interfacing for the EXTI module	346
COTS/MCAL/EXTI/ SYSCFG_private.h This file contains the private information regarding the SYSCFG	360
COTS/MCAL/GPIO/ GPIO_config.h This file contains the GPIO configurations	361
COTS/MCAL/GPIO/ GPIO_interface.h This file contains the interfacing information for the GPIO module	363
COTS/MCAL/GPIO/ GPIO_private.h This file contains the registers information and addresses for the GPIO module	374
COTS/MCAL/GPIO/ GPIO_program.c This file contains the source code of the interfacing for the GPIO modules	375
COTS/MCAL/NVIC/ NVIC_config.h	390
COTS/MCAL/NVIC/ NVIC_interface.h This file contains the interface information and addresses for the NVIC module	390
COTS/MCAL/NVIC/ NVIC_private.h This file contains the registers information and addresses for the NVIC module	399
COTS/MCAL/NVIC/ NVIC_program.c This file contains the source code of the interfacing for the NVIC modules	402
COTS/MCAL/RCC/ MRCC_config.h This file contains the RCC configurations	408
COTS/MCAL/RCC/ MRCC_interface.h This file contains the interfacing information for the RCC module	427
COTS/MCAL/RCC/ MRCC_private.h This file contains the registers information and addresses for the RCC module	435
COTS/MCAL/RCC/ MRCC_program.c This file contains the source code of the interfacing for the RCC module	482
COTS/MCAL/SPI/ SPI_config.h	495
COTS/MCAL/SPI/ SPI_interface.h	504
COTS/MCAL/SPI/ SPI_private.h	516
COTS/MCAL/SPI/ SPI_program.c	532
COTS/MCAL/SysTick/ SysTick_config.h	550
COTS/MCAL/SysTick/ SysTick_interface.h This file contains the interfacing information for the Systick module	551
COTS/MCAL/SysTick/ SysTick_private.h This file contains the registers information and addresses for the Systick module	559
COTS/MCAL/SysTick/ SysTick_program.c This file contains the source code of the interfacing for the Systick modules	561
COTS/MCAL/UART/ UART_config.h	574
COTS/MCAL/UART/ UART_interface.h	578
COTS/MCAL/UART/ UART_private.h	593
COTS/MCAL/UART/ UART_program.c	606
COTS/Services/UsersLogin/ UsersLogin_config.h	622
COTS/Services/UsersLogin/ UsersLogin_interface.h	622
COTS/Services/UsersLogin/ UsersLogin_private.h	623
COTS/Services/UsersLogin/ UsersLogin_program.c	624

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 99 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 106 of file DCM_interface.h.
```

5.2 DC Motor Rotation Directions

Macros

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

5.2.1 Detailed Description

5.2.2 Macro Definition Documentation

5.2.2.1 CW

```
#define CW (1)

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

Motor clockwise rotation direction.

Definition at line 24 of file DCM_private.h.
```

5.2.2.2 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.
```

5.3 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)`

5.3.1 Detailed Description

5.3.2 Macro Definition Documentation

5.3.2.1 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](#).

5.3.2.2 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](#).

5.3.2.3 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](#).

5.3.2.4 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](#).

5.4 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))

5.4.1 Detailed Description

5.4.2 Macro Definition Documentation

5.4.2.1 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](#).

5.4.2.2 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](#).

5.4.2.3 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](#).

5.4.2.4 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](#).

5.5 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

5.5.1 Detailed Description

5.5.2 Macro Definition Documentation

5.5.2.1 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](#).

5.5.2.2 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](#).

5.5.2.3 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](#).

5.5.2.4 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](#).

5.5.2.5 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](#).

5.5.2.6 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](#).

5.5.2.7 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](#).

5.5.2.8 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](#).

5.5.2.9 STATIC

```
#define STATIC static

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

Declare a standard static variable

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

5.6 Utilities macros

Macros

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

5.6.1 Detailed Description

5.6.2 Macro Definition Documentation

5.6.2.1 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](#).

5.7 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`

5.7.1 Detailed Description

5.7.2 Typedef Documentation

5.7.2.1 bool_t

```
bool_t

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

Type definition for boolean

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

5.7.2.2 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](#).

5.7.2.3 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](#).

5.7.2.4 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](#).

5.7.2.5 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](#).

5.7.2.6 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](#).

5.7.2.7 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](#).

5.7.2.8 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](#).

5.7.2.9 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](#).

5.7.2.10 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](#).

5.7.2.11 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](#).

5.7.2.12 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](#).

5.8 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)

5.8.1 Detailed Description

5.8.2 Macro Definition Documentation

5.8.2.1 TRUE

```
#define TRUE (1)

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

Type definition for TRUE

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

5.8.2.2 FALSE

```
#define FALSE (0)

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

Type definition for FALSE

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

5.8.2.3 NULL

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

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

Type definition for NULL

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

5.8.2.4 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](#).

5.8.2.5 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](#).

5.8.2.6 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](#).

5.8.2.7 RUN

```
#define RUN (1)

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

Type definition for RUN

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

5.8.2.8 STOP

```
#define STOP (0)

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

Type definition for STOP

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

5.8.2.9 PRESSED

```
#define PRESSED (1)

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

Type definition for PRESSED

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

5.8.2.10 RELEASED

```
#define RELEASED (0)

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

Type definition for RELEASED

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

5.8.2.11 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](#).

5.8.2.12 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](#).

5.9 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.

5.9.1 Detailed Description

5.9.2 Macro Definition Documentation

5.9.2.1 EXTI_LINE0

```
#define EXTI_LINE0 (0)

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

Line 0.
```

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

5.9.2.2 EXTI_LINE1

```
#define EXTI_LINE1 (1)

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

Line 1.

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

5.9.2.3 EXTI_LINE2

```
#define EXTI_LINE2 (2)

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

Line 2.

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

5.9.2.4 EXTI_LINE3

```
#define EXTI_LINE3 (3)

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

Line 3.

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

5.9.2.5 EXTI_LINE4

```
#define EXTI_LINE4 (4)

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

Line 4.

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

5.9.2.6 EXTI_LINE5

```
#define EXTI_LINE5 (5)

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

Line 5.

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

5.9.2.7 EXTI_LINE6

```
#define EXTI_LINE6 (6)

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

Line 6.

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

5.9.2.8 EXTI_LINE7

```
#define EXTI_LINE7 (7)

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

Line 7.

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

5.9.2.9 EXTI_LINE8

```
#define EXTI_LINE8 (8)

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

Line 8.

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

5.9.2.10 EXTI_LINE9

```
#define EXTI_LINE9 (9)

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

Line 9.

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

5.9.2.11 EXTI_LINE10

```
#define EXTI_LINE10 (10)

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

Line 10.

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

5.9.2.12 EXTI_LINE11

```
#define EXTI_LINE11 (11)

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

Line 11.

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

5.9.2.13 EXTI_LINE12

```
#define EXTI_LINE12 (12)

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

Line 12.

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

5.9.2.14 EXTI_LINE13

```
#define EXTI_LINE13 (13)

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

Line 13.

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

5.9.2.15 EXTI_LINE14

```
#define EXTI_LINE14 (14)

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

Line 14.

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

5.9.2.16 EXTI_LINE15

```
#define EXTI_LINE15 (15)

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

Line 15.

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

5.10 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

5.10.1 Detailed Description

5.10.2 Macro Definition Documentation

5.10.2.1 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.
```

5.10.2.2 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.
```

5.10.2.3 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.
```

5.11 EXTI Memory Addresses

Macros

- #define EXTI_BASE_ADDRESS (0x40013C00)

5.11.1 Detailed Description

5.11.2 Macro Definition Documentation

5.11.2.1 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](#).

5.12 EXTI Registers

Macros

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

5.12.1 Detailed Description

5.12.2 Macro Definition Documentation

5.12.2.1 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](#).

5.13 EXTI Lines Status

Macros

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

5.13.1 Detailed Description

5.13.2 Macro Definition Documentation

5.13.2.1 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](#).

5.13.2.2 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](#).

5.14 EXTI Line Settings

Macros

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

5.14.1 Detailed Description

5.14.2 Macro Definition Documentation

5.14.2.1 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.
```

5.15 SYSCFG Memory Addresses

Macros

- #define `SYSCFG_BASE_ADDR` (0x40013800)

5.15.1 Detailed Description

5.15.2 Macro Definition Documentation

5.15.2.1 `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.
```

5.16 SYSCFG Memory Registers

Macros

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

5.16.1 Detailed Description

5.16.2 Macro Definition Documentation

5.16.2.1 `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.
```

5.17 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.

5.17.1 Detailed Description

5.17.2 Macro Definition Documentation

5.17.2.1 GPIOx_MODE_INPUT

```
#define GPIOx_MODE_INPUT (0b00)

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

Control input mode.

Definition at line 166 of file GPIO\_interface.h.
```

5.17.2.2 GPIOx_MODE_OUTPUT

```
#define GPIOx_MODE_OUTPUT (0b01)

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

Control output mode.

Definition at line 173 of file GPIO\_interface.h.
```

5.17.2.3 GPIOx_MODE_AF

```
#define GPIOx_MODE_AF (0b10)

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

Control alternate function mode.

Definition at line 180 of file GPIO\_interface.h.
```

5.17.2.4 GPIOx_MODE_ANALOG

```
#define GPIOx_MODE_ANALOG (0b11)

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

Control analog mode.

Definition at line 187 of file [GPIO_interface.h](#).

5.18 GPIO Ports

Macros

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

5.18.1 Detailed Description

5.18.2 Macro Definition Documentation

5.18.2.1 [GPIO_PORTA](#)

```
#define GPIO_PORTA (0)

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

GPIO Port A.

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

5.18.2.2 [GPIO_PORTB](#)

```
#define GPIO_PORTB (1)

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

GPIO Port B.

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

5.18.2.3 GPIO_PORTC

```
#define GPIO_PORTC (2)

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

GPIO Port C.

Definition at line 216 of file [GPIO_interface.h](#).

5.19 GPIO Output Types

Macros

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

5.19.1 Detailed Description

5.19.2 Macro Definition Documentation

5.19.2.1 GPIOx_OPENDRAIN

```
#define GPIOx_OPENDRAIN (1)

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

GPIO open-drain.
```

Definition at line 231 of file [GPIO_interface.h](#).

5.19.2.2 GPIOx_PUSH_PULL

```
#define GPIOx_PUSH_PULL (2)

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

GPIO push-pull.
```

Definition at line 238 of file [GPIO_interface.h](#).

5.20 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.

5.20.1 Detailed Description

5.20.2 Macro Definition Documentation

5.20.2.1 **GPIOx_LowSpeed**

```
#define GPIOx_LowSpeed (0b00)

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

GPIO low speed.
```

Definition at line 253 of file [GPIO_interface.h](#).

5.20.2.2 **GPIOx_MediumSpeed**

```
#define GPIOx_MediumSpeed (0b01)

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

GPIO medium speed.
```

Definition at line 260 of file [GPIO_interface.h](#).

5.20.2.3 **GPIOx_HighSpeed**

```
#define GPIOx_HighSpeed (0b10)

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

GPIO high speed.
```

Definition at line 267 of file [GPIO_interface.h](#).

5.20.2.4 GPIOx_VeryHighSpeed

```
#define GPIOx_VeryHighSpeed (0b11)

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

GPIO very high speed.

Definition at line 274 of file [GPIO_interface.h](#).

5.21 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.

5.21.1 Detailed Description

5.21.2 Macro Definition Documentation

5.21.2.1 GPIOx_NoPull

```
#define GPIOx_NoPull (0b00)

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

GPIO No PULL.

Definition at line 289 of file [GPIO_interface.h](#).

5.21.2.2 GPIOx_PullUp

```
#define GPIOx_PullUp (0b01)

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

GPIO Pull UP.

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

5.21.2.3 GPIOx_PullDown

```
#define GPIOx_PullDown (0b10)

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

GPIO Pull Down.

Definition at line 303 of file [GPIO_interface.h](#).

5.22 GPIO Output Values

Macros

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

5.22.1 Detailed Description

5.22.2 Macro Definition Documentation

5.22.2.1 GPIOx_HIGH

```
#define GPIOx_HIGH (1)

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

GPIO output high.
```

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

5.22.2.2 GPIOx_LOW

```
#define GPIOx_LOW (2)

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

GPIO output low.
```

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

5.23 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.

5.23.1 Detailed Description

5.23.2 Macro Definition Documentation

5.23.2.1 **GPIOx_PIN0**

```
#define GPIOx_PIN0 (0)

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

GPIO PIN 0.

Definition at line 340 of file GPIO\_interface.h.
```

5.23.2.2 GPIOx_PIN1

```
#define GPIOx_PIN1 (1)

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

GPIO PIN 1.

Definition at line 347 of file [GPIO_interface.h](#).

5.23.2.3 GPIOx_PIN2

```
#define GPIOx_PIN2 (2)

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

GPIO PIN 2.

Definition at line 354 of file [GPIO_interface.h](#).

5.23.2.4 GPIOx_PIN3

```
#define GPIOx_PIN3 (3)

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

GPIO PIN 3.

Definition at line 361 of file [GPIO_interface.h](#).

5.23.2.5 GPIOx_PIN4

```
#define GPIOx_PIN4 (4)

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

GPIO PIN 4.

Definition at line 368 of file [GPIO_interface.h](#).

5.23.2.6 GPIOx_PIN5

```
#define GPIOx_PIN5 (5)

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

GPIO PIN 5.

Definition at line 375 of file [GPIO_interface.h](#).

5.23.2.7 GPIOx_PIN6

```
#define GPIOx_PIN6 (6)

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

GPIO PIN 6.

Definition at line 382 of file [GPIO_interface.h](#).

5.23.2.8 GPIOx_PIN7

```
#define GPIOx_PIN7 (7)

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

GPIO PIN 7.

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

5.23.2.9 GPIOx_PIN8

```
#define GPIOx_PIN8 (8)

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

GPIO PIN 8.

Definition at line 396 of file [GPIO_interface.h](#).

5.23.2.10 GPIOx_PIN9

```
#define GPIOx_PIN9 (9)

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

brief GPIO PIN 9

Definition at line 403 of file [GPIO_interface.h](#).

5.23.2.11 GPIOx_PIN10

```
#define GPIOx_PIN10 (10)

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

GPIO PIN 10.

Definition at line 410 of file [GPIO_interface.h](#).

5.23.2.12 GPIOx_PIN11

```
#define GPIOx_PIN11 (11)

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

GPIO PIN 11.

Definition at line 417 of file [GPIO_interface.h](#).

5.23.2.13 GPIOx_PIN12

```
#define GPIOx_PIN12 (12)

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

GPIO PIN 12.

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

5.23.2.14 GPIOx_PIN13

```
#define GPIOx_PIN13 (13)

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

GPIO PIN 13.

Definition at line 431 of file [GPIO_interface.h](#).

5.23.2.15 GPIOx_PIN14

```
#define GPIOx_PIN14 (14)

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

GPIO PIN 14.

Definition at line 438 of file [GPIO_interface.h](#).

5.23.2.16 GPIOx_PIN15

```
#define GPIOx_PIN15 (15)

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

GPIO PIN 15.

Definition at line 445 of file [GPIO_interface.h](#).

5.24 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.

5.24.1 Detailed Description

5.24.2 Macro Definition Documentation

5.24.2.1 `GPIOx_AF0`

```
#define GPIOx_AF0 (0)

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

GPIO Alternate function 0.
```

Definition at line 460 of file [GPIO_interface.h](#).

5.24.2.2 `GPIOx_AF1`

```
#define GPIOx_AF1 (1)

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

GPIO Alternate function 1.

Definition at line 467 of file GPIO\_interface.h.
```

5.24.2.3 GPIOx_AF2

```
#define GPIOx_AF2 (2)

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

GPIO Alternate function 2.

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

5.24.2.4 GPIOx_AF3

```
#define GPIOx_AF3 (3)

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

GPIO Alternate function 3.

Definition at line [481](#) of file [GPIO_interface.h](#).

5.24.2.5 GPIOx_AF4

```
#define GPIOx_AF4 (4)

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

GPIO Alternate function 4.

Definition at line [488](#) of file [GPIO_interface.h](#).

5.24.2.6 GPIOx_AF5

```
#define GPIOx_AF5 (5)

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

GPIO Alternate function 5.

Definition at line [495](#) of file [GPIO_interface.h](#).

5.24.2.7 GPIOx_AF6

```
#define GPIOx_AF6 (6)

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

GPIO Alternate function 6.

Definition at line 502 of file [GPIO_interface.h](#).

5.24.2.8 GPIOx_AF7

```
#define GPIOx_AF7 (7)

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

GPIO Alternate function 7.

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

5.24.2.9 GPIOx_AF8

```
#define GPIOx_AF8 (8)

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

GPIO Alternate function 8.

Definition at line 516 of file [GPIO_interface.h](#).

5.24.2.10 GPIOx_AF9

```
#define GPIOx_AF9 (9)

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

GPIO Alternate function 9.

Definition at line 523 of file [GPIO_interface.h](#).

5.24.2.11 GPIOx_AF10

```
#define GPIOx_AF10 (10)

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

GPIO Alternate function 10.

Definition at line 530 of file [GPIO_interface.h](#).

5.24.2.12 GPIOx_AF11

```
#define GPIOx_AF11 (11)

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

GPIO Alternate function 11.

Definition at line 537 of file [GPIO_interface.h](#).

5.24.2.13 GPIOx_AF12

```
#define GPIOx_AF12 (12)

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

GPIO Alternate function 12.

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

5.24.2.14 GPIOx_AF13

```
#define GPIOx_AF13 (13)

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

GPIO Alternate function 13.

Definition at line 551 of file [GPIO_interface.h](#).

5.24.2.15 GPIOx_AF14

```
#define GPIOx_AF14 (14)

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

GPIO Alternate function 14.

Definition at line 558 of file [GPIO_interface.h](#).

5.24.2.16 GPIOx_AF15

```
#define GPIOx_AF15 (15)

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

GPIO Alternate function 15.

Definition at line 565 of file [GPIO_interface.h](#).

5.25 GPIO Addresses

Macros

- #define GPIOA_BASE_ADDRESS (0x40020000)
- #define GPIOB_BASE_ADDRESS (0x40020400)
- #define GPIOC_BASE_ADDRESS (0x40020800)
- #define CLK_SOURCE AHB_DividedBy8

5.25.1 Detailed Description

5.25.2 Macro Definition Documentation

5.25.2.1 GPIOA_BASE_ADDRESS

```
#define GPIOA_BASE_ADDRESS (0x40020000)

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

Port A Base address

5.25.2.2 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](#).

5.25.2.3 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](#).

5.25.2.4 CLK_SOURCE

```
#define CLK_SOURCE AHB_DividedBy8

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

Definition at line 28 of file [SysTick_config.h](#).

5.26 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))

5.26.1 Detailed Description

5.26.2 Macro Definition Documentation

5.26.2.1 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.
```

5.26.2.2 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.
```

5.26.2.3 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.
```

5.27 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

5.27.1 Detailed Description

Interrupt priority grouping values.

Interrupt priority grouping values for PRIGROUP in AIRCR register

See also

[SCB_MemoryMapType::AIRCR](#)

5.27.2 Macro Definition Documentation

5.27.2.1 _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](#).

5.27.2.2 _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](#).

5.27.2.3 _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](#).

5.27.2.4 _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](#).

5.27.2.5 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](#).

5.27.2.6 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](#).

5.27.2.7 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](#).

5.27.2.8 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](#).

5.27.2.9 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](#).

5.27.2.10 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](#).

5.28 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.

5.28.1 Detailed Description

The group priority values.

See also

[MNVIC_vSetPriority](#)

[Sub-Group priorities](#)

5.28.2 Macro Definition Documentation

5.28.2.1 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](#).

5.28.2.2 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](#).

5.28.2.3 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](#).

5.28.2.4 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](#).

5.28.2.5 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](#).

5.28.2.6 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](#).

5.28.2.7 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](#).

5.28.2.8 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](#).

5.28.2.9 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](#).

5.28.2.10 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](#).

5.28.2.11 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](#).

5.28.2.12 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](#).

5.28.2.13 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](#).

5.28.2.14 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](#).

5.28.2.15 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](#).

5.28.2.16 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](#).

5.28.2.17 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](#).

5.29 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_3 (3)`
Priority: 3.
- `#define SUB_PRIORITY_2 (2)`
- `#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.

5.29.1 Detailed Description

The sub-group priority values.

See also

[MNVIC_vSetPriority](#)

[Group priorities](#)

5.29.2 Macro Definition Documentation

5.29.2.1 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](#).

5.29.2.2 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](#).

5.29.2.3 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](#).

5.29.2.4 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](#).

5.29.2.5 SUB_PRIORITY_2

```
#define SUB_PRIORITY_2 (2)

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

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

5.29.2.6 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](#).

5.29.2.7 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](#).

5.29.2.8 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](#).

5.29.2.9 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](#).

5.29.2.10 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](#).

5.29.2.11 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](#).

5.29.2.12 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](#).

5.29.2.13 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](#).

5.29.2.14 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](#).

5.29.2.15 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](#).

5.29.2.16 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](#).

5.29.2.17 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](#).

5.30 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_STREAM5** (68)
DMA2 (Direct Memory Access 2) Stream 3 global interrupt.
- #define **DMA2_STREAM3** (59)
DMA2 (Direct Memory Access 2) Stream 4 global interrupt.
- #define **DMA2_STREAM4** (60)
DMA2 (Direct Memory Access 2) Stream 5 global interrupt.
- #define **OTG_FS** (67)
USB On The Go FS global interrupt.
- #define **DMA2_STREAM6** (69)
DMA2 (Direct Memory Access 2) Stream 6 global interrupt.
- #define **DMA2_STREAM7** (70)
DMA2 (Direct Memory Access 2) Stream 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)`
- `#define FPU (81)`
FPU (Floating Point Unit) global interrupt.
- `#define SPI4 (84)`
SPI4 (Serial Peripheral Interface 4) global interrupt.

5.30.1 Detailed Description

NVIC vector table.

NVIC vector table that contains each peripheral's priority.

The lower the value, the higher the priority

5.30.2 Macro Definition Documentation

5.30.2.1 WWDG

```
#define WWDG (0)

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

Window Watchdog (WWDG) Interrupt.

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

5.30.2.2 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](#).

5.30.2.3 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](#).

5.30.2.4 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](#).

5.30.2.5 FLASH

```
#define FLASH (4)

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

Flash global interrupt.

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

5.30.2.6 RCC

```
#define RCC (5)

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

RCC global interrupt.

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

5.30.2.7 EXTI0

```
#define EXTI0 (6)

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

EXTI Line 0 interrupt.

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

5.30.2.8 EXTI1

```
#define EXTI1 (7)

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

EXTI Line 1 interrupt.

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

5.30.2.9 EXTI2

```
#define EXTI2 (8)

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

EXTI Line 2 interrupt.

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

5.30.2.10 EXTI3

```
#define EXTI3 (9)

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

EXTI Line 3 interrupt.

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

5.30.2.11 EXTI4

```
#define EXTI4 (10)

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

EXTI Line 4 interrupt.

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

5.30.2.12 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](#).

5.30.2.13 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](#).

5.30.2.14 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](#).

5.30.2.15 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](#).

5.30.2.16 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](#).

5.30.2.17 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](#).

5.30.2.18 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](#).

5.30.2.19 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](#).

5.30.2.20 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](#).

5.30.2.21 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](#).

5.30.2.22 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](#).

5.30.2.23 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](#).

5.30.2.24 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](#).

5.30.2.25 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](#).

5.30.2.26 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](#).

5.30.2.27 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](#).

5.30.2.28 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](#).

5.30.2.29 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](#).

5.30.2.30 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](#).

5.30.2.31 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](#).

5.30.2.32 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](#).

5.30.2.33 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](#).

5.30.2.34 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](#).

5.30.2.35 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](#).

5.30.2.36 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](#).

5.30.2.37 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](#).

5.30.2.38 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](#).

5.30.2.39 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](#).

5.30.2.40 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](#).

5.30.2.41 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](#).

5.30.2.42 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](#).

5.30.2.43 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](#).

5.30.2.44 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](#).

5.30.2.45 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](#).

5.30.2.46 DMA2_STREAM5

```
#define DMA2_STREAM5 (68)

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

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

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

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

5.30.2.47 DMA2_STREAM3

```
#define DMA2_STREAM3 (59)

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

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

5.30.2.48 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](#).

5.30.2.49 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](#).

5.30.2.50 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](#).

5.30.2.51 DMA2_STREAM7

```
#define DMA2_STREAM7 (70)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

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

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

5.30.2.52 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](#).

5.30.2.53 I2C3_EV

```
#define I2C3_EV (72)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

I2C3 (Inter-integrated Circuit 3) event interrupt.

I2C3 (Inter-integrated Circuit 3) error interrupt.

5.30.2.54 I2C3_ER

```
#define I2C3_ER (73)  
  
#include <COTS/MCAL/NVIC/NVIC_interface.h>
```

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

5.30.2.55 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](#).

5.30.2.56 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](#).

5.31 NVIC Addresses

NVIC registers base addresses.

Macros

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

5.31.1 Detailed Description

NVIC registers base addresses.

5.31.2 Macro Definition Documentation

5.31.2.1 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](#).

5.31.2.2 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](#).

5.32 NVIC Registers

NVIC registers.

Macros

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

5.32.1 Detailed Description

NVIC registers.

5.32.2 Macro Definition Documentation

5.32.2.1 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](#).

5.32.2.2 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](#).

5.33 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)

5.33.1 Detailed Description

NVIC Settable Priorities for the hardware interrupts.

5.33.2 Macro Definition Documentation

5.33.2.1 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.
```

5.33.2.2 SYSTICK

```
#define SYSTICK (-5)

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

System tick timer

Definition at line 272 of file NVIC\_private.h.
```

5.33.2.3 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.
```

5.33.2.4 MEMORY_MANAGE

```
#define MEMORY_MANAGE (-3)

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

Memory management

Pre-fetch fault, memory access fault

5.33.2.5 BUSFAULT

```
#define BUSFAULT (-2)

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

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

5.33.2.6 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](#).

5.34 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

5.34.1 Detailed Description

5.34.2 Macro Definition Documentation

5.34.2.1 RCC_AHB1

```
#define RCC_AHB1 1

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

AHB1 Bus.
```

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

5.34.2.2 RCC_AHB2

```
#define RCC_AHB2 2

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

AHB2 Bus.
```

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

5.34.2.3 RCC_APB1

```
#define RCC_APB1 3

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

APB1 Bus.
```

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

5.34.2.4 RCC_APB2

```
#define RCC_APB2 4

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

APB2 Bus.
```

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

5.34.2.5 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.
```

5.35 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.

5.35.1 Detailed Description

5.35.2 Macro Definition Documentation

5.35.2.1 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.
```

5.35.2.2 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](#).

5.35.2.3 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](#).

5.35.2.4 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](#).

5.35.2.5 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](#).

5.35.2.6 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](#).

5.35.2.7 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](#).

5.35.2.8 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](#).

5.35.2.9 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](#).

5.35.2.10 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](#).

5.35.2.11 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](#).

5.35.2.12 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](#).

5.35.2.13 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](#).

5.35.2.14 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](#).

5.35.2.15 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](#).

5.35.2.16 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](#).

5.35.2.17 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](#).

5.35.2.18 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](#).

5.35.2.19 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](#).

5.35.2.20 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](#).

5.35.2.21 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](#).

5.35.2.22 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](#).

5.35.2.23 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](#).

5.35.2.24 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](#).

5.35.2.25 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](#).

5.35.2.26 APB2ENR_SYSCFGEN

```
#define APB2ENR_SYSCFGEN 14  
  
#include <COTS/MCAL/RCC/MRCC_interface.h>
```

Enable CLK on SYSCFG peripheral.

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

5.35.2.27 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](#).

5.35.2.28 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](#).

5.35.2.29 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](#).

5.35.2.30 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](#).

5.35.2.31 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](#).

5.35.2.32 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](#).

5.35.2.33 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](#).

5.35.2.34 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](#).

5.36 Systick Interval Mode Configuration

Macros

- `#define SINGLE_INTERVAL_MODE (1)`
Single interval mode.
- `#define PERIODIC_INTERVAL_MODE (2)`
Periodic interval mode.

5.36.1 Detailed Description

5.36.2 Macro Definition Documentation

5.36.2.1 SINGLE_INTERVAL_MODE

```
#define SINGLE_INTERVAL_MODE (1)

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

Single interval mode.

Definition at line 52 of file [SysTick_config.h](#).

5.36.2.2 PERIODIC_INTERVAL_MODE

```
#define PERIODIC_INTERVAL_MODE (2)

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

Periodic interval mode.

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

5.37 Delay Units

Macros

- `#define MILLI_SEC (1)`
Delay in milli seconds.
- `#define MICRO_SEC (2)`
- `#define SEC (3)`
Delay in seconds.

5.37.1 Detailed Description

5.37.2 Macro Definition Documentation

5.37.2.1 MILLI_SEC

```
#define MILLI_SEC (1)

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

Delay in milli seconds.
```

Delay in micro seconds.

5.37.2.2 MICRO_SEC

```
#define MICRO_SEC (2)

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

Definition at line 143 of file SysTick_interface.h.
```

5.37.2.3 SEC

```
#define SEC (3)

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

Delay in seconds.

Definition at line 149 of file SysTick_interface.h.
```

5.38 Systick Addresses

Macros

- `#define SysTick_BASE_ADDRESS (0xE000E010)`

5.38.1 Detailed Description

5.38.2 Macro Definition Documentation

5.38.2.1 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](#).

5.39 Systick Registers

Macros

- `#define SysTick ((volatile P2VAR(SysTick_Type))(SysTick_BASE_ADDRESS))`

5.39.1 Detailed Description

5.39.2 Macro Definition Documentation

5.39.2.1 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](#).

5.40 Systick Register Bits Positions

Macros

- #define COUNTFLAG (16)
- #define CLKSOURCE (2)
- #define TICKINT (1)
- #define COUNTER_ENABLE (0)

5.40.1 Detailed Description

5.40.2 Macro Definition Documentation

5.40.2.1 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](#).

5.40.2.2 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](#).

5.40.2.3 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](#).

5.40.2.4 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 countdown. 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](#).

5.41 Systick Clock Sources

Macros

- #define [AHB_DividedBy8](#) (1)
- #define [AHB](#) (2)

5.41.1 Detailed Description

5.41.2 Macro Definition Documentation

5.41.2.1 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](#).

5.41.2.2 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](#).

5.42 Systick Exception (Interrupt) Status

Macros

- #define Dont AssertRequest (1)
- #define AssertRequest (2)

5.42.1 Detailed Description

5.42.2 Macro Definition Documentation

5.42.2.1 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](#).

5.42.2.2 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](#).

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 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.

6.2.1 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](#).

6.2.2 Field Documentation

6.2.2.1 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\(\)](#).

6.2.2.2 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\(\)](#).

6.2.2.3 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\(\)](#).

6.2.2.4 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\(\)](#).

The documentation for this struct was generated from the following file:

- COTS/HAL/DCMOTOR/[DCM_interface.h](#)

6.3 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.

6.3.1 Detailed Description

Interrupt configuration structure to initialize the interrupt with.

Definition at line 18 of file [EXTI_interface.h](#).

6.3.2 Field Documentation

6.3.2.1 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\(\)](#).

6.3.2.2 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\(\)](#).

6.3.2.3 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](#)

6.4 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.

6.4.1 Detailed Description

EXTI Configuration structure for interrupt initialization process.

Definition at line 18 of file [EXTI_private.h](#).

6.4.2 Field Documentation

6.4.2.1 IMR

`u32_t IMR`

Interrupt mask register.

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

6.4.2.2 EMR

`u32_t EMR`

Event mask register.

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

6.4.2.3 RTSR

`u32_t` RTSR

Rising trigger status register.

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

6.4.2.4 FTSR

`u32_t` FTSR

Falling trigger status register.

Definition at line 35 of file [EXTI_private.h](#).

6.4.2.5 SWIER

`u32_t` SWIER

Software interrupt event register.

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

6.4.2.6 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](#)

6.5 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.*

6.5.1 Detailed Description

GPIO declaration structure for its registers.

Definition at line 18 of file [GPIO_private.h](#).

6.5.2 Field Documentation

6.5.2.1 MODERx

`u32_t MODERx`

Control PIN mode whether **INPUT** or **OUTPUT**

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

6.5.2.2 OTYPERx

`u32_t OTYPERx`

Select the output type between push-pull or open-drain.

Definition at line 28 of file [GPIO_private.h](#).

6.5.2.3 OSPEEDRx

`u32_t OSPEEDRx`

Configure the speed that the PIN is connected to.

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

6.5.2.4 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](#).

6.5.2.5 IDRx

`u32_t IDRx`

Read the input data.

Definition at line 43 of file [GPIO_private.h](#).

6.5.2.6 ODRx

`u32_t ODRx`

Write the output data.

Definition at line 48 of file [GPIO_private.h](#).

6.5.2.7 BSRRx

`u32_t BSRRx`

(Re)set each bit in the output data register

Definition at line 53 of file [GPIO_private.h](#).

6.5.2.8 LCKRx

`u32_t LCKRx`

Lock the GPIO control registers.

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

6.5.2.9 AFRLx

`u32_t AFRLx`

Control alternate function **LOW** register.

Definition at line 63 of file [GPIO_private.h](#).

6.5.2.10 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](#)

6.6 MADC_ConfigType Struct Reference

```
#include "COTS/MCAL/ADC/ADC_interface.h"
```

Data Fields

- `u8_t Port`

6.6.1 Detailed Description

Definition at line 18 of file [ADC_interface.h](#).

6.6.2 Field Documentation

6.6.2.1 Port

`u8_t Port`

Definition at line 22 of file [ADC_interface.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/ADC/[ADC_interface.h](#)

6.7 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.

6.7.1 Detailed Description

MDIO Configuration structure for a specific PIN initialization.

Definition at line 18 of file [GPIO_interface.h](#).

6.7.2 Field Documentation

6.7.2.1 Port

`u8_t Port`

Configures a specific GPIO port.

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

Referenced by [HLCD_vInit\(\)](#), [MGPIOx_vInit\(\)](#), [MSPI_vInit\(\)](#), and [MUSART_vInit\(\)](#).

6.7.2.2 Pin

`u8_t Pin`

Configures a specific GPIO pin.

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

Referenced by [MGPIOx_vInit\(\)](#).

6.7.2.3 Mode

`u8_t Mode`

Configures a specific GPIO mode.

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

Referenced by [MGPIOx_vInit\(\)](#).

6.7.2.4 OutputType

`u8_t OutputType`

Configures a specific GPIO output's type.

Definition at line 35 of file [GPIO_interface.h](#).

Referenced by [MGPIOx_vInit\(\)](#).

6.7.2.5 OutputSpeed

`u8_t` `OutputSpeed`

Configures a specific GPIO output's speed.

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

Referenced by [MGPIOx_vInit\(\)](#).

6.7.2.6 InputType

`u8_t` `InputType`

Configures a specific GPIO input type.

Definition at line 43 of file [GPIO_interface.h](#).

Referenced by [MGPIOx_vInit\(\)](#).

6.7.2.7 AF_Type

`u8_t` `AF_Type`

Configures a specific GPIO Alternative function.

Definition at line 47 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](#)

6.8 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.

6.8.1 Detailed Description

System configuration structure to initialize the SYSCFG module with.

Definition at line 19 of file [SYSCFG_private.h](#).

6.8.2 Field Documentation

6.8.2.1 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](#).

6.8.2.2 PMC

`u32_t` `PMC`

Peripheral mode configuration register.

Definition at line 29 of file [SYSCFG_private.h](#).

6.8.2.3 EXTICR

`u32_t` `EXTICR [4]`

External interrupt configuration 1-4 registers.

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

6.8.2.4 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](#)

6.9 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.

6.9.1 Detailed Description

NVIC configuration structure for NVIC memory map.

Definition at line 18 of file [NVIC_private.h](#).

6.9.2 Field Documentation

6.9.2.1 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](#).

6.9.2.2 RESERVED0x

`u32_t RESERVED0x[24]`

Note

This is a reserved register

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

6.9.2.3 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](#).

6.9.2.4 RSERVED1x

`u32_t RSERVED1x[24]`

Note

This is a reserved register

Definition at line 43 of file [NVIC_private.h](#).

6.9.2.5 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](#).

6.9.2.6 RESERVED2x

`u32_t RESERVED2x[24]`

Note

This is a reserved register

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

6.9.2.7 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](#).

6.9.2.8 RESERVED3x

`u32_t RESERVED3x [24]`

Note

This is a reserved register

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

6.9.2.9 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](#).

6.9.2.10 RESERVED4x

`u32_t` RESERVED4x[56]

Note

This is a reserved register

Definition at line 81 of file [NVIC_private.h](#).

6.9.2.11 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](#)

6.10 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 PLLCFG**
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`
APB1 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.

6.10.1 Detailed Description

RCC declaration structure for its registers.

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

6.10.2 Field Documentation

6.10.2.1 CR

`u32_t CR`

clock control register

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

6.10.2.2 PLLCFG

`u32_t PLLCFG`

PLL configuration register.

Definition at line 35 of file [MRCC_private.h](#).

6.10.2.3 CFGR

`u32_t` CFGR

clock configuration register

Definition at line 40 of file [MRCC_private.h](#).

6.10.2.4 CIR

`u32_t` CIR

clock interrupt register

Definition at line 45 of file [MRCC_private.h](#).

6.10.2.5 AHB1RSTR

`u32_t` AHB1RSTR

AHB1 peripheral reset register.

Definition at line 50 of file [MRCC_private.h](#).

6.10.2.6 AHB2RSTR

`u32_t` AHB2RSTR

AHB2 peripheral reset register.

Definition at line 55 of file [MRCC_private.h](#).

6.10.2.7 Reserved1

`u32_t` Reserved1

Reserved register.

Definition at line 60 of file [MRCC_private.h](#).

6.10.2.8 Reserved2

`u32_t` Reserved2

Reserved register.

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

6.10.2.9 APB1RSTR

`u32_t` APB1RSTR

APB1 peripheral reset register.

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

6.10.2.10 APB2RSTR

`u32_t` APB2RSTR

APB2 peripheral reset register.

Definition at line 75 of file [MRCC_private.h](#).

6.10.2.11 Reserved3

`u32_t` Reserved3

Reserved register.

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

6.10.2.12 Reserved4

`u32_t` Reserved4

Reserved register.

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

6.10.2.13 AHB1ENR

`u32_t` AHB1ENR

AHB1 peripheral clock enable register.

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

6.10.2.14 AHB2ENR

`u32_t` AHB2ENR

AHB2 peripheral clock enable register.

Definition at line 95 of file [MRCC_private.h](#).

6.10.2.15 Reserved5

`u32_t` Reserved5

Reserved register.

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

6.10.2.16 Reserved6

`u32_t` Reserved6

Reserved register.

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

6.10.2.17 APB1ENR

`u32_t` APB1ENR

APB1 peripheral clock enable register.

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

6.10.2.18 APB2ENR

`u32_t` APB2ENR

APB2 peripheral clock enable register.

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

6.10.2.19 Reserved7

`u32_t` Reserved7

Reserved register.

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

6.10.2.20 Reserved8

`u32_t` Reserved8

Reserved register.

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

6.10.2.21 AHB1LPENR

`u32_t` AHB1LPENR

AHB1 peripheral clock enable in low power mode register.

Definition at line 130 of file [MRCC_private.h](#).

6.10.2.22 AHB2LPENR

`u32_t` AHB2LPENR

AHB2 peripheral clock enable in low power mode register.

Definition at line 135 of file [MRCC_private.h](#).

6.10.2.23 Reserved9

`u32_t` Reserved9

Reserved register.

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

6.10.2.24 Reserved10

`u32_t` Reserved10

Reserved register.

Definition at line 145 of file [MRCC_private.h](#).

6.10.2.25 APB1LPENR

`u32_t` APB1LPENR

APB1 peripheral clock enable in low power mode register.

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

6.10.2.26 APB2LPENR

`u32_t` APB2LPENR

APB2 peripheral clock enable in low power mode register.

Definition at line 155 of file [MRCC_private.h](#).

6.10.2.27 Reserved11

`u32_t` Reserved11

Reserved register.

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

6.10.2.28 Reserved12

`u32_t` Reserved12

Reserved register.

Definition at line 165 of file [MRCC_private.h](#).

6.10.2.29 BDCR

`u32_t` BDCR

RCC Backup domain control register.

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

6.10.2.30 CSR

`u32_t` CSR

clock control & status register

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

6.10.2.31 Reserved13

`u32_t` Reserved13

Reserved register.

Definition at line 180 of file [MRCC_private.h](#).

6.10.2.32 Reserved14

`u32_t` Reserved14

Reserved register.

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

6.10.2.33 SSCGR

`u32_t` `SSCGR`

RCC spread spectrum clock generation register.

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

6.10.2.34 PLLI2SCFGR

`u32_t` `PLLI2SCFGR`

PLLI2S configuration register.

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

6.10.2.35 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](#)

6.11 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.

6.11.1 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](#).

6.11.2 Field Documentation

6.11.2.1 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](#).

6.11.2.2 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](#).

6.11.2.3 VTOR

`u32_t VTOR`

Vector table offset register.

Interrupt control and state register

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

6.11.2.4 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](#).

6.11.2.5 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](#).

6.11.2.6 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](#).

6.11.2.7 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](#).

6.11.2.8 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](#).

6.11.2.9 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](#).

6.11.2.10 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](#).

6.11.2.11 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](#).

6.11.2.12 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](#).

6.11.2.13 RESERVED

`u32_t` RESERVED

Note

This is a reserved register

Definition at line 187 of file [NVIC_private.h](#).

6.11.2.14 MMFAR

`u32_t` MMFAR

Memory management fault address register.

Memory management fault address register

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

6.11.2.15 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](#)

6.12 SPI_MemoryMapType Struct Reference

```
#include "COTS/MCAL/SPI/SPI_interface.h"
```

Data Fields

- volatile `u32_t` CR1
- volatile `u32_t` CR2
- volatile `u32_t` SR
- volatile `u32_t` DR
- volatile `u32_t` CRCPR
- volatile `u32_t` RXCRCR
- volatile `u32_t` TXCRCR
- volatile `u32_t` I2SCFGR
- volatile `u32_t` I2SPR

6.12.1 Detailed Description

Definition at line 18 of file [SPI_interface.h](#).

6.12.2 Field Documentation

6.12.2.1 CR1

```
volatile u32_t CR1
```

Definition at line 21 of file [SPI_interface.h](#).

Referenced by [MSPI_vDISABLE\(\)](#), and [MSPI_vInit\(\)](#).

6.12.2.2 CR2

```
volatile u32_t CR2
```

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

Referenced by [MSPI_vInit\(\)](#).

6.12.2.3 SR

```
volatile u32_t SR
```

Definition at line 25 of file [SPI_interface.h](#).

Referenced by [MSPI_u16MasterRecieve\(\)](#), [MSPI_u16SlaveRecieve\(\)](#), [MSPI_u16Transcieve\(\)](#), [MSPI_vDISABLE\(\)](#), [MSPI_vMasterTransmit\(\)](#), and [MSPI_vSlaveTransmit\(\)](#).

6.12.2.4 DR

```
volatile u32_t DR
```

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

Referenced by [MSPI_u16MasterRecieve\(\)](#), [MSPI_u16SlaveRecieve\(\)](#), and [MSPI_u16Transcieve\(\)](#).

6.12.2.5 CRCPR

```
volatile u32_t CRCPR
```

Definition at line 29 of file [SPI_interface.h](#).

6.12.2.6 RXCRCR

```
volatile u32_t RXCRCR
```

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

6.12.2.7 TXCRCR

```
volatile u32_t TXCRCR
```

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

6.12.2.8 I2SCFGR

```
volatile u32_t I2SCFGR
```

Definition at line 35 of file [SPI_interface.h](#).

6.12.2.9 I2SPR

```
volatile u32_t I2SPR
```

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

The documentation for this struct was generated from the following file:

- COTS/MCAL/SPI/SPI_interface.h

6.13 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.

6.13.1 Detailed Description

Systick memory map structure declaration for the Systick's registers.

Definition at line 18 of file [SysTick_private.h](#).

6.13.2 Field Documentation

6.13.2.1 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](#).

6.13.2.2 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](#).

6.13.2.3 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](#).

6.13.2.4 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](#)

6.14 USART_ClockInitTypeDef Struct Reference

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- `u8_t ClockOutput`
- `u8_t ClockPolarity`
- `u8_t ClockPhase`
- `u8_t LastBitClockPulse`

6.14.1 Detailed Description

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

6.14.2 Field Documentation

6.14.2.1 ClockOutput

`u8_t` `ClockOutput`

Definition at line 67 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.14.2.2 ClockPolarity

`u8_t ClockPolarity`

Definition at line 69 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.14.2.3 ClockPhase

`u8_t ClockPhase`

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

Referenced by [MUSART_vInit\(\)](#).

6.14.2.4 LastBitClockPulse

`u8_t LastBitClockPulse`

Definition at line 73 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](#)

6.15 USART_InitType Struct Reference

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- `u32_t BaudRate`
- `u8_t DataWidth`
- `u8_t StopBits`
- `u8_t Parity_Enable`
- `u8_t Parity_Selection`
- `u8_t TransferDirection`
- `u8_t HardwareFlowControl`
- `u8_t Oversampling`

6.15.1 Detailed Description

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

6.15.2 Field Documentation

6.15.2.1 BaudRate

`u32_t` BaudRate

Definition at line 46 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.15.2.2 DataWidth

`u8_t` DataWidth

Definition at line 48 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.15.2.3 StopBits

`u8_t` StopBits

Definition at line 50 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.15.2.4 Parity_Enable

`u8_t` Parity_Enable

Definition at line 52 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.15.2.5 Parity_Selection

`u8_t` Parity_Selection

Definition at line 54 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.15.2.6 TransferDirection

`u8_t` TransferDirection

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

Referenced by [MUSART_vInit\(\)](#).

6.15.2.7 HardwareFlowControl

`u8_t` HardwareFlowControl

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

6.15.2.8 Oversampling

`u8_t` Oversampling

Definition at line 60 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](#)

6.16 USART_MemoryMapType Struct Reference

```
#include "COTS/MCAL/UART/UART_interface.h"
```

Data Fields

- volatile u32_t SR_REG
- volatile u32_t DR_REG
- volatile u32_t BRR_REG
- volatile u32_t CR1_REG
- volatile u32_t CR2_REG
- volatile u32_t CR3_REG
- volatile u32_t GTPR_REG

6.16.1 Detailed Description

Definition at line 19 of file [UART_interface.h](#).

6.16.2 Field Documentation

6.16.2.1 SR_REG

```
volatile u32_t SR_REG
```

Definition at line 22 of file [UART_interface.h](#).

Referenced by [MUSART_u8ReceiveByteSynchBlocking\(\)](#), [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#), [MUSART_vClearFlags\(\)](#), [MUSART_vInit\(\)](#), and [MUSART_vTransmitByte\(\)](#).

6.16.2.2 DR_REG

```
volatile u32_t DR_REG
```

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

Referenced by [MUSART_u8ReadDataRegister\(\)](#), [MUSART_u8ReceiveByteSynchBlocking\(\)](#), [MUSART_u8ReceiveByteSynchNonBlo](#) and [MUSART_vTransmitByte\(\)](#).

6.16.2.3 BRR_REG

```
volatile u32_t BRR_REG
```

Definition at line 26 of file [UART_interface.h](#).

Referenced by [MUSART_vInit\(\)](#).

6.16.2.4 CR1_REG

```
volatile u32_t CR1_REG
```

Definition at line 28 of file [UART_interface.h](#).

Referenced by [MUSART_vDisable\(\)](#), [MUSART_vEnable\(\)](#), [MUSART_vInit\(\)](#), and [MUSART_vRxIntSetStatus\(\)](#).

6.16.2.5 CR2_REG

```
volatile u32_t CR2_REG
```

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

Referenced by [MUSART_vInit\(\)](#).

6.16.2.6 CR3_REG

```
volatile u32_t CR3_REG
```

Definition at line 32 of file [UART_interface.h](#).

6.16.2.7 GTPR_REG

```
volatile u32_t GTPR_REG
```

Definition at line 34 of file [UART_interface.h](#).

The documentation for this struct was generated from the following file:

- COTS/MCAL/UART/[UART_interface.h](#)

6.17 UserInfo_t Struct Reference

```
#include "COTS/Services/UsersLogin/UsersLogin_private.h"
```

Data Fields

- [c8_t strUsername \[\(10\)\]](#)
- [c8_t strPassword \[\(10\)\]](#)

6.17.1 Detailed Description

Definition at line [6](#) of file [UsersLogin_private.h](#).

6.17.2 Field Documentation

6.17.2.1 strUsername

`c8_t strUsername[(10)]`

Definition at line [8](#) of file [UsersLogin_private.h](#).

6.17.2.2 strPassword

`c8_t strPassword[(10)]`

Definition at line [9](#) of file [UsersLogin_private.h](#).

The documentation for this struct was generated from the following file:

- COTS/Services/UsersLogin/[UsersLogin_private.h](#)

Chapter 7

File Documentation

7.1 D:/GProject/ADAS_Project/README.md File Reference

7.2 COTS/HAL/Bluetooth/Bluetooth_config.c File Reference

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../MCAL/RCC/MRCC_interface.h"
#include "../../MCAL/GPIO/GPIO_interface.h"
#include "../../MCAL/UART/UART_interface.h"
#include "Bluetooth_config.h"
```

Variables

- `USART_InitType Bluetooth_Config`
- `USART_ClockInitTypeDef Bluetooth_CLK = { 0 , 0, 0, 0 }`
- `USART_MemoryMapType * Bluetooth_UART_ID = ((USART_MemoryMapType*) 0x40011000)`

7.2.1 Variable Documentation

7.2.1.1 Bluetooth_Config

`USART_InitType` `Bluetooth_Config`

Initial value:

```
=
{
    .BaudRate      =  9600 ,     .DataWidth      =  0 ,
    .StopBits       =  0 ,     .Parity_Enable   =  0 ,
    .Parity_Selection =  0 ,     .TransferDirection =  2 ,
    .HardwareFlowControl =  0 ,     .Oversampling   =  0
}
```

Definition at line 26 of file `Bluetooth_config.c`.

Referenced by `HBluetooth_vInit()`.

7.2.1.2 Bluetooth_CLK

```
USART_ClockInitTypeDef Bluetooth_CLK = { 0 , 0, 0, 0 }
```

Definition at line 37 of file [Bluetooth_config.c](#).

Referenced by [HBluetooth_vInit\(\)](#).

7.2.1.3 Bluetooth_UART_ID

```
USART_MemoryMapType* Bluetooth_UART_ID = ( ( USART_MemoryMapType*) 0x40011000 )
```

Definition at line 42 of file [Bluetooth_config.c](#).

Referenced by [HBluetooth_u8ReceiveByte\(\)](#), [HBluetooth_vDisable\(\)](#), [HBluetooth_vEnable\(\)](#), [HBluetooth_vInit\(\)](#), [HBluetooth_vReceiveString\(\)](#), [HBluetooth_vSendByte\(\)](#), and [HBluetooth_vSendString\(\)](#).

7.3 Bluetooth_config.c

[Go to the documentation of this file.](#)

```
00001 /*
00002 * Bluetooth_config.c
00003 *
00004 * Created on: Jan 5, 2023
00005 * Author: Ali El Bana
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/GPIO/GPIO_interface.h"
00018 #include "../../MCAL/UART/UART_interface.h"
00019
00020 #include "Bluetooth_config.h"
00021
00022 /*****
00023 *           Configuration parameters
00024 *****/
00025
00026 USART_InitTypeDef Bluetooth_Config =
00027 {
00028     .BaudRate      = __BAUDRATE__ , .DataWidth      = MODE_8BIT ,
00029     .StopBits       = STOP_BIT_1   , .Parity_Enable   = DISABLE ,
00030     .Parity_Selection = EVEN_PARITY , .TransferDirection = TX_RX ,
00031     .HardwareFlowControl = DISABLE , .Oversampling    = OVER_SAMPLING_16
00032 };
00033
00034 USART_ClockInitTypeDef Bluetooth_CLK = { DISABLE, 0, 0, 0 } ;
00035
00036
00037 USART_MemoryMapType * Bluetooth_UART_ID = USART1_REG ;
00038
00039
00040
00041 USART_MemoryMapType * Bluetooth_UART_ID = USART1_REG ;
00042
00043
00044
00045
00046
00047
```

7.4 COTS/HAL/Bluetooth/Bluetooth_config.h File Reference

7.5 Bluetooth_config.h

[Go to the documentation of this file.](#)

```
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
```

7.6 COTS/HAL/Bluetooth/Bluetooth_interface.h File Reference

Functions

- void [HBluetooth_vInit](#) (void)
- void [HBluetooth_vSendByte](#) ([VAR\(u8_t\)](#) A_u8Byte)
- [u8_t](#) [HBluetooth_u8ReceiveByte](#) (void)
- void [HBluetooth_vSendString](#) ([P2VAR\(c8_t\)](#) A_ptrc8String)
- void [HBluetooth_vReceiveString](#) ([P2VAR\(c8_t\)](#) A_c8YourString)
- [u8_t](#) [HBluetooth_u8CompStrings](#) ([P2VAR\(c8_t\)](#) String1, [P2VAR\(c8_t\)](#) String2)
- void [HBluetooth_vEnable](#) (void)
- void [HBluetooth_vDisable](#) (void)
- void [HBluetooth_ExchangeString](#) ([P2VAR\(c8_t\)](#) L_strMessagePlaceholder, [P2VAR\(c8_t\)](#) L_strInfo)
- [u8_t](#) [HBluetooth_ExchangeByte](#) ([P2VAR\(c8_t\)](#) L_strMessagePlaceholder)

7.6.1 Function Documentation

7.6.1.1 HBluetooth_vInit()

```
void HBluetooth_vInit (
    void )
```

Definition at line 38 of file [Bluetooth_program.c](#).

```
00039 {
00040
00041     MUSART_vInit( &Bluetooth_Config, &Bluetooth_CLK, Bluetooth_UART_ID ) ;
00042
00043 }
```

References [Bluetooth_CLK](#), [Bluetooth_Config](#), [Bluetooth_UART_ID](#), and [MUSART_vInit\(\)](#).

7.6.1.2 HBluetooth_vSendByte()

```
void HBluetooth_vSendByte (
    VAR(u8_t) A_u8Byte )
```

7.6.1.3 HBluetooth_u8ReceiveByte()

```
u8_t HBluetooth_u8ReceiveByte (
    void )
```

Definition at line 58 of file [Bluetooth_program.c](#).

```
00059 {
00060
00061     u8_t L_u8ReceivedByte = INITIAL_ZERO ;
00062
00063     L_u8ReceivedByte = MUSART_u8ReceiveByteSynchBlocking( Bluetooth_UART_ID ) ;
00064
00065     return L_u8ReceivedByte ;
00066
00067 }
```

References [Bluetooth_UART_ID](#), [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSynchBlocking\(\)](#).

Referenced by [HBluetooth_ExchangeByte\(\)](#).

7.6.1.4 HBluetooth_vSendString()

```
void HBluetooth_vSendString (
    P2VAR(c8_t) A_ptrc8String )
```

Referenced by [UsersLogin_vUserLoginProcess\(\)](#).

7.6.1.5 HBluetooth_vReceiveString()

```
void HBluetooth_vReceiveString (
    P2VAR(c8_t) A_c8YourString )
```

7.6.1.6 HBluetooth_u8CompStrings()

```
u8_t HBluetooth_u8CompStrings (
    P2VAR(c8_t) String1,
    P2VAR(c8_t) String2 )
```

7.6.1.7 HBluetooth_vEnable()

```
void HBluetooth_vEnable (
    void )
```

Definition at line 106 of file [Bluetooth_program.c](#).

```
00107 {
00108     MUSART_vEnable( Bluetooth_UART_ID ) ;
00110
00111 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vEnable\(\)](#).

7.6.1.8 HBluetooth_vDisable()

```
void HBluetooth_vDisable (
    void )
```

Definition at line 116 of file [Bluetooth_program.c](#).

```
00117 {
00118     MUSART_vDisable( Bluetooth_UART_ID ) ;
00120
00121 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vDisable\(\)](#).

7.6.1.9 HBluetooth_ExchangeString()

```
void HBluetooth_ExchangeString (
    P2VAR(c8_t) L_strMessagePlaceholder,
    P2VAR(c8_t) L_strInfo )
```

Referenced by [UsersLogin_vUserLoginProcess\(\)](#).

7.6.1.10 HBluetooth_ExchangeByte()

```
u8_t HBluetooth_ExchangeByte (
    P2VAR(c8_t) L_strMessagePlaceholder )
```

7.7 Bluetooth_interface.h

[Go to the documentation of this file.](#)

```
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 /***** Functions prototypes ****/
00010 /*
00011 *****/
00012
00013 void HBluetooth_vInit(void);
00014
00015 void HBluetooth_vSendByte(VAR(u8_t) A_u8Byte);
00016
00017 u8_t HBluetooth_u8ReceiveByte(void);
00018
00019 void HBluetooth_vSendString(P2VAR(c8_t) A_ptrc8String);
00020
00021 void HBluetooth_vReceiveString(P2VAR(c8_t) A_c8YourString);
00022
00023 u8_t HBluetooth_u8CompStrings(P2VAR(c8_t) String1, P2VAR(c8_t) String2);
00024
00025 void HBluetooth_vEnable(void);
00026
00027 void HBluetooth_vDisable(void);
00028
00029 void HBluetooth_ExchangeString(P2VAR(c8_t) L_strMessagePlaceholder, P2VAR(c8_t) L_strInfo);
00030 u8_t HBluetooth_ExchangeByte(P2VAR(c8_t) L_strMessagePlaceholder);
00031
00032 #endif // _Bluetooth_interface_H
```

7.8 COTS/HAL/Bluetooth/Bluetooth_private.h File Reference

7.9 Bluetooth_private.h

[Go to the documentation of this file.](#)

```
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
```

7.10 COTS/HAL/Bluetooth/Bluetooth_program.c File Reference

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../MCAL/RCC/MRCC_interface.h"
#include "../../MCAL/GPIO/GPIO_interface.h"
#include "../../MCAL/UART/UART_interface.h"
#include "Bluetooth_interface.h"
#include "Bluetooth_private.h"
#include "Bluetooth_config.h"
```

Functions

- void [HBluetooth_vInit](#) (void)
- void [HBluetooth_vSendByte](#) ([u8_t](#) A_u8Byte)
- [u8_t](#) [HBluetooth_u8ReceiveByte](#) (void)
- void [HBluetooth_vSendString](#) ([c8_t](#) *A_ptrc8String)
- void [HBluetooth_vReceiveString](#) ([c8_t](#) *A_c8YourString)
- [u8_t](#) [HBluetooth_u8CompStrings](#) ([c8_t](#) *String1, [c8_t](#) *String2)
- void [HBluetooth_vEnable](#) (void)
- void [HBluetooth_vDisable](#) (void)
- void [HBluetooth_ExchangeString](#) ([c8_t](#) *L_strMessagePlaceholder, [c8_t](#) *L_strInfo)
- [u8_t](#) [HBluetooth_ExchangeByte](#) ([c8_t](#) *L_strMessagePlaceholder)

Variables

- [USART_InitType](#) Bluetooth_Config
- [USART_ClockInitTypeDef](#) Bluetooth_CLK
- [USART_MemoryMapType](#) * Bluetooth_UART_ID

7.10.1 Function Documentation

7.10.1.1 HBluetooth_vInit()

```
void HBluetooth_vInit (
    void )
```

Definition at line 38 of file [Bluetooth_program.c](#).

```
00039 {
00040
00041     MUSART_vInit( &Bluetooth_Config, &Bluetooth_CLK, Bluetooth_UART_ID ) ;
00042
00043 }
```

References [Bluetooth_CLK](#), [Bluetooth_Config](#), [Bluetooth_UART_ID](#), and [MUSART_vInit\(\)](#).

7.10.1.2 HBluetooth_vSendByte()

```
void HBluetooth_vSendByte (
    u8\_t A_u8Byte )
```

Definition at line 48 of file [Bluetooth_program.c](#).

```
00049 {
00050
00051     MUSART_vTransmitByte( Bluetooth_UART_ID, A_u8Byte ) ;
00052
00053 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vTransmitByte\(\)](#).

7.10.1.3 HBluetooth_u8ReceiveByte()

```
u8_t HBluetooth_u8ReceiveByte (
    void )
```

Definition at line 58 of file [Bluetooth_program.c](#).

```
00059 {
00060
00061     u8_t L_u8ReceivedByte = INITIAL_ZERO ;
00062
00063     L_u8ReceivedByte = MUSART_u8ReceiveByteSyncBlocking( Bluetooth_UART_ID ) ;
00064
00065     return L_u8ReceivedByte ;
00066
00067 }
```

References [Bluetooth_UART_ID](#), [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSyncBlocking\(\)](#).

Referenced by [HBluetooth_ExchangeByte\(\)](#).

7.10.1.4 HBluetooth_vSendString()

```
void HBluetooth_vSendString (
    c8_t * A_ptrc8String )
```

Definition at line 72 of file [Bluetooth_program.c](#).

```
00073 {
00074
00075     MUSART_vTransmitString( Bluetooth_UART_ID, A_ptrc8String ) ;
00076
00077 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vTransmitString\(\)](#).

Referenced by [HBluetooth_ExchangeByte\(\)](#), and [HBluetooth_ExchangeString\(\)](#).

7.10.1.5 HBluetooth_vReceiveString()

```
void HBluetooth_vReceiveString (
    c8_t * A_c8YourString )
```

Definition at line 82 of file [Bluetooth_program.c](#).

```
00083 {
00084
00085     MUSART_vRecieveString( Bluetooth_UART_ID, A_c8YourString ) ;
00086
00087 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vRecieveString\(\)](#).

Referenced by [HBluetooth_ExchangeString\(\)](#).

7.10.1.6 HBluetooth_u8CompStrings()

```
u8_t HBluetooth_u8CompStrings (
    c8_t * String1,
    c8_t * String2 )
```

Definition at line 92 of file [Bluetooth_program.c](#).

```
00093 {
00094     u8_t L_u8Result = INITIAL_ZERO ;
00095
00096     L_u8Result = MUSART_u8CompareString( String1, String2 ) ;
00097
00098     return L_u8Result ;
00099
00100
00101 }
```

References [INITIAL_ZERO](#), and [MUSART_u8CompareString\(\)](#).

7.10.1.7 HBluetooth_vEnable()

```
void HBluetooth_vEnable (
    void )
```

Definition at line 106 of file [Bluetooth_program.c](#).

```
00107 {
00108     MUSART_vEnable( Bluetooth_UART_ID ) ;
00109
00110
00111 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vEnable\(\)](#).

7.10.1.8 HBluetooth_vDisable()

```
void HBluetooth_vDisable (
    void )
```

Definition at line 116 of file [Bluetooth_program.c](#).

```
00117 {
00118     MUSART_vDisable( Bluetooth_UART_ID ) ;
00119
00120
00121 }
```

References [Bluetooth_UART_ID](#), and [MUSART_vDisable\(\)](#).

7.10.1.9 HBluetooth_ExchangeString()

```
void HBluetooth_ExchangeString (
    c8_t * L_strMessagePlaceholder,
    c8_t * L_strInfo )
```

Definition at line 126 of file [Bluetooth_program.c](#).

```
00127 {
00128     HBluetooth_vSendString(L_strMessagePlaceholder);
00129     HBluetooth_vReceiveString(L_strInfo);
00130 }
```

References [HBluetooth_vReceiveString\(\)](#), and [HBluetooth_vSendString\(\)](#).

7.10.1.10 HBluetooth_ExchangeByte()

```
u8_t HBluetooth_ExchangeByte (
    c8_t * L_strMessagePlaceholder )
```

Definition at line 134 of file [Bluetooth_program.c](#).

```
00135 {
00136     HBluetooth_vSendString(L_strMessagePlaceholder);
00137     return HBluetooth_u8ReceiveByte();
00138 }
```

References [HBluetooth_u8ReceiveByte\(\)](#), and [HBluetooth_vSendString\(\)](#).

7.10.2 Variable Documentation

7.10.2.1 Bluetooth_Config

```
USART_InitType Bluetooth_Config [extern]
```

Definition at line 26 of file [Bluetooth_config.c](#).

Referenced by [HBluetooth_vInit\(\)](#).

7.10.2.2 Bluetooth_CLK

```
USART_ClockInitTypeDef Bluetooth_CLK [extern]
```

Definition at line 37 of file [Bluetooth_config.c](#).

Referenced by [HBluetooth_vInit\(\)](#).

7.10.2.3 Bluetooth_UART_ID

```
USART_MemoryMapType* Bluetooth_UART_ID [extern]
```

Definition at line 42 of file [Bluetooth_config.c](#).

Referenced by [HBluetooth_u8ReceiveByte\(\)](#), [HBluetooth_vDisable\(\)](#), [HBluetooth_vEnable\(\)](#), [HBluetooth_vInit\(\)](#), [HBluetooth_vReceiveString\(\)](#), [HBluetooth_vSendByte\(\)](#), and [HBluetooth_vSendString\(\)](#).

7.11 Bluetooth_program.c

[Go to the documentation of this file.](#)

```

00001 /* FILENAME: Bluetooth_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Thu 01/05/2023
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/UART/UART_interface.h"
00019
00020 #include "Bluetooth_interface.h"
00021 #include "Bluetooth_private.h"
00022 #include "Bluetooth_config.h"
00023
00024 /***** Externed variables *****/
00025 /*           * */
00026 /***** */
00027
00028 extern USART_InitType      Bluetooth_Config      ;
00029
00030 extern USART_ClockInitTypeDef    Bluetooth_CLK      ;
00031
00032 extern USART_MemoryMapType *   Bluetooth_UART_ID  ;
00033
00034 /***** Functions' implementations *****/
00035 /*           * */
00036 /***** */
00037
00038 FUNC(void) HBluetooth_vInit( void )
00039 {
00040
00041     MUSART_vInit( &Bluetooth_Config, &Bluetooth_CLK, Bluetooth_UART_ID ) ;
00042
00043 }
00044
00045
00046 /***** */
00047 /***** */
00048 void HBluetooth_vSendByte(VAR(u8_t) A_u8Byte)
00049 {
00050
00051     MUSART_vTransmitByte( Bluetooth_UART_ID, A_u8Byte ) ;
00052
00053 }
00054
00055
00056 /***** */
00057
00058 FUNC(u8_t) HBluetooth_u8ReceiveByte( void )
00059 {
00060
00061     u8_t L_u8ReceivedByte = INITIAL_ZERO ;
00062
00063     L_u8ReceivedByte = MUSART_u8ReceiveByteSyncBlocking( Bluetooth_UART_ID ) ;
00064
00065     return L_u8ReceivedByte ;
00066
00067 }
00068
00069
00070 /***** */
00071
00072 void HBluetooth_vSendString(P2VAR(c8_t) A_ptrc8String)
00073 {
00074
00075     MUSART_vTransmitString( Bluetooth_UART_ID, A_ptrc8String ) ;
00076

```

```
00077 }
00078
00079
00080 /*****
00081 ****
00082 void HBluetooth_vReceiveString(P2VAR(c8_t) A_c8YourString)
00083 {
00084
00085     MUSART_vRecieveString( Bluetooth_UART_ID, A_c8YourString ) ;
00086
00087 }
00088
00089
00090 /*****
00091 ****
00092 FUNC(u8_t) HBluetooth_u8CompStrings(P2VAR(c8_t) String1, P2VAR(c8_t) String2)
00093 {
00094
00095     u8_t L_u8Result = INITIAL_ZERO ;
00096
00097     L_u8Result = MUSART_u8CompareString( String1, String2 ) ;
00098
00099     return L_u8Result ;
00100
00101 }
00102
00103
00104 /*****
00105 ****
00106 void HBluetooth_vEnable( void )
00107 {
00108
00109     MUSART_vEnable( Bluetooth_UART_ID ) ;
00110
00111 }
00112
00113
00114 /*****
00115 ****
00116 void HBluetooth_vDisable( void )
00117 {
00118
00119     MUSART_vDisable( Bluetooth_UART_ID ) ;
00120
00121 }
00122
00123
00124 /*****
00125 ****
00126 void HBluetooth_ExchangeString(P2VAR(c8_t) L_strMessagePlaceholder, P2VAR(c8_t) L_strInfo)
00127 {
00128     HBluetooth_vSendString(L_strMessagePlaceholder);
00129     HBluetooth_vReceiveString(L_strInfo);
00130 }
00131
00132
00133 /*****
00134 ****
00135 FUNC(u8_t) HBluetooth_ExchangeByte(P2VAR(c8_t) L_strMessagePlaceholder)
00136 {
00137     HBluetooth_vSendString(L_strMessagePlaceholder);
00138     return HBluetooth_u8ReceiveByte();
00139 }
00140
00141 /*****
```

7.12 COTS/HAL/CarControl/CarControl_config.h File Reference

7.13 CarControl_config.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _CARCONTROL_CONFIG_H
00002 #define _CARCONTROL_CONFIG_H
00003
00004 #endif // _CARCONTROL_CONFIG_H
```

7.14 COTS/HAL/CarControl/CarControl_interface.h File Reference

Functions

- void [HCarControl_vInitCar](#) (void)
- void [HCarControl_vMoveForward](#) (void)
- void [HCarControl_vMoveBackward](#) (void)
- void [HCarControl_vTurnRight](#) (void)
- void [HCarControl_vTurnLeft](#) (void)
- void [HCarControl_vStopCar](#) (void)

7.14.1 Function Documentation

7.14.1.1 HCarControl_vInitCar()

```
void HCarControl_vInitCar (
    void )
```

Definition at line 61 of file [CarControl_program.c](#).

```
00062 {
00063     HDCM_vInitMotor(&dcmMotor1);
00064     HDCM_vInitMotor(&dcmMotor2);
00065     HDCM_vInitMotor(&dcmMotor3);
00066     HDCM_vInitMotor(&dcmMotor4);
00067 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vInitMotor\(\)](#).

7.14.1.2 HCarControl_vMoveForward()

```
void HCarControl_vMoveForward (
    void )
```

Definition at line 69 of file [CarControl_program.c](#).

```
00070 {
00071     HDCM_vMoveForward(&dcmMotor1);
00072     HDCM_vMoveForward(&dcmMotor2);
00073     HDCM_vMoveForward(&dcmMotor3);
00074     HDCM_vMoveForward(&dcmMotor4);
00075 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vMoveForward\(\)](#).

7.14.1.3 HCarControl_vMoveBackward()

```
void HCarControl_vMoveBackward (
    void )
```

Definition at line 77 of file [CarControl_program.c](#).

```
00078 {
00079     HDCM_vMoveBackward(&dcmMotor1);
00080     HDCM_vMoveBackward(&dcmMotor2);
00081     HDCM_vMoveBackward(&dcmMotor3);
00082     HDCM_vMoveBackward(&dcmMotor4);
00083 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vMoveBackward\(\)](#).

7.14.1.4 HCarControl_vTurnRight()

```
void HCarControl_vTurnRight (
    void )
```

Definition at line 85 of file [CarControl_program.c](#).

```
00086 {
00087     HDCM_vMoveForward(&dcmMotor1);
00088     HDCM_vMoveBackward(&dcmMotor2);
00089     HDCM_vMoveForward(&dcmMotor3);
00090     HDCM_vMoveBackward(&dcmMotor4);
00091 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), [HDCM_vMoveBackward\(\)](#), and [HDCM_vMoveForward\(\)](#).

7.14.1.5 HCarControl_vTurnLeft()

```
void HCarControl_vTurnLeft (
    void )
```

Definition at line 93 of file [CarControl_program.c](#).

```
00094 {
00095     HDCM_vMoveBackward(&dcmMotor1);
00096     HDCM_vMoveForward(&dcmMotor2);
00097     HDCM_vMoveBackward(&dcmMotor3);
00098     HDCM_vMoveForward(&dcmMotor4);
00099 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), [HDCM_vMoveBackward\(\)](#), and [HDCM_vMoveForward\(\)](#).

7.14.1.6 HCarControl_vStopCar()

```
void HCarControl_vStopCar (
    void )
```

Definition at line 101 of file [CarControl_program.c](#).

```
00102 {
00103     HDCM_vStopMotor(&dcmMotor1);
00104     HDCM_vStopMotor(&dcmMotor2);
00105     HDCM_vStopMotor(&dcmMotor3);
00106     HDCM_vStopMotor(&dcmMotor4);
00107 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vStopMotor\(\)](#).

7.15 CarControl_interface.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _CARCONTROL_INTERFACE_H
00002 #define _CARCONTROL_INTERFACE_H
00003
00004 void HCarControl_vInitCar(void);
00005 void HCarControl_vMoveForward(void);
00006 void HCarControl_vMoveBackward(void);
00007 void HCarControl_vTurnRight(void);
00008 void HCarControl_vTurnLeft(void);
00009 void HCarControl_vStopCar(void);
00010
00011 #endif // _CARCONTROL_INTERFACE_H
```

7.16 COTS/HAL/CarControl/CarControl_private.h File Reference

7.17 CarControl_private.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _CARCONTROL_PRIVATE_H
00002 #define _CARCONTROL_PRIVATE_H
00003
00004 #endif // _CARCONTROL_PRIVATE_H
```

7.18 COTS/HAL/CarControl/CarControl_program.c File Reference

```
#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 "../../HAL/DCMOTOR/DCM_interface.h"
#include "CarControl_interface.h"
#include "CarControl_private.h"
#include "CarControl_config.h"
```

Functions

- void [HCarControl_vInitCar](#) (void)
- void [HCarControl_vMoveForward](#) (void)
- void [HCarControl_vMoveBackward](#) (void)
- void [HCarControl_vTurnRight](#) (void)
- void [HCarControl_vTurnLeft](#) (void)
- void [HCarControl_vStopCar](#) (void)

Variables

- [DCM_MotorConfiguration](#) dcmMotor1
- [DCM_MotorConfiguration](#) dcmMotor2
- [DCM_MotorConfiguration](#) dcmMotor3
- [DCM_MotorConfiguration](#) dcmMotor4

7.18.1 Function Documentation

7.18.1.1 HCarControl_vInitCar()

```
void HCarControl_vInitCar (
    void )
```

Definition at line 61 of file [CarControl_program.c](#).

```
00062 {
00063     HDCM_vInitMotor(&dcmMotor1);
00064     HDCM_vInitMotor(&dcmMotor2);
00065     HDCM_vInitMotor(&dcmMotor3);
00066     HDCM_vInitMotor(&dcmMotor4);
00067 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vInitMotor\(\)](#).

7.18.1.2 HCarControl_vMoveForward()

```
void HCarControl_vMoveForward (
    void )
```

Definition at line 69 of file [CarControl_program.c](#).

```
00070 {
00071     HDCM_vMoveForward(&dcmMotor1);
00072     HDCM_vMoveForward(&dcmMotor2);
00073     HDCM_vMoveForward(&dcmMotor3);
00074     HDCM_vMoveForward(&dcmMotor4);
00075 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vMoveForward\(\)](#).

7.18.1.3 HCarControl_vMoveBackward()

```
void HCarControl_vMoveBackward (
    void )
```

Definition at line 77 of file [CarControl_program.c](#).

```
00078 {
00079     HDCM_vMoveBackward(&dcmMotor1);
00080     HDCM_vMoveBackward(&dcmMotor2);
00081     HDCM_vMoveBackward(&dcmMotor3);
00082     HDCM_vMoveBackward(&dcmMotor4);
00083 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vMoveBackward\(\)](#).

7.18.1.4 HCarControl_vTurnRight()

```
void HCarControl_vTurnRight (
    void )
```

Definition at line 85 of file [CarControl_program.c](#).

```
00086 {
00087     HDCM_vMoveForward(&dcmMotor1);
00088     HDCM_vMoveBackward(&dcmMotor2);
00089     HDCM_vMoveForward(&dcmMotor3);
00090     HDCM_vMoveBackward(&dcmMotor4);
00091 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), [HDCM_vMoveBackward\(\)](#), and [HDCM_vMoveForward\(\)](#).

7.18.1.5 HCarControl_vTurnLeft()

```
void HCarControl_vTurnLeft (
    void )
```

Definition at line 93 of file [CarControl_program.c](#).

```
00094 {
00095     HDCM_vMoveBackward(&dcmMotor1);
00096     HDCM_vMoveForward(&dcmMotor2);
00097     HDCM_vMoveBackward(&dcmMotor3);
00098     HDCM_vMoveForward(&dcmMotor4);
00099 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), [HDCM_vMoveBackward\(\)](#), and [HDCM_vMoveForward\(\)](#).

7.18.1.6 HCarControl_vStopCar()

```
void HCarControl_vStopCar (
    void )
```

Definition at line 101 of file [CarControl_program.c](#).

```
00102 {
00103     HDCM_vStopMotor(&dcmMotor1);
00104     HDCM_vStopMotor(&dcmMotor2);
00105     HDCM_vStopMotor(&dcmMotor3);
00106     HDCM_vStopMotor(&dcmMotor4);
00107 }
```

References [dcmMotor1](#), [dcmMotor2](#), [dcmMotor3](#), [dcmMotor4](#), and [HDCM_vStopMotor\(\)](#).

7.18.2 Variable Documentation

7.18.2.1 dcmMotor1

`DCM_MotorConfiguration dcmMotor1`

Initial value:

```
=
{
    .u8Port = (0) ,
    .u8Pin1 = (0) ,
    .u8Pin2 = (1) ,
    .u8Direction = (0)
}
```

Definition at line 22 of file [CarControl_program.c](#).

Referenced by [HCarControl_vInitCar\(\)](#), [HCarControl_vMoveBackward\(\)](#), [HCarControl_vMoveForward\(\)](#), [HCarControl_vStopCar\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.18.2.2 dcmMotor2

`DCM_MotorConfiguration dcmMotor2`

Initial value:

```
=
{
    .u8Port      = (0) ,
    .u8Pin1     = (2) ,
    .u8Pin2     = (3) ,
    .u8Direction = (0)
}
```

Definition at line 31 of file [CarControl_program.c](#).

Referenced by [HCarControl_vInitCar\(\)](#), [HCarControl_vMoveBackward\(\)](#), [HCarControl_vMoveForward\(\)](#), [HCarControl_vStopCar\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.18.2.3 dcmMotor3

`DCM_MotorConfiguration dcmMotor3`

Initial value:

```
=
{
    .u8Port      = (0) ,
    .u8Pin1     = (8) ,
    .u8Pin2     = (9) ,
    .u8Direction = (0)
}
```

Definition at line 40 of file [CarControl_program.c](#).

Referenced by [HCarControl_vInitCar\(\)](#), [HCarControl_vMoveBackward\(\)](#), [HCarControl_vMoveForward\(\)](#), [HCarControl_vStopCar\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.18.2.4 dcmMotor4

```
DCM_MotorConfiguration dcmMotor4
```

Initial value:

```
=
{
    .u8Port      = (0) ,
    .u8Pin1     = (10) ,
    .u8Pin2     = (11) ,
    .u8Direction = (0)
}
```

Definition at line 49 of file [CarControl_program.c](#).

Referenced by [HCarControl_vInitCar\(\)](#), [HCarControl_vMoveBackward\(\)](#), [HCarControl_vMoveForward\(\)](#), [HCarControl_vStopCar\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.19 CarControl_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 #include "../../LIB/LSTD_VALUES.h"
00008 #include "../../LIB/LSTD_BITMATH.h"
00009
00010 #include "../../MCAL/GPIO/GPIO_interface.h"
00011 #include "../../HAL/DCMOTOR/DCM_interface.h"
00012
00013 #include "CarControl_interface.h"
00014 #include "CarControl_private.h"
00015 #include "CarControl_config.h"
00016
00017 /***** *****
00018 /*           Functions' implementations                  */
00019 /***** *****
00020
00021 // Initialize motor 1
00022 VAR(DCM_MotorConfiguration) dcmMotor1 =
00023 {
00024     .u8Port = GPIO_PORTA,
00025     .u8Pin1 = GPIOx_PIN0,
00026     .u8Pin2 = GPIOx_PIN1,
00027     .u8Direction = INITIAL_ZERO
00028 };
00029
00030 // Initialize motor 2
00031 VAR(DCM_MotorConfiguration) dcmMotor2 =
00032 {
00033     .u8Port      = GPIO_PORTA,
00034     .u8Pin1     = GPIOx_PIN2,
00035     .u8Pin2     = GPIOx_PIN3,
00036     .u8Direction = INITIAL_ZERO
00037 };
00038
00039 // // Initialize motor 3
00040 VAR(DCM_MotorConfiguration) dcmMotor3 =
00041 {
00042     .u8Port      = GPIO_PORTA,
00043     .u8Pin1     = GPIOx_PIN8,
00044     .u8Pin2     = GPIOx_PIN9,
00045     .u8Direction = INITIAL_ZERO
00046 };
00047
00048 // // Initialize motor 4
00049 VAR(DCM_MotorConfiguration) dcmMotor4 =
00050 {
00051     .u8Port      = GPIO_PORTA ,
00052     .u8Pin1     = GPIOx_PIN10,
00053     .u8Pin2     = GPIOx_PIN11,
00054     .u8Direction = INITIAL_ZERO
00055 };
```

```

00056
00057 // Motors Layout
00058 // M1 ----- M2
00059 // M3 ----- M4
00060
00061 FUNC(void) HCarControl_vInitCar(void)
00062 {
00063     HDCM_vInitMotor(&dcmMotor1);
00064     HDCM_vInitMotor(&dcmMotor2);
00065     HDCM_vInitMotor(&dcmMotor3);
00066     HDCM_vInitMotor(&dcmMotor4);
00067 }
00068
00069 FUNC(void) HCarControl_vMoveForward(void)
00070 {
00071     HDCM_vMoveForward(&dcmMotor1);
00072     HDCM_vMoveForward(&dcmMotor2);
00073     HDCM_vMoveForward(&dcmMotor3);
00074     HDCM_vMoveForward(&dcmMotor4);
00075 }
00076
00077 FUNC(void) HCarControl_vMoveBackward(void)
00078 {
00079     HDCM_vMoveBackward(&dcmMotor1);
00080     HDCM_vMoveBackward(&dcmMotor2);
00081     HDCM_vMoveBackward(&dcmMotor3);
00082     HDCM_vMoveBackward(&dcmMotor4);
00083 }
00084
00085 FUNC(void) HCarControl_vTurnRight(void)
00086 {
00087     HDCM_vMoveForward(&dcmMotor1);
00088     HDCM_vMoveBackward(&dcmMotor2);
00089     HDCM_vMoveForward(&dcmMotor3);
00090     HDCM_vMoveBackward(&dcmMotor4);
00091 }
00092
00093 FUNC(void) HCarControl_vTurnLeft(void)
00094 {
00095     HDCM_vMoveBackward(&dcmMotor1);
00096     HDCM_vMoveForward(&dcmMotor2);
00097     HDCM_vMoveBackward(&dcmMotor3);
00098     HDCM_vMoveForward(&dcmMotor4);
00099 }
00100
00101 FUNC(void) HCarControl_vStopCar(void)
00102 {
00103     HDCM_vStopMotor(&dcmMotor1);
00104     HDCM_vStopMotor(&dcmMotor2);
00105     HDCM_vStopMotor(&dcmMotor3);
00106     HDCM_vStopMotor(&dcmMotor4);
00107 }

```

7.20 COTS/HAL/DCMOTOR/DCM_config.h File Reference

This file contains the configuration information for the DC Motor module.

7.20.1 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](#).

7.21 DCM_config.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _DCM_config_H
00011 #define _DCM_config_H
00012
00013
00014
00015
00016
00017
00018
00019
00020
00021 #endif // _DCM_config_H
```

7.22 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.

Functions

- void [HDCM_vInitMotor \(P2VAR\(DCM_MotorConfiguration\) pMotorConfiguration\)](#)
Initialize a motor Initialize a motor with a certain GPIO Port & Pin based on the passed motor configuration.
- void [HDCM_vMoveForward \(P2VAR\(DCM_MotorConfiguration\) pMotorConfiguration\)](#)
Move a certain motor forward.
- void [HDCM_vMoveBackward \(P2VAR\(DCM_MotorConfiguration\) pMotorConfiguration\)](#)
Move a certain motor backward.
- void [HDCM_vStopMotor \(P2VAR\(DCM_MotorConfiguration\) pMotorConfiguration\)](#)
Stop a certain motor's movement completely.
- void [HDCM_vMotorSpeedCntrl \(VAR\(u8_t\) A_u8SpeedRatio\)](#)

7.22.1 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](#).

7.22.2 Function Documentation

7.22.2.1 HDCM_vInitMotor()

```
void HDCM_vInitMotor (
    P2VAR(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](#)

Referenced by [HCarControl_vInitCar\(\)](#).

7.22.2.2 HDCM_vMoveForward()

```
void HDCM_vMoveForward (
    P2VAR(DCM_MotorConfiguration) pMotorConfiguration )
```

Move a certain motor forward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move forward
----	----------------------------	---

Referenced by [HCarControl_vMoveForward\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.22.2.3 HDCM_vMoveBackward()

```
void HDCM_vMoveBackward (
    P2VAR(DCM_MotorConfiguration) pMotorConfiguration )
```

Move a certain motor backward.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to move backward
----	----------------------------	--

Referenced by [HCarControl_vMoveBackward\(\)](#), [HCarControl_vTurnLeft\(\)](#), and [HCarControl_vTurnRight\(\)](#).

7.22.2.4 HDCM_vStopMotor()

```
void HDCM_vStopMotor (
    P2VAR(DCM_MotorConfiguration) pMotorConfiguration )
```

Stop a certain motor's movement completely.

Parameters

in	<i>pMotorConfiguration</i>	The motor's configuration to stop
----	----------------------------	-----------------------------------

Referenced by [HCarControl_vStopCar\(\)](#).

7.22.2.5 HDCM_vMotorSpeedCntrl()

```
void HDCM_vMotorSpeedCntrl (
    VAR(u8_t) A_u8SpeedRatio )
```

7.23 DCM_interface.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
0010 #ifndef _DCM_interface_H
0011 #define _DCM_interface_H
0012
0013 /***** Include headers *****/
0014 /*           Include headers           */
0015 /***** Include headers *****/
0016 #include "DCM_private.h"
0017
0018 /***** */
0019 /***** */
0020
0028 typedef struct
0029 {
0033     u8_t u8Port;
0037     u8_t u8Pin1;
0041     u8_t u8Pin2;
0045     u8_t u8Direction;
0046
0047 } DCM_MotorConfiguration;
0048
0049 /***** Functions prototypes *****/
0050 /*           Functions prototypes           */
0051 /***** */
0052
0059 void HDCM_vInitMotor(P2VAR(DCM_MotorConfiguration) pMotorConfiguration);
0060
0065 void HDCM_vMoveForward(P2VAR(DCM_MotorConfiguration) pMotorConfiguration);
0066
0071 void HDCM_vMoveBackward(P2VAR(DCM_MotorConfiguration) pMotorConfiguration);
0072
0077 void HDCM_vStopMotor(P2VAR(DCM_MotorConfiguration) pMotorConfiguration);
0078
0079 // TODO: After developing a timer driver:
0080 void HDCM_vMotorSpeedCntrl(VAR(u8_t) A_u8SpeedRatio);
0081
0082
0083
0084 /***** Interfacing macros *****/
0085 /*           Interfacing macros           */
0086 /***** */
0087
0099 #define FORWARD      (CW)
0100
0106 #define BACKWARD     (CCW)
0107
0110 #endif // _DCM_interface_H

```

7.24 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.

7.24.1 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](#).

7.25 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
```

7.26 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 "DCM_interface.h"
#include "DCM_private.h"
#include "DCM_config.h"
```

Functions

- void [HDCM_vInitMotor](#) (DCM_MotorConfiguration *pMotorConfiguration)
- void [HDCM_vMoveForward](#) (DCM_MotorConfiguration *pMotorConfiguration)
- void [HDCM_vMoveBackward](#) (DCM_MotorConfiguration *pMotorConfiguration)
- void [HDCM_vStopMotor](#) (DCM_MotorConfiguration *pMotorConfiguration)

7.26.1 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](#).

7.26.2 Function Documentation

7.26.2.1 HDCM_vInitMotor()

```
void HDCM_vInitMotor (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Definition at line 29 of file [DCM_program.c](#).

```
00030 {
00031
00032     VAR(MGPIox_ConfigType) DCMotorPin1Init =
00033     {
00034         .Port      = pMotorConfiguration->u8Port ,
00035         .Pin       = pMotorConfiguration->u8Pin1 ,
00036         .Mode      = GPIOx_MODE_OUTPUT
00037         .OutputType = GPIOx_PUSH_PULL
00038         .OutputSpeed = GPIOx_LowSpeed
00039         .InputType   = GPIOx_NoPull
00040     };
00041
00042     VAR(MGPIox_ConfigType) DCMotorPin2Init =
00043     {
00044         .Port      = pMotorConfiguration->u8Port ,
00045         .Pin       = pMotorConfiguration->u8Pin2 ,
00046         .Mode      = GPIOx_MODE_OUTPUT
00047         .OutputType = GPIOx_PUSH_PULL
00048         .OutputSpeed = GPIOx_LowSpeed
00049         .InputType   = GPIOx_NoPull
00050     };
00051
00052     MGPIox_vInit(&DCMotorPin1Init);
00053     MGPIox_vInit(&DCMotorPin2Init);
00054
00055 // Give an initial direction to the DC Motor:
00056 switch( pMotorConfiguration->u8Direction )
00057 {
00058
00059     case FORWARD :  HDCM_vMoveForward(pMotorConfiguration);      break;
00060
00061     case BACKWARD :  HDCM_vMoveBackward(pMotorConfiguration);   break;
00062
00063     /* Default case: stop the motor */
00064     default       :  HDCM_vStopMotor(pMotorConfiguration);      break;
00065
00066 }
00067
00068 }
```

References [BACKWARD](#), [FORWARD](#), [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSH_PULL](#), [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), [HDCM_vStopMotor\(\)](#), [MGPIox_vInit\(\)](#), [DCM_MotorConfiguration::u8Direction](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), [DCM_MotorConfiguration::u8Port](#), and [VAR](#).

7.26.2.2 HDCM_vMoveForward()

```
void HDCM_vMoveForward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Definition at line 73 of file [DCM_program.c](#).

```
00074 {
00075     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_HIGH);
00076     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00077 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIox_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.26.2.3 HDCM_vMoveBackward()

```
void HDCM_vMoveBackward (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Definition at line 82 of file [DCM_program.c](#).

```
00083 {
00084     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00085     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_HIGH);
00086 }
```

References [GPIOx_HIGH](#), [GPIOx_LOW](#), [MGPIox_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#) and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.26.2.4 HDCM_vStopMotor()

```
void HDCM_vStopMotor (
    DCM_MotorConfiguration * pMotorConfiguration )
```

Definition at line 91 of file [DCM_program.c](#).

```
00092 {
00093     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00094     MGPIox_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00095 }
```

References [GPIOx_LOW](#), [MGPIox_vSetValue\(\)](#), [DCM_MotorConfiguration::u8Pin1](#), [DCM_MotorConfiguration::u8Pin2](#), and [DCM_MotorConfiguration::u8Port](#).

Referenced by [HDCM_vInitMotor\(\)](#).

7.27 DCM_program.c

[Go to the documentation of this file.](#)

```
00001
00009 /*****
00010 *          Include headers
00011 *****/
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
00021 #include "DCM_interface.h"
00022 #include "DCM_private.h"
00023 #include "DCM_config.h"
00024
00025 /*****
00026 *          Functions implementations
00027 *****/
00028
00029 FUNC(void) HDCM_vInitMotor(P2VAR(DCM_MotorConfiguration) pMotorConfiguration)
00030 {
00031
00032     VAR(MGPIox_ConfigType) DCMotorPin1Init =
00033     {
00034         .Port      = pMotorConfiguration->u8Port ,
00035         .Pin       = pMotorConfiguration->u8Pin1 ,
00036         .Mode      = GPIOx_MODE_OUTPUT
00037     },
00038 }
```

```

00037     .OutputType    = GPIOx_PUSH_PULL
00038     .OutputSpeed   = GPIOx_LowSpeed
00039     .InputType     = GPIOx_NoPull
00040 };
00041
00042 VAR(MGPIOx_ConfigType) DCMotorPin2Init =
00043 {
00044     .Port          = pMotorConfiguration->u8Port,
00045     .Pin           = pMotorConfiguration->u8Pin2,
00046     .Mode          = GPIOx_MODE_OUTPUT
00047     .OutputType    = GPIOx_PUSH_PULL,
00048     .OutputSpeed   = GPIOx_LowSpeed,
00049     .InputType     = GPIOx_NoPull
00050 };
00051
00052 MGPIOx_vInit(&DCMotorPin1Init);
00053 MGPIOx_vInit(&DCMotorPin2Init);
00054
00055 // Give an initial direction to the DC Motor:
00056 switch( pMotorConfiguration->u8Direction )
00057 {
00058     case FORWARD :  HDCM_vMoveForward(pMotorConfiguration);      break;
00059     case BACKWARD : HDCM_vMoveBackward(pMotorConfiguration);    break;
00060
00061     /* Default case: stop the motor */
00062     default       :  HDCM_vStopMotor(pMotorConfiguration);      break;
00063 }
00064
00065 }
00066 }
00067
00068 }
00069
00070 ****
00071 ****
00072
00073 FUNC(void) HDCM_vMoveForward(P2VAR(DCM_MotorConfiguration) pMotorConfiguration)
00074 {
00075     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_HIGH);
00076     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00077 }
00078
00079 ****
00080 ****
00081
00082 FUNC(void) HDCM_vMoveBackward(P2VAR(DCM_MotorConfiguration) pMotorConfiguration)
00083 {
00084     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00085     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_HIGH);
00086 }
00087
00088 ****
00089 ****
00090
00091 FUNC(void) HDCM_vStopMotor(P2VAR(DCM_MotorConfiguration) pMotorConfiguration)
00092 {
00093     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin1, GPIOx_LOW);
00094     MGPIOx_vSetValue(pMotorConfiguration->u8Port, pMotorConfiguration->u8Pin2, GPIOx_LOW);
00095 }
00096
00097 ****
00098 ****
00099
00100
00101
00102
00103
00104

```

7.28 COTS/HAL/LCD/LCD_config.h File Reference

```
#include "LCD_private.h"
```

Macros

- `#define HLCD_CTRL_PORT GPIO_PORTB`
- `#define HLCD_RS_PIN GPIOx_PIN12`
- `#define HLCD_RW_PIN GPIOx_PIN13`
- `#define HLCD_EN_PIN GPIOx_PIN14`
- `#define HLCD_DATA_PORT GPIO_PORTA`
- `#define D0_PIN GPIOx_PIN1`
- `#define D1_PIN GPIOx_PIN2`
- `#define D2_PIN GPIOx_PIN3`
- `#define D3_PIN GPIOx_PIN4`
- `#define D4_PIN GPIOx_PIN5`
- `#define D5_PIN GPIOx_PIN6`
- `#define D6_PIN GPIOx_PIN7`
- `#define D7_PIN GPIOx_PIN8`
- `#define HLCD_FuctionSet_Cmd FuctionSetCmd`
- `#define HLCD_DisponOffCTRL_Cmd DispOnOffCTRLCmd`
- `#define HLCD_DispoFF_Cmd DispOffCmd`
- `#define HLCD_DispcursorWithBlinking_Cmd DispCursorWithBlinkingCmd`
- `#define HLCD_SetCursorBlinkingOFF_Cmd SetCursorBlinkingOFFCmd`
- `#define HLCD_Dispclear_Cmd DispClearCmd`
- `#define HLCD_EntryModeSet_Cmd EntryModeSet_Cmd`
- `#define HLCD_EntryModeSet_ShiftLeftOn_Cmd EntryModeSet_ShiftLeftOn_Cmd`
- `#define HLCD_CharactersNums 16`
- `#define FIRST_ROW_IDX 0`
- `#define FIRST_ROW_START 0x00`
- `#define SEC_ROW_START 0x40`
- `#define SET_DDRAM_AC_MASK 0x80`
- `#define SET_CGRAM_AC_MASK 0x40`
- `#define NumOf_CGRAM_Patterns 8`
- `#define FirstByteInCGRAM_Pattern 0`
- `#define LastByteInCGRAM_Pattern 8`

7.28.1 Macro Definition Documentation

7.28.1.1 HLCD_CTRL_PORT

```
#define HLCD_CTRL_PORT GPIO_PORTB
```

Definition at line 15 of file [LCD_config.h](#).

7.28.1.2 HLCD_RS_PIN

```
#define HLCD_RS_PIN GPIOx_PIN12
```

Definition at line 17 of file [LCD_config.h](#).

7.28.1.3 **HLCD_RW_PIN**

```
#define HLCD_RW_PIN GPIOx_PIN13
```

Definition at line 18 of file [LCD_config.h](#).

7.28.1.4 **HLCD_EN_PIN**

```
#define HLCD_EN_PIN GPIOx_PIN14
```

Definition at line 19 of file [LCD_config.h](#).

7.28.1.5 **HLCD_DATA_PORT**

```
#define HLCD_DATA_PORT GPIO_PORTA
```

Definition at line 22 of file [LCD_config.h](#).

7.28.1.6 **D0_PIN**

```
#define D0_PIN GPIOx_PIN1
```

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

7.28.1.7 **D1_PIN**

```
#define D1_PIN GPIOx_PIN2
```

Definition at line 25 of file [LCD_config.h](#).

7.28.1.8 **D2_PIN**

```
#define D2_PIN GPIOx_PIN3
```

Definition at line 26 of file [LCD_config.h](#).

7.28.1.9 D3_PIN

```
#define D3_PIN GPIOx_PIN4
```

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

7.28.1.10 D4_PIN

```
#define D4_PIN GPIOx_PIN5
```

Definition at line 28 of file [LCD_config.h](#).

7.28.1.11 D5_PIN

```
#define D5_PIN GPIOx_PIN6
```

Definition at line 29 of file [LCD_config.h](#).

7.28.1.12 D6_PIN

```
#define D6_PIN GPIOx_PIN7
```

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

7.28.1.13 D7_PIN

```
#define D7_PIN GPIOx_PIN8
```

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

7.28.1.14 HLCD_FuctionSet_Cmd

```
#define HLCD_FunctionSet_Cmd FuctionSetCmd
```

Definition at line 34 of file [LCD_config.h](#).

7.28.1.15 **HLCD_DispOnOffCTRL_Cmd**

```
#define HLCD_DispOnOffCTRL_Cmd DispOnOffCTRLCmd
```

Definition at line 36 of file [LCD_config.h](#).

7.28.1.16 **HLCD_DispOff_Cmd**

```
#define HLCD_DispOff_Cmd DispOffCmd
```

Definition at line 38 of file [LCD_config.h](#).

7.28.1.17 **HLCD_DispCursorWithBlinking_Cmd**

```
#define HLCD_DispCursorWithBlinking_Cmd DispCursorWithBlinkingCmd
```

Definition at line 40 of file [LCD_config.h](#).

7.28.1.18 **HLCD_SetCursorBlinkingOFF_Cmd**

```
#define HLCD_SetCursorBlinkingOFF_Cmd SetCursorBlinkingOFFCmd
```

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

7.28.1.19 **HLCD_DispClear_Cmd**

```
#define HLCD_DispClear_Cmd DispClearCmd
```

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

7.28.1.20 **HLCD_EntryModeSet_Cmd**

```
#define HLCD_EntryModeSet_Cmd EntryModeSet_Cmd
```

Definition at line 46 of file [LCD_config.h](#).

7.28.1.21 HLCD_EntryModeSet_ShiftLeftOn_Cmd

```
#define HLCD_EntryModeSet_ShiftLeftOn_Cmd EntryModeSet_ShiftLeftOn_Cmd
```

Definition at line 48 of file [LCD_config.h](#).

7.28.1.22 HLCD_CharactersNums

```
#define HLCD_CharactersNums 16
```

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

7.28.1.23 FIRST_ROW_IDX

```
#define FIRST_ROW_IDX 0
```

Definition at line 54 of file [LCD_config.h](#).

7.28.1.24 FIRST_ROW_START

```
#define FIRST_ROW_START 0x00
```

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

7.28.1.25 SEC_ROW_START

```
#define SEC_ROW_START 0x40
```

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

7.28.1.26 SET_DDRAM_AC_MASK

```
#define SET_DDRAM_AC_MASK 0x80
```

Definition at line 60 of file [LCD_config.h](#).

7.28.1.27 SET_CGRAM_AC_MASK

```
#define SET_CGRAM_AC_MASK 0x40
```

Definition at line 62 of file [LCD_config.h](#).

7.28.1.28 NumOf_CGRAM_Patterns

```
#define NumOf_CGRAM_Patterns 8
```

Definition at line 64 of file [LCD_config.h](#).

7.28.1.29 FirstByteInCGRAM_Pattern

```
#define FirstByteInCGRAM_Pattern 0
```

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

7.28.1.30 LastByteInCGRAM_Pattern

```
#define LastByteInCGRAM_Pattern 8
```

Definition at line 68 of file [LCD_config.h](#).

7.29 LCD_config.h

[Go to the documentation of this file.](#)

```
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_PIN1
00025 #define D1_PIN GPIOx_PIN2
00026 #define D2_PIN GPIOx_PIN3
```

```

00027 #define D3_PIN GPIOx_PIN4
00028 #define D4_PIN GPIOx_PIN5
00029 #define D5_PIN GPIOx_PIN6
00030 #define D6_PIN GPIOx_PIN7
00031 #define D7_PIN GPIOx_PIN8
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_ */

```

7.30 COTS/HAL/LCD/LCD_interface.h File Reference

Macros

- #define MIN_IDX_OF_ROWS 0
- #define MAX_IDX_OF_ROWS 1
- #define MIN_IDX_OF_COL 0
- #define MAX_IDX_OF_COL 15
- #define HLCD_LINE1 0
- #define HLCD_LINE2 1
- #define HLCD_Square1 0
- #define HLCD_Square2 1
- #define HLCD_Square3 2
- #define HLCD_Square4 3
- #define HLCD_Square5 4
- #define HLCD_Square6 5
- #define HLCD_Square7 6
- #define HLCD_Square8 7
- #define HLCD_Square9 8
- #define HLCD_Square10 9
- #define HLCD_Square11 10
- #define HLCD_Square12 11

- #define `HLCD_Square13` 12
- #define `HLCD_Square14` 13
- #define `HLCD_Square15` 14
- #define `HLCD_Square16` 15
- #define `CGRAM_AddressOfPattern0` 0
- #define `CGRAM_AddressOfPattern1` 1
- #define `CGRAM_AddressOfPattern2` 2
- #define `CGRAM_AddressOfPattern3` 3
- #define `CGRAM_AddressOfPattern4` 4
- #define `CGRAM_AddressOfPattern5` 5
- #define `CGRAM_AddressOfPattern7` 6
- #define `CGRAM_AddressOfPattern8` 7

Functions

- void `HLCD_vInit` (void)
- void `HLCD_vSendCommand` (`VAR(u8_t)` A_u8Cmd)
- void `HLCD_vSendData` (`VAR(u8_t)` A_u8Data)
- void `HLCD_vClear` (void)
- void `HLCD_vDispString` (`P2VAR(c8_t)` A_c8Char)
- void `HLCD_vDispNumber` (`VAR(s32_t)` A_s32Num)
- void `HLCD_vSaveCustomChar` (`VAR(u8_t)` A_u8Address, `P2VAR(u8_t)` A_u8CustomChar)
- void `HLCD_vSaveDispChar` (`VAR(u8_t)` A_u8Address, `P2VAR(u8_t)` A_u8CustomChar)
- void `HLCD_vGoTo` (`VAR(u8_t)` A_u8Row, `VAR(u8_t)` A_u8Col)
- void `HLCD_vSetShiftLeftOn` (void)
- void `HLCD_vSetDispOFF` (void)
- void `HLCD_vDispCursorWithBlinking` (void)
- void `HLCD_vSetCursorBlinkingOFF` (void)
- void `HLCD_vDispShiftLeftString` (`P2VAR(c8_t)` A_c8Char)
- void `HLCD_vClearChar` (`VAR(u8_t)` A_u8Row, `VAR(u8_t)` A_u8Col)

7.30.1 Macro Definition Documentation

7.30.1.1 MIN_IDX_OF_ROWS

```
#define MIN_IDX_OF_ROWS 0
```

Definition at line 50 of file [LCD_interface.h](#).

7.30.1.2 MAX_IDX_OF_ROWS

```
#define MAX_IDX_OF_ROWS 1
```

Definition at line 52 of file [LCD_interface.h](#).

7.30.1.3 MIN_IDX_OF_COL

```
#define MIN_IDX_OF_COL 0
```

Definition at line 54 of file [LCD_interface.h](#).

7.30.1.4 MAX_IDX_OF_COL

```
#define MAX_IDX_OF_COL 15
```

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

7.30.1.5 HLCD_LINE1

```
#define HLCD_LINE1 0
```

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

7.30.1.6 HLCD_LINE2

```
#define HLCD_LINE2 1
```

Definition at line 61 of file [LCD_interface.h](#).

7.30.1.7 HLCD_Square1

```
#define HLCD_Square1 0
```

Definition at line 64 of file [LCD_interface.h](#).

7.30.1.8 HLCD_Square2

```
#define HLCD_Square2 1
```

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

7.30.1.9 **HLCD_Square3**

```
#define HLCD_Square3 2
```

Definition at line [68](#) of file [LCD_interface.h](#).

7.30.1.10 **HLCD_Square4**

```
#define HLCD_Square4 3
```

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

7.30.1.11 **HLCD_Square5**

```
#define HLCD_Square5 4
```

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

7.30.1.12 **HLCD_Square6**

```
#define HLCD_Square6 5
```

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

7.30.1.13 **HLCD_Square7**

```
#define HLCD_Square7 6
```

Definition at line [76](#) of file [LCD_interface.h](#).

7.30.1.14 **HLCD_Square8**

```
#define HLCD_Square8 7
```

Definition at line [78](#) of file [LCD_interface.h](#).

7.30.1.15 HLCD_Square9

```
#define HLCD_Square9 8
```

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

7.30.1.16 HLCD_Square10

```
#define HLCD_Square10 9
```

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

7.30.1.17 HLCD_Square11

```
#define HLCD_Square11 10
```

Definition at line 84 of file [LCD_interface.h](#).

7.30.1.18 HLCD_Square12

```
#define HLCD_Square12 11
```

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

7.30.1.19 HLCD_Square13

```
#define HLCD_Square13 12
```

Definition at line 88 of file [LCD_interface.h](#).

7.30.1.20 HLCD_Square14

```
#define HLCD_Square14 13
```

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

7.30.1.21 **HLCD_Square15**

```
#define HLCD_Square15 14
```

Definition at line 92 of file [LCD_interface.h](#).

7.30.1.22 **HLCD_Square16**

```
#define HLCD_Square16 15
```

Definition at line 94 of file [LCD_interface.h](#).

7.30.1.23 **CGRAM_AddressOfPattern0**

```
#define CGRAM_AddressOfPattern0 0
```

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

7.30.1.24 **CGRAM_AddressOfPattern1**

```
#define CGRAM_AddressOfPattern1 1
```

Definition at line 99 of file [LCD_interface.h](#).

7.30.1.25 **CGRAM_AddressOfPattern2**

```
#define CGRAM_AddressOfPattern2 2
```

Definition at line 101 of file [LCD_interface.h](#).

7.30.1.26 **CGRAM_AddressOfPattern3**

```
#define CGRAM_AddressOfPattern3 3
```

Definition at line 103 of file [LCD_interface.h](#).

7.30.1.27 CGRAM_AddressOfPattern4

```
#define CGRAM_AddressOfPattern4 4
```

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

7.30.1.28 CGRAM_AddressOfPattern5

```
#define CGRAM_AddressOfPattern5 5
```

Definition at line 107 of file [LCD_interface.h](#).

7.30.1.29 CGRAM_AddressOfPattern7

```
#define CGRAM_AddressOfPattern7 6
```

Definition at line 109 of file [LCD_interface.h](#).

7.30.1.30 CGRAM_AddressOfPattern8

```
#define CGRAM_AddressOfPattern8 7
```

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

7.30.2 Function Documentation

7.30.2.1 LCD_vInit()

```
void LCD_vInit (
    void )
```

Definition at line 30 of file [LCD_program.c](#).

```
00031 {
00032     MSysTick_vInit( );
00033
00034     MSysTick_vDelayMilliSec( 1000 );
00035
00036     // Set your pin directions:
00037     MGPIOx_ConfigType RS =
00038     {
00039
00040         .Port      = LCD_CTRL_PORT, .Pin = LCD_RS_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00041             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00042             .InputType  = GPIOx_NoPull
00043
00044     } ;
00045
00046     MGPIOx_ConfigType RW =
00047     {
00048
00049         .Port      = LCD_CTRL_PORT, .Pin = LCD_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      = LCD_CTRL_PORT, .Pin = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_CTRL_PORT, .Pin = D5_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00106             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00107             .InputType = GPIOx_NoPull
00108     } ;
00109
00110     MGPIOx_ConfigType D6 =
00111     {
00112
00113         .Port      = LCD_CTRL_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_CTRL_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_DisClear_Cmd ) ;
00150
00151 MSysTick_vDelayMilliSec( 3 ) ;
00152
00153 LCD_vSendCommand( LCD_EntryModeSet_Cmd ) ;
00154
00155 MSysTick_vDelayMilliSec( 1 ) ;
00156
00157 }

```

References `D0_PIN`, `D1_PIN`, `D2_PIN`, `D3_PIN`, `D4_PIN`, `D5_PIN`, `D6_PIN`, `D7_PIN`, `GPIOx_LowSpeed`, `GPIOx_MODE_OUTPUT`, `GPIOx_NoPull`, `GPIOx_PUSH_PULL`, `LCD_CTRL_PORT`, `LCD_DATA_PORT`, `LCD_DisClear_Cmd`, `LCD_DisponOffCTRL_Cmd`, `LCD_EN_PIN`, `LCD_EntryModeSet_Cmd`, `LCD_FunctionSet_Cmd`, `LCD_RS_PIN`, `LCD_RW_PIN`, `LCD_vSendCommand()`, `MGPIOx_vInit()`, `MSysTick_vDelayMilliSec()`, `MSysTick_vInit()`, and `MGPIOx_ConfigType::Port`.

7.30.2.2 LCD_vSendCommand()

```

void LCD_vSendCommand (
    VAR(u8_t) A_u8Cmd )

```

7.30.2.3 LCD_vSendData()

```
void LCD_vSendData (
    VAR(u8_t) A_u8Data )
```

7.30.2.4 LCD_vClear()

```
void LCD_vClear (
    void )
```

Definition at line 224 of file [LCD_program.c](#).

```
00225 {
00226
00227     MSysTick_vDelayMilliSec( 1 ) ;
00228
00229     LCD_vSendCommand( LCD_Dispclear_Cmd ) ;
00230
00231     MSysTick_vDelayMilliSec( 1 ) ;
00232
00233 }
```

References [LCD_Dispclear_Cmd](#), [LCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.30.2.5 LCD_vDispString()

```
void LCD_vDispString (
    P2VAR(c8_t) A_c8Char )
```

7.30.2.6 LCD_vDispNumber()

```
void LCD_vDispNumber (
    VAR(s32_t) A_s32Num )
```

7.30.2.7 LCD_vSaveCustomChar()

```
void LCD_vSaveCustomChar (
    VAR(u8_t) A_u8Address,
    P2VAR(u8_t) A_u8CustomChar )
```

7.30.2.8 HLCD_vSaveDispChar()

```
void HLCD_vSaveDispChar (
    VAR(u8_t) A_u8Address,
    P2VAR(u8_t) A_u8CustomChar )
```

7.30.2.9 HLCD_vGoTo()

```
void HLCD_vGoTo (
    VAR(u8_t) A_u8Row,
    VAR(u8_t) A_u8Col )
```

7.30.2.10 HLCD_vSetShiftLeftOn()

```
void HLCD_vSetShiftLeftOn (
    void )
```

Definition at line 385 of file [LCD_program.c](#).

```
00386 {
00387
00388     MSysTick_vDelayMilliSec( 500 ) ;
00389
00390     HLCD_vSendCommand( HLCD_EntryModeSet_ShiftLeftOn_Cmd ) ;
00391
00392     MSysTick_vDelayMilliSec( 500 ) ;
00393
00394 }
```

References [HLCD_EntryModeSet_ShiftLeftOn_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

Referenced by [HLCD_vDispShiftLeftString\(\)](#).

7.30.2.11 HLCD_vSetDispOFF()

```
void HLCD_vSetDispOFF (
    void )
```

Definition at line 400 of file [LCD_program.c](#).

```
00401 {
00402
00403     MSysTick_vDelayMilliSec( 500 ) ;
00404
00405     HLCD_vSendCommand( HLCD_DispOff_Cmd ) ;
00406
00407     MSysTick_vDelayMilliSec( 500 ) ;
00408
00409 }
```

References [HLCD_DispOff_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.30.2.12 **HLCD_vDispCursorWithBlinking()**

```
void HLCD_vDispCursorWithBlinking (
    void )
```

Definition at line 415 of file [LCD_program.c](#).

```
00416 {
00417
00418     MSysTick_vDelayMilliSec( 500 ) ;
00419
00420     HLCD_vSendCommand( HLCD_DispcursorWithBlinking_Cmd ) ;
00421
00422     MSysTick_vDelayMilliSec( 500 ) ;
00423
00424 }
```

References [HLCD_DispcursorWithBlinking_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.30.2.13 **HLCD_vSetCursorBlinkingOFF()**

```
void HLCD_vSetCursorBlinkingOFF (
    void )
```

Definition at line 429 of file [LCD_program.c](#).

```
00430 {
00431
00432     MSysTick_vDelayMilliSec( 500 ) ;
00433
00434     HLCD_vSendCommand( HLCD_SetCursorBlinkingOFF_Cmd ) ;
00435
00436     MSysTick_vDelayMilliSec( 500 ) ;
00437
00438 }
```

References [HLCD_SetCursorBlinkingOFF_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.30.2.14 **HLCD_vDispShiftLeftString()**

```
void HLCD_vDispShiftLeftString (
    P2VAR(c8_t) A_c8Char )
```

7.30.2.15 **HLCD_vClearChar()**

```
void HLCD_vClearChar (
    VAR(u8_t) A_u8Row,
    VAR(u8_t) A_u8Col )
```

7.31 LCD_interface.h

[Go to the documentation of this file.](#)

```

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 /* ***** Interfacing macros ***** */
00014 /***** */

00015
00016 void HLCD_vInit
00017
00018 void HLCD_vSendCommand
00019
00020 void HLCD_vSendData
00021
00022 void HLCD_vClear
00023
00024 void HLCD_vDispString
00025
00026 void HLCD_vDispNumber
00027
00028 void HLCD_vSaveCustomChar
00029
00030 void HLCD_vSaveDispChar
00031
00032 void HLCD_vGoTo
00033
00034 void HLCD_vSetShiftLeftOn
00035
00036 void HLCD_vSetDispOFF
00037
00038 void HLCD_vDispCursorWithBlinking
00039
00040 void HLCD_vSetCursorBlinkingOFF
00041
00042 void HLCD_vDispShiftLeftString
00043
00044 void HLCD_vClearChar
00045
00046 /***** Macros *****/
00047 /* ***** Interfacing macros ***** */
00048 /***** */

00049
00050 #define MIN_IDX_OF_ROWS 0
00051
00052 #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_ */

```

7.32 COTS/HAL/LCD/LCD_private.h File Reference

Macros

- #define FuctionSetCmd 0b00111000
- #define DispOnOffCTRLCmd 0b00001100
- #define DispCursorWithBlinkingCmd 0b00001111
- #define SetCursorBlinkingOFFCmd 0b00001110
- #define DispOffCmd 0b00001000
- #define DispClearCmd 0b00000001
- #define EntryModeSet_Cmd 0b00000110
- #define EntryModeSet_ShiftLeftOn_Cmd 0b0000011000

7.32.1 Macro Definition Documentation

7.32.1.1 FuctionSetCmd

```
#define FunctionSetCmd 0b00111000
```

Definition at line 14 of file [LCD_private.h](#).

7.32.1.2 DispOnOffCTRLCmd

```
#define DispOnOffCTRLCmd 0b000001100
```

Definition at line 16 of file [LCD_private.h](#).

7.32.1.3 DispCursorWithBlinkingCmd

```
#define DispCursorWithBlinkingCmd 0b000001111
```

Definition at line 18 of file [LCD_private.h](#).

7.32.1.4 SetCursorBlinkingOFFCmd

```
#define SetCursorBlinkingOFFCmd 0b000001110
```

Definition at line 20 of file [LCD_private.h](#).

7.32.1.5 DispOffCmd

```
#define DispOffCmd 0b000001000
```

Definition at line 22 of file [LCD_private.h](#).

7.32.1.6 DispClearCmd

```
#define DispClearCmd 0b000000001
```

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

7.32.1.7 EntryModeSet_Cmd

```
#define EntryModeSet_Cmd 0b00000110
```

Definition at line 26 of file [LCD_private.h](#).

7.32.1.8 EntryModeSet_ShiftLeftOn_Cmd

```
#define EntryModeSet_ShiftLeftOn_Cmd 0b0000011000
```

Definition at line 28 of file [LCD_private.h](#).

7.33 LCD_private.h

[Go to the documentation of this file.](#)

```
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_ */
```

7.34 COTS/HAL/LCD/LCD_program.c File Reference

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../MCAL/RCC/MRCC_interface.h"
#include "../../MCAL/GPIO(GPIO_interface.h"
#include "../../MCAL/SysTick/SysTick_interface.h"
#include "LCD_config.h"
#include "LCD_interface.h"
#include "LCD_private.h"
```

Functions

- void **HLCD_vInit** (void)
- void **HLCD_vSendCommand** (u8_t A_u8Cmd)
- void **HLCD_vSendData** (u8_t A_u8Data)
- void **HLCD_vClear** (void)
- void **HLCD_vDispString** (c8_t *A_c8Char)
- void **HLCD_vDispNumber** (s32_t A_s32Num)
- void **HLCD_vSaveCustomChar** (u8_t A_u8Address, u8_t *A_u8CustomChar)
- void **HLCD_vGoTo** (u8_t A_u8Row, u8_t A_u8Col)
- void **HLCD_vSetShiftLeftOn** (void)
- void **HLCD_vSetDispOFF** (void)
- void **HLCD_vDispCursorWithBlinking** (void)
- void **HLCD_vSetCursorBlinkingOFF** (void)
- void **HLCD_vDispShiftLeftString** (c8_t *A_c8Char)
- void **HLCD_vClearChar** (u8_t A_u8Row, u8_t A_u8Col)

7.34.1 Function Documentation

7.34.1.1 **HLCD_vInit()**

```
void HLCD_vInit (
    void )
```

Definition at line 30 of file [LCD_program.c](#).

```
00031 {
00032
00033     MSysTick_vInit( ) ;
00034
00035     MSysTick_vDelayMilliSec( 1000 ) ;
00036
00037     // Set your pin directions:
00038     MGPIOx_ConfigType RS =
00039     {
00040         .Port      = HLCD_CTRL_PORT, .Pin = HLCD_RS_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00041             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00042             .InputType = GPIOx_NoPull
00043     } ;
00045 }
```

```

00046     MGPIOx_ConfigType RW =
00047     {
00048
00049         .Port      = LCD_CTRL_PORT, .Pin = LCD_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      = LCD_CTRL_PORT, .Pin = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_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      = LCD_DATA_PORT, .Pin = D5_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00106             GPIOx_PUSH_PULL, .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

```

```

00123 }
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 }

```

References [D0_PIN](#), [D1_PIN](#), [D2_PIN](#), [D3_PIN](#), [D4_PIN](#), [D5_PIN](#), [D6_PIN](#), [D7_PIN](#), [GPIOx_LowSpeed](#), [GPIOx_MODE_OUTPUT](#), [GPIOx_NoPull](#), [GPIOx_PUSH_PULL](#), [LCD_CTRL_PORT](#), [LCD_DATA_PORT](#), [LCD_Dispclear_Cmd](#), [LCD_DisponOffCTRL_Cmd](#), [LCD_EN_PIN](#), [LCD_EntryModeSet_Cmd](#), [LCD_FunctionSet_Cmd](#), [LCD_RS_PIN](#), [LCD_RW_PIN](#), [LCD_vSendCommand\(\)](#), [MGPIOx_vInit\(\)](#), [MSysTick_vDelayMilliSec\(\)](#), [MSysTick_vInit\(\)](#), and [MGPIOx_ConfigType::Port](#).

7.34.1.2 LCD_vSendCommand()

```

void LCD_vSendCommand (
    u8_t A_u8Cmd )

```

Definition at line 162 of file [LCD_program.c](#).

```

00163 {
00164
00165 // Set RS pin low to send a command.
00166     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_RS_PIN, GPIOx_LOW ) ;
00167
00168 // Set RW pin low to write the command.
00169     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_RW_PIN, GPIOx_LOW ) ;
00170
00171     MSysTick_vDelayMilliSec( 1 ) ;
00172
00173 // Put the command on the data bus.
00174     GPIO_vSetPortValue( LCD_DATA_PORT, A_u8Cmd ) ;
00175
00176     MSysTick_vDelayMilliSec( 1 ) ;
00177
00178 // Set the enable pin high.
00179     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_HIGH ) ;
00180
00181     MSysTick_vDelayMilliSec( 1 ) ;
00182
00183 // Set the enable pin low.
00184     MGPIOx_vSetPinValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_LOW ) ;
00185
00186     MSysTick_vDelayMilliSec( 1 ) ;
00187
00188 }

```

References [GPIO_vSetPortValue\(\)](#), [GPIOx_HIGH](#), [GPIOx_LOW](#), [HLCD_CTRL_PORT](#), [HLCD_DATA_PORT](#), [HLCD_EN_PIN](#), [HLCD_RS_PIN](#), [HLCD_RW_PIN](#), [MGPIOx_vSetPinValue\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

Referenced by [HLCD_vClear\(\)](#), [HLCD_vDispCursorWithBlinking\(\)](#), [HLCD_vGoTo\(\)](#), [HLCD_vInit\(\)](#), [HLCD_vSaveCustomChar\(\)](#), [HLCD_vSetCursorBlinkingOFF\(\)](#), [HLCD_vSetDispOFF\(\)](#), and [HLCD_vSetShiftLeftOn\(\)](#).

7.34.1.3 HLCD_vSendData()

```
void HLCD_vSendData (
    u8_t A_u8Data )
```

Definition at line 193 of file [LCD_program.c](#).

```
00194 {
00195
00196     // Set RS pin high to send data.
00197     MGPIOx_vSetPinValue( HLCD_CTRL_PORT, HLCD_RS_PIN, GPIOx_HIGH ) ;
00198
00199     // Set RW pin low to write the data.
00200     MGPIOx_vSetPinValue( HLCD_CTRL_PORT, HLCD_RW_PIN, GPIOx_LOW ) ;
00201
00202     MSysTick_vDelayMilliSec( 1 ) ;
00203
00204     // Put the data on the data bus.
00205     GPIO_vSetPortValue( HLCD_DATA_PORT, A_u8Data ) ;
00206
00207     MSysTick_vDelayMilliSec( 1 ) ;
00208
00209     // Set the enable pin high.
00210     MGPIOx_vSetPinValue( HLCD_CTRL_PORT, HLCD_EN_PIN, GPIOx_HIGH ) ;
00211
00212     MSysTick_vDelayMilliSec( 1 ) ;
00213
00214     // Set the enable pin low.
00215     MGPIOx_vSetPinValue( HLCD_CTRL_PORT, HLCD_EN_PIN, GPIOx_LOW ) ;
00216
00217     MSysTick_vDelayMilliSec( 1 ) ;
00218
00219 }
```

References [GPIO_vSetPortValue\(\)](#), [GPIOx_HIGH](#), [GPIOx_LOW](#), [HLCD_CTRL_PORT](#), [HLCD_DATA_PORT](#), [HLCD_EN_PIN](#), [HLCD_RS_PIN](#), [HLCD_RW_PIN](#), [MGPIOx_vSetPinValue\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

Referenced by [HLCD_vClearChar\(\)](#), [HLCD_vDispNumber\(\)](#), [HLCD_vDispShiftLeftString\(\)](#), [HLCD_vDispString\(\)](#), and [HLCD_vSaveCustomChar\(\)](#).

7.34.1.4 HLCD_vClear()

```
void HLCD_vClear (
    void )
```

Definition at line 224 of file [LCD_program.c](#).

```
00225 {
00226
00227     MSysTick_vDelayMilliSec( 1 ) ;
00228
00229     HLCD_vSendCommand( HLCD_Dispclear_Cmd ) ;
00230
00231     MSysTick_vDelayMilliSec( 1 ) ;
00232
00233 }
```

References [HLCD_Dispclear_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.34.1.5 HLCD_vDispString()

```
void HLCD_vDispString (
    c8_t * A_c8Char )
```

Definition at line 238 of file [LCD_program.c](#).

```
00239 {
00240     for( VAR(u8_t) L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00241     {
00242         HLCD_vSendData( A_c8Char[L_u8I] ) ;
00243     }
00244 }
```

References [HLCD_vSendData\(\)](#), and [VAR](#).

7.34.1.6 HLCD_vDispNumber()

```
void HLCD_vDispNumber (
    s32_t A_s32Num )
```

Definition at line 253 of file [LCD_program.c](#).

```
00254 {
00255     VAR(u8_t) L_u8Counter      = 0 ;
00256     VAR(u8_t) LR_u8Digits[10]   = {0} ;
00257
00258     // In case the number is signed.
00259     if ( A_s32Num < 0 )
00260     {
00261         HLCD_vSendData( '-' ) ;
00262
00263         A_s32Num = A_s32Num * ( -1 ) ;
00264     }
00265
00266     // In case the number contains zeros.
00267     if( A_s32Num == 0 )
00268     {
00269         HLCD_vSendData( '0' ) ;
00270     }
00271
00272     // Save reversed digits in the array.
00273     for( L_u8Counter = 0 ; A_s32Num != 0 ; L_u8Counter++ )
00274     {
00275
00276         LR_u8Digits[L_u8Counter] = A_s32Num % 10 ;
00277
00278         A_s32Num /= 10 ;
00279     }
00280
00281     // For arrangement the digits.
00282     for ( VAR(s8_t) L_s8ArrangedNums = (L_u8Counter-1) ; L_s8ArrangedNums >= 0 ; L_s8ArrangedNums-- )
00283     {
00284
00285         HLCD_vSendData( '0' + LR_u8Digits[ L_s8ArrangedNums ] ) ;
00286
00287     }
00288 }
```

References [HLCD_vSendData\(\)](#), and [VAR](#).

7.34.1.7 LCD_vSaveCustomChar()

```
void LCD_vSaveCustomChar (
    u8_t A_u8Address,
    u8_t * A_u8CustomChar )
```

Definition at line 333 of file [LCD_program.c](#).

```
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( VAR(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 }
```

References [FirstByteInCGRAM_Pattern](#), [LCD_vSendCommand\(\)](#), [LCD_vSendData\(\)](#), [LastByteInCGRAM_Pattern](#), [NumOf_CGRAM_Patterns](#), [SET_CGRAM_AC_MASK](#), [SET_DDRAM_AC_MASK](#), and [VAR](#).

7.34.1.8 LCD_vGoTo()

```
void LCD_vGoTo (
    u8_t A_u8Row,
    u8_t A_u8Col )
```

Definition at line 355 of file [LCD_program.c](#).

```
00356 {
00357
00358     // Valid Range.
00359     if ( (A_u8Row <= MAX_IDX_OF_ROWS) && (A_u8Col <= MAX_IDX_OF_COL) )
00360     {
00361
00362         switch( A_u8Row )
00363         {
00364
00365             case 0: LCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + FIRST_ROW_START ) ; break;
00366
00367             case 1: LCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + SEC_ROW_START ) ; break;
00368
00369         }
00370     }
00371
00372     else
00373     {
00374
00375         // Do Nothing
00376     }
00377
00378 }
```

References [FIRST_ROW_START](#), [LCD_vSendCommand\(\)](#), [MAX_IDX_OF_COL](#), [MAX_IDX_OF_ROWS](#), [SEC_ROW_START](#), and [SET_DDRAM_AC_MASK](#).

Referenced by [LCD_vClearChar\(\)](#).

7.34.1.9 HLCD_vSetShiftLeftOn()

```
void HLCD_vSetShiftLeftOn (
    void )
```

Definition at line 385 of file [LCD_program.c](#).

```
00386 {
00387     MSysTick_vDelayMilliSec( 500 ) ;
00388     HLCD_vSendCommand( HLCD_EntryModeSet_ShiftLeftOn_Cmd ) ;
00389     MSysTick_vDelayMilliSec( 500 ) ;
00390     MSysTick_vDelayMilliSec( 500 ) ;
00391     MSysTick_vDelayMilliSec( 500 ) ;
00392     MSysTick_vDelayMilliSec( 500 ) ;
00393 }
00394 }
```

References [HLCD_EntryModeSet_ShiftLeftOn_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

Referenced by [HLCD_vDispShiftLeftString\(\)](#).

7.34.1.10 HLCD_vSetDispOFF()

```
void HLCD_vSetDispOFF (
    void )
```

Definition at line 400 of file [LCD_program.c](#).

```
00401 {
00402     MSysTick_vDelayMilliSec( 500 ) ;
00403     HLCD_vSendCommand( HLCD_DispOff_Cmd ) ;
00404     MSysTick_vDelayMilliSec( 500 ) ;
00405     MSysTick_vDelayMilliSec( 500 ) ;
00406     MSysTick_vDelayMilliSec( 500 ) ;
00407     MSysTick_vDelayMilliSec( 500 ) ;
00408 }
00409 }
```

References [HLCD_DispOff_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.34.1.11 HLCD_vDispCursorWithBlinking()

```
void HLCD_vDispCursorWithBlinking (
    void )
```

Definition at line 415 of file [LCD_program.c](#).

```
00416 {
00417     MSysTick_vDelayMilliSec( 500 ) ;
00418     HLCD_vSendCommand( HLCD_DispCursorWithBlinking_Cmd ) ;
00419     MSysTick_vDelayMilliSec( 500 ) ;
00420     HLCD_vSendCommand( HLCD_DispCursorWithBlinking_Cmd ) ;
00421     MSysTick_vDelayMilliSec( 500 ) ;
00422     MSysTick_vDelayMilliSec( 500 ) ;
00423 }
00424 }
```

References [HLCD_DispCursorWithBlinking_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.34.1.12 HLCD_vSetCursorBlinkingOFF()

```
void HLCD_vSetCursorBlinkingOFF (
    void )
```

Definition at line 429 of file [LCD_program.c](#).

```
00430 {
00431     MSysTick_vDelayMilliSec( 500 ) ;
00432     HLCD_vSendCommand( HLCD_SetCursorBlinkingOFF_Cmd ) ;
00433     MSysTick_vDelayMilliSec( 500 ) ;
00434     MSysTick_vDelayMilliSec( 500 ) ;
00435 }
```

References [HLCD_SetCursorBlinkingOFF_Cmd](#), [HLCD_vSendCommand\(\)](#), and [MSysTick_vDelayMilliSec\(\)](#).

7.34.1.13 HLCD_vDispShiftLeftString()

```
void HLCD_vDispShiftLeftString (
    c8_t * A_c8Char )
```

Definition at line 443 of file [LCD_program.c](#).

```
00444 {
00445     VAR(u8_t) L_u8I = 0 ;
00446     for( L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00447     {
00448         HLCD_vSendData( A_c8Char[L_u8I] ) ;
00449     }
00450     MSysTick_vDelayMilliSec( 500 ) ;
00451     if( L_u8I > HLCD_CharactersNums )
00452     {
00453         for( VAR(u8_t) i = 0 ; i < L_u8I - HLCD_CharactersNums ; i++ )
00454         {
00455             HLCD_vSetShiftLeftOn() ;
00456         }
00457     }
00458 }
```

References [HLCD_CharactersNums](#), [HLCD_vSendData\(\)](#), [HLCD_vSetShiftLeftOn\(\)](#), [MSysTick_vDelayMilliSec\(\)](#), and [VAR](#).

7.34.1.14 HLCD_vClearChar()

```
void HLCD_vClearChar (
    u8_t A_u8Row,
    u8_t A_u8Col )
```

Definition at line 474 of file [LCD_program.c](#).

```
00475 {
00476     HLCD_vGoTo( A_u8Row, A_u8Col ) ;
00477     HLCD_vSendData( ' ' ) ;
00478 }
```

References [HLCD_vGoTo\(\)](#), and [HLCD_vSendData\(\)](#).

7.35 LCD_program.c

[Go to the documentation of this file.](#)

```

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 /***** Functions implementations *****/
00027 /*           Functions implementations           */
00028 /***** Functions implementations *****/
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         .Port      = LCD_DATA_PORT, .Pin = D2_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00081             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00082             .InputType = GPIOx_NoPull
00083     } ;
00084
00085     MGPIOx_ConfigType D3 =
00086     {
00087         .Port      = LCD_DATA_PORT, .Pin = D3_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00088             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00089             .InputType = GPIOx_NoPull
00090
00091     } ;
00092
00093     MGPIOx_ConfigType D4 =
00094     {
00095         .Port      = LCD_DATA_PORT, .Pin = D4_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00096             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00097             .InputType = GPIOx_NoPull
00098
00099     } ;
00100
00101     MGPIOx_ConfigType D5 =
00102     {
00103         .Port      = LCD_DATA_PORT, .Pin = D5_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00104             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00105             .InputType = GPIOx_NoPull
00106
00107     } ;
00108
00109     MGPIOx_ConfigType D6 =
00110     {
00111         .Port      = LCD_DATA_PORT, .Pin = D6_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00112             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00113             .InputType = GPIOx_NoPull
00114
00115     } ;
00116
00117     MGPIOx_ConfigType D7 =
00118     {
00119         .Port      = LCD_DATA_PORT, .Pin = D7_PIN, .Mode = GPIOx_MODE_OUTPUT, .OutputType =
00120             GPIOx_PUSH_PULL, .OutputSpeed = GPIOx_LowSpeed,
00121             .InputType = GPIOx_NoPull
00122
00123     } ;
00124
00125     MGPIOx_vInit( &RS ) ;
00126     MGPIOx_vInit( &RW ) ;
00127     MGPIOx_vInit( &EN ) ;
00128     MGPIOx_vInit( &D0 ) ;
00129     MGPIOx_vInit( &D1 ) ;
00130     MGPIOx_vInit( &D2 ) ;
00131     MGPIOx_vInit( &D3 ) ;
00132     MGPIOx_vInit( &D4 ) ;
00133     MGPIOx_vInit( &D5 ) ;
00134     MGPIOx_vInit( &D6 ) ;
00135     MGPIOx_vInit( &D7 ) ;
00136
00137     // Start initialization sequence.
00138     MSysTick_vDelayMilliSec( 50 ) ;
00139
00140     LCD_vSendCommand( LCD_FuctionSet_Cmd ) ;
00141
00142     MSysTick_vDelayMilliSec( 1 ) ;
00143
00144     LCD_vSendCommand( LCD_DisponOffCTRL_Cmd ) ;
00145
00146     MSysTick_vDelayMilliSec( 1 ) ;
00147
00148     LCD_vSendCommand( LCD_Dispclear_Cmd ) ;
00149
00150     MSysTick_vDelayMilliSec( 3 ) ;
00151
00152     LCD_vSendCommand( LCD_EntryModeSet_Cmd ) ;
00153
00154     MSysTick_vDelayMilliSec( 1 ) ;
00155
00156 }
00157

```

```
00159 //*****
00160 //*****
00161
00162 void LCD_vSendCommand( VAR(u8_t) A_u8Cmd )
00163 {
00164
00165     // Set RS pin low to send a command.
00166     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_RS_PIN, GPIOx_LOW ) ;
00167
00168     // Set RW pin low to write the command.
00169     GPIOx_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_vSetPortValue( LCD_DATA_PORT, A_u8Cmd ) ;
00175
00176     MSysTick_vDelayMilliSec( 1 ) ;
00177
00178     // Set the enable pin high.
00179     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_HIGH ) ;
00180
00181     MSysTick_vDelayMilliSec( 1 ) ;
00182
00183     // Set the enable pin low.
00184     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_LOW ) ;
00185
00186     MSysTick_vDelayMilliSec( 1 ) ;
00187
00188 }
00189
00190 //*****
00191 //*****
00192
00193 void LCD_vSendData(VAR(u8_t) A_u8Data)
00194 {
00195
00196     // Set RS pin high to send data.
00197     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_RS_PIN, GPIOx_HIGH ) ;
00198
00199     // Set RW pin low to write the data.
00200     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_RW_PIN, GPIOx_LOW ) ;
00201
00202     MSysTick_vDelayMilliSec( 1 ) ;
00203
00204     // Put the data on the data bus.
00205     GPIO_vSetPortValue( LCD_DATA_PORT, A_u8Data ) ;
00206
00207     MSysTick_vDelayMilliSec( 1 ) ;
00208
00209     // Set the enable pin high.
00210     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_HIGH ) ;
00211
00212     MSysTick_vDelayMilliSec( 1 ) ;
00213
00214     // Set the enable pin low.
00215     GPIOx_vSetValue( LCD_CTRL_PORT, LCD_EN_PIN, GPIOx_LOW ) ;
00216
00217     MSysTick_vDelayMilliSec( 1 ) ;
00218
00219 }
00220
00221 //*****
00222 //*****
00223
00224 void LCD_vClear(void)
00225 {
00226
00227     MSysTick_vDelayMilliSec( 1 ) ;
00228
00229     LCD_vSendCommand( LCD_Dispclear_Cmd ) ;
00230
00231     MSysTick_vDelayMilliSec( 1 ) ;
00232
00233 }
00234
00235 //*****
00236 //*****
```

```

00238 void LCD_vDispString( P2VAR(c8_t) A_c8Char )
00239 {
00240
00241     for( VAR(u8_t) L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00242     {
00243         LCD_vSendData( A_c8Char[L_u8I] ) ;
00244     }
00245
00246 }
00247
00248 }
00249
00250
00251 //*****
00252 //*****
00253 void LCD_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 contains zeros.
00271     if( A_s32Num == 0 )
00272     {
00273
00274         LCD_vSendData( '0' ) ;
00275
00276     }
00277
00278     // Save reversed digits in the array.
00279     for( L_u8Counter = 0 ; A_s32Num != 0 ; L_u8Counter++ )
00280     {
00281
00282         LR_u8Digits[L_u8Counter] = A_s32Num % 10 ;
00283
00284         A_s32Num /= 10 ;
00285
00286     }
00287
00288     // For arrangement the digits.
00289     for( VAR(s8_t) L_s8ArrangedNums = (L_u8Counter-1) ; L_s8ArrangedNums >= 0 ; L_s8ArrangedNums-- )
00290     {
00291
00292         LCD_vSendData( '0' + LR_u8Digits[ L_s8ArrangedNums ] ) ;
00293
00294     }
00295 }
00296
00297
00298 //*****
00299 //*****
00300 //void LCD_vDispNum(s32 A_s32Num)
00301 //{
00302 //
00303 //
00304 //    u32 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 HLCD_vSaveCustomChar( VAR(u8_t) A_u8Address, P2VAR(u8_t) A_u8CustomChar )
00334 {
00335
00336 // 1-Set CGRAM Address.
00337 HLCD_vSendCommand( SET_CGRAM_AC_MASK + (NumOf_CGRAM_Patterns * A_u8Address) ) ;
00338
00339 // 2- Send custom char data.
00340 for( VAR(u8_t) L_u8I = FirstByteInCGRAM_Pattern ; L_u8I < LastByteInCGRAM_Pattern ; L_u8I++ )
00341 {
00342
00343     HLCD_vSendData( A_u8CustomChar[L_u8I] ) ;
00344
00345 }
00346
00347 // 3-Set DDRAM address
00348 HLCD_vSendCommand( SET_DDRAM_AC_MASK ) ;
00349
00350 }
00351
00352
00353 //*****
00354 //*****
00355 void HLCD_vGoTo( VAR(u8_t) A_u8Row, VAR(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
00362     switch( A_u8Row )
00363     {
00364
00365         case 0: HLCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + FIRST_ROW_START ) ; break;
00366
00367         case 1: HLCD_vSendCommand( SET_DDRAM_AC_MASK + A_u8Col + SEC_ROW_START ) ; break;
00368
00369     }
00370
00371 }
00372
00373 else
00374 {
00375
00376     // Do Nothing
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     MSysTick_vDelayMilliSec( 500 ) ;
00403     HLCD_vSendCommand( HLCD_DispOff_Cmd ) ;
00404     MSysTick_vDelayMilliSec( 500 ) ;
00405 }
00406
00407
00408
00409 }
00410
00411
00412 //*****
00413 //*****
00414
00415 void HLCD_vDispCursorWithBlinking(void)
00416 {
00417     MSysTick_vDelayMilliSec( 500 ) ;
00418     HLCD_vSendCommand( HLCD_DispCursorWithBlinking_Cmd ) ;
00419     MSysTick_vDelayMilliSec( 500 ) ;
00420
00421
00422
00423
00424 }
00425
00426
00427
00428
00429 void HLCD_vSetCursorBlinkingOFF(void)
00430 {
00431     MSysTick_vDelayMilliSec( 500 ) ;
00432     HLCD_vSendCommand( HLCD_SetCursorBlinkingOFF_Cmd ) ;
00433     MSysTick_vDelayMilliSec( 500 ) ;
00434
00435
00436
00437
00438 }
00439
00440
00441
00442
00443 void HLCD_vDispShiftLeftString( P2VAR(c8_t) A_c8Char )
00444 {
00445     VAR(u8_t) L_u8I = 0 ;
00446
00447     for( L_u8I = 0 ; A_c8Char[L_u8I] != '\0' ; L_u8I++ )
00448     {
00449
00450         HLCD_vSendData( A_c8Char[L_u8I] ) ;
00451
00452     }
00453
00454
00455     MSysTick_vDelayMilliSec( 500 ) ;
00456
00457     if( L_u8I > HLCD_CharactersNums )
00458     {
00459
00460         for( VAR(u8_t) i = 0 ; i < L_u8I - HLCD_CharactersNums ; i++ )
00461         {
00462
00463             HLCD_vSetShiftLeftOn() ;
00464
00465         }
00466
00467     }
00468
00469 }
00470
00471
00472
00473
00474 void HLCD_vClearChar( VAR(u8_t) A_u8Row, VAR(u8_t) A_u8Col )
00475 {
00476
00477     HLCD_vGoTo( A_u8Row, A_u8Col ) ;
00478

```

```

00479     LCD_vSendData( ' ' );
00480
00481 }
00482
00483
00484 /*****
00485 ****
00486 ****
00487 ****
00488 ****
00489 ****
00490 ****
00491 ****
00492 ****
00493

```

7.36 COTS/HAL/LDR/LDR_config.c File Reference

```

#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/ADC/ADC_interface.h"
#include "LDR_config.h"

```

Variables

- MGPIox_ConfigType LDR_Vo
- u8_t LDR_CHANNEL = 0

7.36.1 Variable Documentation

7.36.1.1 LDR_Vo

[MGPIox_ConfigType](#) LDR_Vo

Initial value:

```

= {
    .Port      =  (0)      ,
    .Pin       =  (0)      ,
    .Mode      =  (0b11)   ,
}

```

Definition at line 26 of file [LDR_config.c](#).

Referenced by [HLDR_vInit\(\)](#).

7.36.1.2 LDR_CHANNEL

`u8_t LDR_CHANNEL = 0`

Definition at line 41 of file [LDR_config.c](#).

Referenced by [HLDR_u16DigitalOutputValue\(\)](#).

7.37 LDR_config.c

[Go to the documentation of this file.](#)

```

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 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 VAR(u8_t) LDR_CHANNEL = CHANNEL0 ;
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052

```

7.38 COTS/HAL/LDR/LDR_config.h File Reference

Macros

- `#define LDR_Vo_PORT GPIO_PORTA`
- `#define LDR_Vo_PIN GPIOx_PIN0`

7.38.1 Macro Definition Documentation

7.38.1.1 LDR_Vo_PORT

```
#define LDR_Vo_PORT GPIO_PORTA
```

Definition at line 13 of file [LDR_config.h](#).

7.38.1.2 LDR_Vo_PIN

```
#define LDR_Vo_PIN GPIOx_PIN0
```

Definition at line 15 of file [LDR_config.h](#).

7.39 LDR_config.h

[Go to the documentation of this file.](#)

```
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_PORTA
00014
00015 #define LDR_Vo_PIN GPIOx_PIN0
00016
00017
00018
00019 #endif // _LDR_config_H
```

7.40 COTS/HAL/LDR/LDR_interface.h File Reference

Macros

- `#define DARK 3500`
- `#define VERY_DIM 3000`
- `#define MODERATE 2000`
- `#define OVERCAST 1500`
- `#define VERY_BRIGHT 1000`

Functions

- `void HLDR_vInit (void)`
- `u16_t HLDR_u16DigitalOutputValue (void)`

7.40.1 Macro Definition Documentation

7.40.1.1 DARK

```
#define DARK 3500
```

Definition at line 14 of file [LDR_interface.h](#).

7.40.1.2 VERY_DIM

```
#define VERY_DIM 3000
```

Definition at line 16 of file [LDR_interface.h](#).

7.40.1.3 MODERATE

```
#define MODERATE 2000
```

Definition at line 18 of file [LDR_interface.h](#).

7.40.1.4 OVERCAST

```
#define OVERCAST 1500
```

Definition at line 20 of file [LDR_interface.h](#).

7.40.1.5 VERY_BRIGHT

```
#define VERY_BRIGHT 1000
```

Definition at line 22 of file [LDR_interface.h](#).

7.40.2 Function Documentation

7.40.2.1 HLDR_vInit()

```
void HLDR_vInit (
    void )
```

Definition at line 36 of file [LDR_program.c](#).

```
00037 {
00038
00039     MGPIox_vInit( &LDR_Vo ) ;
00040
00041     MADC_vInit( ) ;
00042
00043 }
```

References [LDR_Vo](#), [MADC_vInit\(\)](#), and [MGPIox_vInit\(\)](#).

7.40.2.2 HLDR_u16DigitalOutputValue()

```
u16_t HLDR_u16DigitalOutputValue (
    void )
```

Definition at line 48 of file [LDR_program.c](#).

```
00049 {
00050
00051     VAR(u16_t) L_u16DigitalResult = INITIAL_ZERO ;
00052
00053     L_u16DigitalResult = MADC_u16ConvertToDigital( LDR_CHANNEL ) ;
00054
00055     return L_u16DigitalResult ;
00056
00057 }
```

References [INITIAL_ZERO](#), [LDR_CHANNEL](#), [MADC_u16ConvertToDigital\(\)](#), and [VAR](#).

7.41 LDR_interface.h

[Go to the documentation of this file.](#)

```
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 /***** Functions prototypes         */
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
```

7.42 COTS/HAL/LDR/LDR_private.h File Reference

7.43 LDR_private.h

[Go to the documentation of this file.](#)

```
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
```

7.44 COTS/HAL/LDR/LDR_program.c File Reference

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../MCAL/RCC/MRCC_interface.h"
#include "../../MCAL/GPIO/GPIO_interface.h"
#include "../../MCAL/ADC/ADC_interface.h"
#include "LDR_interface.h"
#include "LDR_private.h"
#include "LDR_config.h"
```

Functions

- void [HLDR_vInit](#) (void)
- [u16_t HLDR_u16DigitalOutputValue](#) (void)

Variables

- [MGPIOx_ConfigType LDR_Vo](#)
- [u8_t LDR_CHANNEL](#)

7.44.1 Function Documentation

7.44.1.1 HLDR_vInit()

```
void HLDR_vInit (
    void )
```

Definition at line 36 of file [LDR_program.c](#).

```
00037 {
00038
00039     MGPIox_vInit( &LDR_Vo ) ;
00040
00041     MADC_vInit( ) ;
00042
00043 }
```

References [LDR_Vo](#), [MADC_vInit\(\)](#), and [MGPIox_vInit\(\)](#).

7.44.1.2 HLDR_u16DigitalOutputValue()

```
u16_t HLDR_u16DigitalOutputValue (
    void )
```

Definition at line 48 of file [LDR_program.c](#).

```
00049 {
00050
00051     VAR(u16_t) L_u16DigitalResult = INITIAL_ZERO ;
00052
00053     L_u16DigitalResult = MADC_u16ConvertToDigital( LDR_CHANNEL ) ;
00054
00055     return L_u16DigitalResult ;
00056
00057 }
```

References [INITIAL_ZERO](#), [LDR_CHANNEL](#), [MADC_u16ConvertToDigital\(\)](#), and [VAR](#).

7.44.2 Variable Documentation

7.44.2.1 LDR_Vo

```
MGPIox_ConfigType LDR_Vo [extern]
```

Definition at line 26 of file [LDR_config.c](#).

Referenced by [HLDR_vInit\(\)](#).

7.44.2.2 LDR_CHANNEL

```
u8_t LDR_CHANNEL [extern]
```

Definition at line 41 of file [LDR_config.c](#).

Referenced by [HLDR_u16DigitalOutputValue\(\)](#).

7.45 LDR_program.c

[Go to the documentation of this file.](#)

```

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 MGPIOx_ConfigType LDR_Vo      ;
00029
00030 extern VAR(u8_t) LDR_CHANNEL        ;
00031
00032 /***** Functions' implementations *****/
00033 /*           *
00034 /***** */
00035
00036 void HLDR_vInit( void )
00037 {
00038     MGPIOx_vInit( &LDR_Vo ) ;
00039
00040     MADC_vInit( ) ;
00041
00042 }
00043
00044
00045
00046 /***** */
00047
00048 u16_t HLDR_u16DigitalOutputValue( void )
00049 {
00050
00051     VAR(u16_t) L_u16DigitalResult = INITIAL_ZERO ;
00052
00053     L_u16DigitalResult = MADC_u16ConvertToDigital( LDR_CHANNEL ) ;
00054
00055     return L_u16DigitalResult ;
00056
00057 }
00058
00059
00060 /***** */
00061
00062
00063
00064
00065
00066
00067
00068

```

7.46 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))

7.46.1 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](#).

7.47 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_ */

```

7.48 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

7.48.1 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](#).

7.49 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_ */
```

7.50 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);

7.50.1 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](#).

7.51 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_ */
```

7.52 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](#)

7.52.1 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](#).

7.53 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_ */
```

7.54 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)

7.54.1 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](#).

7.55 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
00060 #define FLAG_SET (1)
00061 #endif /* ifndef FLAG_SET */
00062
00063 #ifndef FLAG_CLEARED
00064 #define FLAG_CLEARED (0)
00065 #endif /* ifndef FLAG_CLEARED */
00066
00067 #ifndef RUN
00068 #define RUN (1)
00069 #endif /* ifndef RUN */
00070
00071 #ifndef STOP
00072 #define STOP (0)
00073 #endif /* ifndef STOP */
00074
00075 #ifndef PRESSED
00076 #define PRESSED (1)
00077 #endif /* ifndef PRESSED */
00078
00079 #ifndef RELEASED
00080 #define RELEASED (0)
00081 #endif /* ifndef RELEASED */
00082
00083 #ifndef SAME_STRING
00084 #define SAME_STRING (0)
00085 #endif /* ifndef SAME_STRING */
00086
00087 #ifndef DIFFERENT_STRING
00088 #define DIFFERENT_STRING (1)
00089 #endif /* ifndef DIFFERENT_STRING */
00090
00091 #ifndef COTS_LIB_LSTD_VALUES_H_
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
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136 #endif /* COTS_LIB_LSTD_VALUES_H_ */

```

7.56 COTS/MCAL/ADC/ADC_config.h File Reference

Macros

- #define OVERRUN_INT_DISABLE
- #define RESOLUTION_12_BITS
- #define ANALOG_WDT_DISABLE
- #define INJ_ANALOG_WDT_NOTBYBASED
- #define DISC_CHANNEL_NUM_1_CHANNEL
- #define INJ_DISC_MODE_DISABLE
- #define DISC_MODE_DISABLE
- #define INJ_AUTO_MODE_DISABLE
- #define AWDSGL_MODE_DISABLE
- #define SCAN_MODE_DISABLE
- #define JEOCIE_MODE_DISABLE
- #define AWDIE_MODE_DISABLE
- #define EOCIE_MODE_DISABLE
- #define AWD_CHANNEL_ZERO
- #define REG_START_CONV_ENABLE
- #define REG_EXTERNAL_TRIGGER_DISABLE
- #define REG_EXT_EVENT_SRC EXTI_LINE11
- #define INJ_START_CONV_DISABLE
- #define INJ_EXTERNAL_TRIGGER_DISABLE
- #define INJ_EXT_EVENT_SRC EXTI_LINE15
- #define DATA_ALIGN RIGHT
- #define EOC_SELECTION REG_CONV
- #define DMA_MODE_DISABLE

- #define DMA_DIS_SELECTION_DISABLE
- #define CONT_CONV_ENABLE
- #define ADC_ON_ENABLE
- #define SMP10_3_CYCLES
- #define SMP11_3_CYCLES
- #define SMP12_3_CYCLES
- #define SMP13_3_CYCLES
- #define SMP14_3_CYCLES
- #define SMP15_3_CYCLES
- #define SMP16_3_CYCLES
- #define SMP17_3_CYCLES
- #define SMP18_3_CYCLES
- #define SMP0_3_CYCLES
- #define SMP1_3_CYCLES
- #define SMP2_3_CYCLES
- #define SMP3_3_CYCLES
- #define SMP4_3_CYCLES
- #define SMP5_3_CYCLES
- #define SMP6_3_CYCLES
- #define SMP7_3_CYCLES
- #define SMP8_3_CYCLES
- #define SMP9_3_CYCLES
- #define REG_SQ_LENGTH_1_CONV
- #define TEMP_SENSOR_AND_VREF_DISABLE
- #define V_BATTERY_DISABLE
- #define ADC_PRESCALER_DIV_BY_2
- #define SQ1_BIT_MANIPULATION 0xFFFFFE0

7.56.1 Macro Definition Documentation

7.56.1.1 OVERRUN_INT

```
#define OVERRUN_INT DISABLE
```

Definition at line 18 of file [ADC_config.h](#).

7.56.1.2 RESOLUTION

```
#define RESOLUTION _12_BITS
```

Definition at line 28 of file [ADC_config.h](#).

7.56.1.3 ANALOG_WDT

```
#define ANALOG_WDT DISABLE
```

Definition at line 36 of file [ADC_config.h](#).

7.56.1.4 INJ_ANALOG_WDT

```
#define INJ_ANALOG_WDT NOTBYBASED
```

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

7.56.1.5 DISC_CHANNEL_NUM

```
#define DISC_CHANNEL_NUM _1_CHANNEL
```

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

7.56.1.6 INJ_DISC_MODE

```
#define INJ_DISC_MODE DISABLE
```

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

7.56.1.7 DISC_MODE

```
#define DISC_MODE DISABLE
```

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

7.56.1.8 INJ_AUTO_MODE

```
#define INJ_AUTO_MODE DISABLE
```

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

7.56.1.9 AWDSGL_MODE

```
#define AWDSGL_MODE DISABLE
```

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

7.56.1.10 SCAN_MODE

```
#define SCAN_MODE DISABLE
```

Definition at line 98 of file [ADC_config.h](#).

7.56.1.11 JEOCIE_MODE

```
#define JEOCIE_MODE DISABLE
```

Definition at line 106 of file [ADC_config.h](#).

7.56.1.12 AWDIE_MODE

```
#define AWDIE_MODE DISABLE
```

Definition at line 114 of file [ADC_config.h](#).

7.56.1.13 EOCIE_MODE

```
#define EOCIE_MODE DISABLE
```

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

7.56.1.14 AWD_CHANNEL

```
#define AWD_CHANNEL ZERO
```

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

7.56.1.15 REG_START_CONV

```
#define REG_START_CONV ENABLE
```

Definition at line 156 of file [ADC_config.h](#).

7.56.1.16 REG_EXTERNAL_TRIGGER

```
#define REG_EXTERNAL_TRIGGER DISABLE
```

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

7.56.1.17 REG_EXT_EVENT_SRC

```
#define REG_EXT_EVENT_SRC EXTI_LINE11
```

Definition at line 186 of file [ADC_config.h](#).

7.56.1.18 INJ_START_CONV

```
#define INJ_START_CONV DISABLE
```

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

7.56.1.19 INJ_EXTERNAL_TRIGGER

```
#define INJ_EXTERNAL_TRIGGER DISABLE
```

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

7.56.1.20 INJ_EXT_EVENT_SRC

```
#define INJ_EXT_EVENT_SRC EXTI_LINE15
```

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

7.56.1.21 DATA_ALIGN

```
#define DATA_ALIGN RIGHT
```

Definition at line 231 of file [ADC_config.h](#).

7.56.1.22 EOC_SELECTION

```
#define EOC_SELECTION REG_CONV
```

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

7.56.1.23 DMA_MODE

```
#define DMA_MODE DISABLE
```

Definition at line 247 of file [ADC_config.h](#).

7.56.1.24 DMA_DIS_SELECTION

```
#define DMA_DIS_SELECTION DISABLE
```

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

7.56.1.25 CONT_CONV

```
#define CONT_CONV ENABLE
```

Definition at line 263 of file [ADC_config.h](#).

7.56.1.26 ADC_ON

```
#define ADC_ON ENABLE
```

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

7.56.1.27 SMP10

```
#define SMP10 _3_CYCLES
```

Definition at line [287](#) of file [ADC_config.h](#).

7.56.1.28 SMP11

```
#define SMP11 _3_CYCLES
```

Definition at line [301](#) of file [ADC_config.h](#).

7.56.1.29 SMP12

```
#define SMP12 _3_CYCLES
```

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

7.56.1.30 SMP13

```
#define SMP13 _3_CYCLES
```

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

7.56.1.31 SMP14

```
#define SMP14 _3_CYCLES
```

Definition at line [343](#) of file [ADC_config.h](#).

7.56.1.32 SMP15

```
#define SMP15 _3_CYCLES
```

Definition at line [357](#) of file [ADC_config.h](#).

7.56.1.33 SMP16

```
#define SMP16 _3_CYCLES
```

Definition at line 371 of file [ADC_config.h](#).

7.56.1.34 SMP17

```
#define SMP17 _3_CYCLES
```

Definition at line 385 of file [ADC_config.h](#).

7.56.1.35 SMP18

```
#define SMP18 _3_CYCLES
```

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

7.56.1.36 SMP0

```
#define SMP0 _3_CYCLES
```

Definition at line 415 of file [ADC_config.h](#).

7.56.1.37 SMP1

```
#define SMP1 _3_CYCLES
```

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

7.56.1.38 SMP2

```
#define SMP2 _3_CYCLES
```

Definition at line 443 of file [ADC_config.h](#).

7.56.1.39 SMP3

```
#define SMP3 _3_CYCLES
```

Definition at line [457](#) of file [ADC_config.h](#).

7.56.1.40 SMP4

```
#define SMP4 _3_CYCLES
```

Definition at line [471](#) of file [ADC_config.h](#).

7.56.1.41 SMP5

```
#define SMP5 _3_CYCLES
```

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

7.56.1.42 SMP6

```
#define SMP6 _3_CYCLES
```

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

7.56.1.43 SMP7

```
#define SMP7 _3_CYCLES
```

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

7.56.1.44 SMP8

```
#define SMP8 _3_CYCLES
```

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

7.56.1.45 SMP9

```
#define SMP9 _3_CYCLES
```

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

7.56.1.46 REG_SQ_LENGTH

```
#define REG_SQ_LENGTH _1_CONV
```

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

7.56.1.47 TEMP_SENSOR_AND_VREF

```
#define TEMP_SENSOR_AND_VREF DISABLE
```

Definition at line 572 of file [ADC_config.h](#).

7.56.1.48 V_BATTERY

```
#define V_BATTERY DISABLE
```

Definition at line 580 of file [ADC_config.h](#).

7.56.1.49 ADC_PRESCALER

```
#define ADC_PRESCALER DIV_BY_2
```

Definition at line 590 of file [ADC_config.h](#).

7.56.1.50 SQ1_BIT_MANIPULATION

```
#define SQ1_BIT_MANIPULATION 0xFFFFFE0
```

Definition at line 596 of file [ADC_config.h](#).

7.57 ADC_config.h

[Go to the documentation of this file.](#)

```

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 /***** ADC_CR1 configurations *****/
00021
00022 /*options:
00023 *_12_BITS
00024 *_10_BITS
00025 *_8_BITS
00026 *_6_BITS
00027 */
00028 #define RESOLUTION _12_BITS
00029
00030 /***** ADC_CR1 configurations *****/
00031
00032 /*options:
00033 *ENABLE
00034 *DISABLE
00035 */
00036 #define ANALOG_WDT DISABLE
00037
00038 /***** ADC_CR1 configurations *****/
00039
00040 /*options:
00041 *ENABLE
00042 *DISABLE
00043 */
00044 #define INJ_ANALOG_WDT NOTBYBASED
00045
00046 /***** ADC_CR1 configurations *****/
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 /***** ADC_CR1 configurations *****/
00061
00062 /*options:
00063 *ENABLE
00064 *DISABLE
00065 */
00066 #define INJ_DISC_MODE DISABLE
00067
00068 /***** ADC_CR1 configurations *****/
00069
00070 /*options:
00071 *ENABLE
00072 *DISABLE
00073 */
00074 #define DISC_MODE DISABLE
00075
00076 /***** ADC_CR1 configurations *****/
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 AWDSDL_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 *THIRTEEN
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_CC2
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_CC9
00211 *T2_CC1
00212 *INJ_T2_TRGO
00213 *T3_CC2
00214 *T3_CC4
00215 *T4_CC1
00216 *T4_CC2
00217 *T4_CC3
00218 *T4_CC9
00219 *T5_CC4
00220 *T5_CC9
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 /*****
00530
00531 /*options:
00532 *_3_CYCLES
00533 *_15_CYCLES
00534 *_28_CYCLES
00535 *_56_CYCLES
00536 *_84_CYCLES
00537 *_112_CYCLES
00538 *_144_CYCLES
00539 *_480_CYCLES
00540 */
00541 #define SMP9 _3_CYCLES
00542
00543 /*****
00544 /* ADC_SQRL configurations */
00545 /*****
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 /*****
00565 /* ADC_CCR configurations */
00566 /*****
00567
00568 /*options:
00569 *ENABLE
00570 *DISABLE
00571 */
00572 #define TEMP_SENSOR_AND_VREF DISABLE
00573
00574 /*****
00575
00576 /*options:
00577 *ENABLE
00578 *DISABLE
00579 */
00580 #define V_BATTERY DISABLE
00581
00582 /*****
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 /* Bit Manipulation Macros */
00594 /*****
00595
00596 #define SQ1_BIT_MANIPULATION 0xFFFFFE0
00597
00598
00599
00600
00601
00602
00603 #endif // _ADC_config_H

```

7.58 COTS/MCAL/ADC/ADC_interface.h File Reference

Data Structures

- struct [MADC_ConfigType](#)

Macros

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

Functions

- void [MADC_vInit](#) (void)
- [u16_t MADC_u16ConvertToDigital](#) ([VAR\(u8_t\)](#) A_u8ChannelNum)
- void [MADC_vDisable](#) (void)
- void [MADC_vEnable](#) (void)
- void [MADC_vRegINT_Disable](#) (void)
- void [MADC_vRegINTEnable](#) (void)
- void [MADC_vSelectMode](#) ([VAR\(u8_t\)](#) A_u8ModeNum)
- void [MADC_vSelectChannel](#) ([VAR\(u8_t\)](#) A_u8ChannelNum)
- void [MADC_vStartConversion](#) ([VAR\(u8_t\)](#) A_u8ChannelNum)
- void [MADC_vSelectChannelsOrder](#) ([VAR\(u8_t\)](#) A_u8ChannelOrder)
- void [MADC_vSelectSampleTime](#) ([VAR\(u8_t\)](#) A_u8ChannelNum, [VAR\(u8_t\)](#) A_u8SampleTime)
- void [MADC_vSelectResolution](#) ([VAR\(u8_t\)](#) A_u8Resolution)
- [u16_t MADC_u16GetADCData](#) (void)
- void [MADC_vInitStruct](#) ([P2VAR\(MADC_ConfigType\)](#) A_ADCConfig)
- void [MADC_vSetCallBack](#) (void(*[MEXTI_vpPointerTo_ISR_function](#))(void))

7.58.1 Macro Definition Documentation

7.58.1.1 CHANNEL0

```
#define CHANNEL0 0
```

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

7.58.1.2 CHANNEL1

```
#define CHANNEL1 1
```

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

7.58.1.3 CHANNEL2

```
#define CHANNEL2 2
```

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

7.58.1.4 CHANNEL3

```
#define CHANNEL3 3
```

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

7.58.1.5 CHANNEL4

```
#define CHANNEL4 4
```

Definition at line [75](#) of file [ADC_interface.h](#).

7.58.1.6 CHANNEL5

```
#define CHANNEL5 5
```

Definition at line [76](#) of file [ADC_interface.h](#).

7.58.1.7 CHANNEL6

```
#define CHANNEL6 6
```

Definition at line [77](#) of file [ADC_interface.h](#).

7.58.1.8 CHANNEL7

```
#define CHANNEL7 7
```

Definition at line [78](#) of file [ADC_interface.h](#).

7.58.1.9 CHANNEL8

```
#define CHANNEL8 8
```

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

7.58.1.10 CHANNEL9

```
#define CHANNEL9 9
```

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

7.58.1.11 CHANNEL10

```
#define CHANNEL10 10
```

Definition at line [81](#) of file [ADC_interface.h](#).

7.58.1.12 CHANNEL11

```
#define CHANNEL11 11
```

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

7.58.1.13 CHANNEL12

```
#define CHANNEL12 12
```

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

7.58.1.14 CHANNEL13

```
#define CHANNEL13 13
```

Definition at line [84](#) of file [ADC_interface.h](#).

7.58.1.15 CHANNEL14

```
#define CHANNEL14 14
```

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

7.58.1.16 CHANNEL15

```
#define CHANNEL15 15
```

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

7.58.1.17 CHANNEL16

```
#define CHANNEL16 16
```

Definition at line [87](#) of file [ADC_interface.h](#).

7.58.1.18 CHANNEL17

```
#define CHANNEL17 17
```

Definition at line [88](#) of file [ADC_interface.h](#).

7.58.1.19 CHANNEL18

```
#define CHANNEL18 18
```

Definition at line 89 of file [ADC_interface.h](#).

7.58.2 Function Documentation

7.58.2.1 MADC_vInit()

```
void MADC_vInit (
    void )
```

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     MRC_C_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
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
00107 // EN/DIS Injected INT:
00108 #if JEOCIE_MODE == ENABLE
00109
00110     SET_BIT( ADC->CR1, JEOCIE ) ;
00111
00112 #elif JEOCIE_MODE == DISABLE
00113
00114     CLR_BIT( ADC->CR1, JEOCIE ) ;
00115
00116 #endif
00117
00118 // EN/DIS Regular INT:
00119 #if EOCIE_MODE == ENABLE
00120
00121     SET_BIT( ADC->CR1, EOCIE ) ;
00122
00123 #elif EOCIE_MODE == DISABLE
00124
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     SET_BIT( ADC->CR1, DISCEN ) ;
00269
00270 #elif DISC_MODE == DISABLE
00271     CLR_BIT( ADC->CR1, DISCEN ) ;
00272
00273 #endif
00274
00275 // Discontinuous mode channel count (1/2/3/4/.../8):
00276 #if DISC_CHANNEL_NUM == _1_CHANNEL
00277
00278     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00279
00280     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00281
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
00289     CLR_BIT( ADC->CR1, DISCNUM2 ) ;
00290
00291 #elif DISC_CHANNEL_NUM == _3_CHANNELS
00292
00293     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00294     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00295
00296 #elif DISC_CHANNEL_NUM == _4_CHANNELS
00297
00298     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00299     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00300
00301 #elif DISC_CHANNEL_NUM == _5_CHANNELS
00302
00303     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00304     CLR_BIT( ADC->CR1, DISCNUM1 ) ;
00305
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
00313     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00314
00315 #elif DISC_CHANNEL_NUM == _7_CHANNELS
00316
00317     CLR_BIT( ADC->CR1, DISCNUM0 ) ;
00318
00319 #elif DISC_CHANNEL_NUM == _8_CHANNELS
00320
00321     SET_BIT( ADC->CR1, DISCNUM0 ) ;
00322     SET_BIT( ADC->CR1, DISCNUM1 ) ;
00323
00324     SET_BIT( ADC->CR1, DISCNUM2 ) ;
00325
00326
00327 #endif
00328
00329 // EN/DIS Continuous mode:
00330 #if CONT_CONV == ENABLE
00331     SET_BIT( ADC->CR2, CONT ) ;
00332
00333 #elif CONT_CONV == DISABLE
00334     CLR_BIT( ADC->CR2, CONT ) ;
00335
00336 #endif
00337
00338
00339
00340 }

```

References [ADC](#), [ADCPRE0](#), [ADON](#), [ALIGN](#), [APB2ENR_ADC1EN](#), [CLR_BIT](#), [CONT](#), [DISCEN](#), [DISCNUM0](#), [DISCNUM1](#), [DISCNUM2](#), [EOCIE](#), [JEOCIE](#), [L0](#), [L1](#), [L2](#), [L3](#), [MRCC_vEnablePeriphralCLK\(\)](#), [OVRIE](#), [RCC_APB2](#), [RES0](#), [RES1](#), [SCAN](#), and [SET_BIT](#).

Referenced by [HLDR_vInit\(\)](#).

7.58.2.2 MADC_u16ConvertToDigital()

```
u16_t MADC_u16ConvertToDigital (
    VAR(u8_t) A_u8ChannelNum )
```

Referenced by [HLDR_u16DigitalOutputValue\(\)](#).

7.58.2.3 MADC_vDisable()

```
void MADC_vDisable (
    void )
```

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 [ADC](#), [ADON](#), [APB2ENR_ADC1EN](#), [CLR_BIT](#), [MRCC_vDisablePeriphralCLK\(\)](#), and [RCC_APB2](#).

7.58.2.4 MADC_vEnable()

```
void MADC_vEnable (
    void )
```

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 [ADC](#), [ADON](#), [APB2ENR_ADC1EN](#), [MRCC_vEnablePeriphralCLK\(\)](#), [RCC_APB2](#), and [SET_BIT](#).

7.58.2.5 MADC_vRegINT_Disable()

```
void MADC_vRegINT_Disable (
    void )
```

Definition at line 407 of file [ADC_program.c](#).
00408 {
00409 CLR_BIT(ADC->CR1, EOCIE) ;
00410 }

References [ADC](#), [CLR_BIT](#), and [EOCIE](#).

7.58.2.6 MADC_vRegINTEnable()

```
void MADC_vRegINTEnable (
    void )
```

Definition at line 415 of file [ADC_program.c](#).

```
00416 {
00417     SET_BIT( ADC->CR1, EOCIE ) ;
00418 }
```

References [ADC](#), [EOCIE](#), and [SET_BIT](#).

7.58.2.7 MADC_vSelectMode()

```
void MADC_vSelectMode (
    VAR(u8_t) A_u8ModeNum )
```

7.58.2.8 MADC_vSelectChannel()

```
void MADC_vSelectChannel (
    VAR(u8_t) A_u8ChannelNum )
```

7.58.2.9 MADC_vStartConversion()

```
void MADC_vStartConversion (
    VAR(u8_t) A_u8ChannelNum )
```

7.58.2.10 MADC_vSelectChannelsOrder()

```
void MADC_vSelectChannelsOrder (
    VAR(u8_t) A_u8ChannelOrder )
```

7.58.2.11 MADC_vSelectSampleTime()

```
void MADC_vSelectSampleTime (
    VAR(u8_t) A_u8ChannelNum,
    VAR(u8_t) A_u8SampleTime )
```

7.58.2.12 MADC_vSelectResolution()

```
void MADC_vSelectResolution (
    VAR(u8_t) A_u8Resolution )
```

7.58.2.13 MADC_u16GetADCData()

```
u16_t MADC_u16GetADCData (
    void )
```

7.58.2.14 MADC_vInitStruct()

```
void MADC_vInitStruct (
    P2VAR(MADC_ConfigType) A_ADCConfig )
```

7.58.2.15 MADC_vSetCallBack()

```
void MADC_vSetCallBack (
    void(*) (void) MEXTI_vpPointerTo_ISR_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 }
```

References [MADC_vpPointerToFunction](#), and [NULL](#).

7.59 ADC_interface.h

[Go to the documentation of this file.](#)

```
00001 /* FILENAME: ADC_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 12/14/2022
00005 */
00006 #ifndef _ADC_interface_H
00007 #define _ADC_interface_H
00008
00009
00010
00011
00012
00013
00014
00015
00016
```

```

00017
00018 typedef struct
00019 {
00020
00021     u8_t Port ;
00022
00023
00024
00025
00026
00027 } MADC_ConfigType ;
00028
00029
00030 *****
00031 /*          Functions prototypes          */
00032 *****
00033
00034
00035 void MADC_vInit( void ) ;
00036
00037 u16_t MADC_u16ConvertToDigital( VAR(u8_t) A_u8ChannelNum ) ;
00038
00039 void MADC_vDisable( void ) ;
00040
00041 void MADC_vEnable( void ) ;
00042
00043 void MADC_vRegINT_Disable( void ) ;
00044
00045 void MADC_vRegINTEnable( void ) ;
00046
00047 void MADC_vSelectMode( VAR(u8_t) A_u8ModeNum ) ;
00048
00049 void MADC_vSelectChannel( VAR(u8_t) A_u8ChannelNum ) ;
00050
00051 void MADC_vStartConversion( VAR(u8_t) A_u8ChannelNum ) ;
00052
00053 void MADC_vSelectChannelsOrder( VAR(u8_t) A_u8ChannelOrder ) ;
00054
00055 void MADC_vSelectSampleTime( VAR(u8_t) A_u8ChannelNum, VAR(u8_t) A_u8SampleTime ) ;
00056
00057 void MADC_vSelectResolution( VAR(u8_t) A_u8Resolution ) ;
00058
00059 u16_t MADC_u16GetADCData( void ) ;
00060
00061 void MADC_vInitStruct( P2VAR(MADC_ConfigType) A_ADCConfig ) ;
00062
00063 void MADC_vSetCallBack( void (*MEXTI_vpPointerTo_ISR_function) (void) ) ;
00064
00065
00066
00067 *****
00068 /*          Interfacing macros          */
00069 *****
00070
00071 #define CHANNEL0    0
00072 #define CHANNEL1    1
00073 #define CHANNEL2    2
00074 #define CHANNEL3    3
00075 #define CHANNEL4    4
00076 #define CHANNEL5    5
00077 #define CHANNEL6    6
00078 #define CHANNEL7    7
00079 #define CHANNEL8    8
00080 #define CHANNEL9    9
00081 #define CHANNEL10   10
00082 #define CHANNEL11   11
00083 #define CHANNEL12   12
00084 #define CHANNEL13   13
00085 #define CHANNEL14   14
00086 #define CHANNEL15   15
00087 #define CHANNEL16   16
00088 #define CHANNEL17   17
00089 #define CHANNEL18   18
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100 #endif // _ADC_interface_H

```

7.60 COTS/MCAL/ADC/ADC_private.h File Reference

Data Structures

- struct [ADC_MemoryMapType](#)
ADC declaration structure for its registers.

Macros

- #define [ADC_BASE_ADDRESS](#) 0x40012000
- #define [ADC](#) (volatile P2VAR(ADC_MemoryMapType)) ([ADC_BASE_ADDRESS](#)))
- #define [OVR](#) 5
- #define [STRT](#) 4
- #define [JSTRT](#) 3
- #define [JEOC](#) 2
- #define [EOC](#) 1
- #define [AWD](#) 0
- #define [OVRIE](#) 26
- #define [RES0](#) 24
- #define [RES1](#) 25
- #define [AWDEN](#) 23
- #define [JAWDEN](#) 22
- #define [DISCNUM0](#) 13
- #define [DISCNUM1](#) 14
- #define [DISCNUM2](#) 15
- #define [JDISCEN](#) 13
- #define [DISCEN](#) 11
- #define [JAUTO](#) 10
- #define [AWDSGL](#) 9
- #define [SCAN](#) 8
- #define [JEOCIE](#) 7
- #define [AWDIE](#) 6
- #define [EOCIE](#) 5
- #define [AWDCH0](#) 0
- #define [AWDCH1](#) 1
- #define [AWDCH2](#) 2
- #define [AWDCH3](#) 3
- #define [AWDCH4](#) 4
- #define [SWSTART](#) 30
- #define [EXTENO](#) 28
- #define [EXTEN1](#) 29
- #define [EXTSEL0](#) 24
- #define [EXTSEL1](#) 25
- #define [EXTSEL2](#) 26
- #define [EXTSEL3](#) 27
- #define [JSWSTART](#) 22
- #define [JEXTEN0](#) 20
- #define [JEXTEN1](#) 21
- #define [JEXTSEL0](#) 16
- #define [JEXTSEL1](#) 17
- #define [JEXTSEL2](#) 18
- #define [JEXTSEL3](#) 19
- #define [ALIGN](#) 11

- #define EOCS 10
- #define DDS 9
- #define DMA 8
- #define CONT 1
- #define ADON 0
- #define SMP10_b0 0
- #define SMP10_b1 1
- #define SMP10_b2 2
- #define SMP11_b0 3
- #define SMP11_b1 4
- #define SMP11_b2 5
- #define SMP12_b0 6
- #define SMP12_b1 7
- #define SMP12_b2 8
- #define SMP13_b0 9
- #define SMP13_b1 10
- #define SMP13_b2 11
- #define SMP14_b0 12
- #define SMP14_b1 13
- #define SMP14_b2 14
- #define SMP15_b0 15
- #define SMP15_b1 16
- #define SMP15_b2 17
- #define SMP15_b0 15
- #define SMP15_b1 16
- #define SMP15_b2 17
- #define SMP16_b0 18
- #define SMP16_b1 19
- #define SMP16_b2 20
- #define SMP17_b0 21
- #define SMP17_b1 22
- #define SMP17_b2 23
- #define SMP18_b0 24
- #define SMP18_b1 25
- #define SMP18_b2 26
- #define SMP0_b0 0
- #define SMP0_b1 1
- #define SMP0_b2 2
- #define SMP1_b0 3
- #define SMP1_b1 4
- #define SMP1_b2 5
- #define SMP2_b0 6
- #define SMP2_b1 7
- #define SMP2_b2 8
- #define SMP3_b0 9
- #define SMP3_b1 10
- #define SMP3_b2 11
- #define SMP4_b0 12
- #define SMP4_b1 13
- #define SMP4_b2 14
- #define SMP5_b0 15
- #define SMP5_b1 16
- #define SMP5_b2 17
- #define SMP5_b0 15
- #define SMP5_b1 16

- #define SMP5_b2 17
- #define SMP6_b0 18
- #define SMP6_b1 19
- #define SMP6_b2 20
- #define SMP7_b0 21
- #define SMP7_b1 22
- #define SMP7_b2 23
- #define SMP8_b0 24
- #define SMP8_b1 25
- #define SMP8_b2 26
- #define SMP9_b0 27
- #define SMP9_b1 28
- #define SMP9_b2 29
- #define L0 20
- #define L1 21
- #define L2 22
- #define L3 23
- #define SQ16_b0 15
- #define SQ16_b1 16
- #define SQ16_b2 17
- #define SQ16_b3 18
- #define SQ16_b4 19
- #define SQ15_b0 10
- #define SQ15_b1 11
- #define SQ15_b2 12
- #define SQ15_b3 13
- #define SQ15_b4 14
- #define SQ14_b0 5
- #define SQ14_b1 6
- #define SQ14_b2 7
- #define SQ14_b3 8
- #define SQ14_b4 9
- #define SQ13_b0 0
- #define SQ13_b1 1
- #define SQ13_b2 2
- #define SQ13_b3 3
- #define SQ13_b4 4
- #define SQ12_b0 25
- #define SQ12_b1 26
- #define SQ12_b2 27
- #define SQ12_b3 28
- #define SQ12_b4 29
- #define SQ11_b0 20
- #define SQ11_b1 21
- #define SQ11_b2 22
- #define SQ11_b3 23
- #define SQ11_b4 24
- #define SQ10_b0 15
- #define SQ10_b1 16
- #define SQ10_b2 17
- #define SQ10_b3 18
- #define SQ10_b4 19
- #define SQ9_b0 10
- #define SQ9_b1 11
- #define SQ9_b2 12

- #define SQ9_b3 13
- #define SQ9_b4 14
- #define SQ8_b0 5
- #define SQ8_b1 6
- #define SQ8_b2 7
- #define SQ8_b3 8
- #define SQ8_b4 9
- #define SQ7_b0 0
- #define SQ7_b1 1
- #define SQ7_b2 2
- #define SQ7_b3 3
- #define SQ7_b4 4
- #define SQ6_b0 25
- #define SQ6_b1 26
- #define SQ6_b2 27
- #define SQ6_b3 28
- #define SQ6_b4 29
- #define SQ5_b0 20
- #define SQ5_b1 21
- #define SQ5_b2 22
- #define SQ5_b3 23
- #define SQ5_b4 24
- #define SQ4_b0 15
- #define SQ4_b1 16
- #define SQ4_b2 17
- #define SQ4_b3 18
- #define SQ4_b4 19
- #define SQ3_b0 10
- #define SQ3_b1 11
- #define SQ3_b2 12
- #define SQ3_b3 13
- #define SQ3_b4 14
- #define SQ2_b0 5
- #define SQ2_b1 6
- #define SQ2_b2 7
- #define SQ2_b3 8
- #define SQ2_b4 9
- #define SQ1_b0 0
- #define SQ1_b1 1
- #define SQ1_b2 2
- #define SQ1_b3 3
- #define SQ1_b4 4
- #define TSVREFE 23
- #define VBATTE 22
- #define ADCPRE0 16
- #define ADCPRE1 17
- #define ENABLE 1
- #define DISABLE 0
- #define _12_BITS 0
- #define _10_BITS 1
- #define _8_BITS 2
- #define _6_BITS 3
- #define _1_CHANNEL 0
- #define _2_CHANNELS 1
- #define _3_CHANNELS 2

- #define _4_CHANNELS 3
- #define _5_CHANNELS 4
- #define _6_CHANNELS 5
- #define _7_CHANNELS 6
- #define _8_CHANNELS 7
- #define ZERO 0
- #define ONE 1
- #define THREE 2
- #define FOUR 3
- #define FIVE 4
- #define SIX 5
- #define SEVEN 6
- #define EIGHT 7
- #define NINE 8
- #define TEN 9
- #define ELEVEN 10
- #define TWELVE 11
- #define THIRTEEN 12
- #define FOURTEEN 13
- #define FIFTEEN 14
- #define SIXTEEN 15
- #define SEVENTEEN 16
- #define EIGHTEEN 17
- #define T1_CC1 0
- #define T1_CC2 1
- #define T1_CC3 2
- #define T2_CC2 3
- #define T2_CC3 4
- #define T2_CC4 5
- #define REG_T2_TRGO 6
- #define T3_CC1 7
- #define T3_TRGO 8
- #define T4_CC4 9
- #define T5_CC1 10
- #define T5_CC2 11
- #define T5_CC3 12
- #define EXTI_LINE11 15
- #define RISING 1
- #define FALLING 2
- #define ON_CHANGE 3
- #define T1_CC4 0
- #define T1_TRGO 1
- #define T2_CC1 2
- #define INJ_T2_TRGO 3
- #define T3_CC2 4
- #define T3_CC4 5
- #define T4_CC1 6
- #define T4_CC2 7
- #define T4_CC3 8
- #define T4_TRGO 9
- #define T5_CC4 10
- #define T5_TRGO 11
- #define EXTI_LINE15 15
- #define RIGHT 0
- #define LEFT 1

- #define _3_CYCLES 0
- #define _15_CYCLES 1
- #define _28_CYCLES 2
- #define _56_CYCLES 3
- #define _84_CYCLES 4
- #define _112_CYCLES 5
- #define _144_CYCLES 6
- #define _480_CYCLES 7
- #define _1_CONV 0
- #define _2_CONV 1
- #define _3_CONV 2
- #define _4_CONV 3
- #define _5_CONV 4
- #define _6_CONV 5
- #define _7_CONV 6
- #define _8_CONV 7
- #define _9_CONV 8
- #define _10_CONV 9
- #define _11_CONV 10
- #define _15_CONV 11
- #define _16_CONV 15
- #define DIV_BY_2 0
- #define DIV_BY_4 1
- #define DIV_BY_6 2
- #define DIV_BY_8 3

7.60.1 Macro Definition Documentation

7.60.1.1 ADC_BASE_ADDRESS

```
#define ADC_BASE_ADDRESS 0x40012000
```

ADC Base Address

Definition at line 137 of file [ADC_private.h](#).

7.60.1.2 ADC

```
#define ADC ( (volatile P2VAR(ADC_MemoryMapType) ) (ADC_BASE_ADDRESS) )
```

ADC register

Definition at line 144 of file [ADC_private.h](#).

7.60.1.3 OVR

```
#define OVR 5
```

Definition at line 151 of file [ADC_private.h](#).

7.60.1.4 STRT

```
#define STRT 4
```

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

7.60.1.5 JSTRT

```
#define JSTRT 3
```

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

7.60.1.6 JEOP

```
#define JEOP 2
```

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

7.60.1.7 EOC

```
#define EOC 1
```

Definition at line 155 of file [ADC_private.h](#).

7.60.1.8 AWD

```
#define AWD 0
```

Definition at line 156 of file [ADC_private.h](#).

7.60.1.9 OVRIE

```
#define OVRIE 26
```

Definition at line 162 of file [ADC_private.h](#).

7.60.1.10 RES0

```
#define RES0 24
```

Definition at line 163 of file [ADC_private.h](#).

7.60.1.11 RES1

```
#define RES1 25
```

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

7.60.1.12 AWDEN

```
#define AWDEN 23
```

Definition at line 165 of file [ADC_private.h](#).

7.60.1.13 JAWDEN

```
#define JAWDEN 22
```

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

7.60.1.14 DISCNUM0

```
#define DISCNUM0 13
```

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

7.60.1.15 DISCNUM1

```
#define DISCNUM1 14
```

Definition at line 168 of file [ADC_private.h](#).

7.60.1.16 DISCNUM2

```
#define DISCNUM2 15
```

Definition at line 169 of file [ADC_private.h](#).

7.60.1.17 JDISCEN

```
#define JDISCEN 13
```

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

7.60.1.18 DISCEN

```
#define DISCEN 11
```

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

7.60.1.19 JAUTO

```
#define JAUTO 10
```

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

7.60.1.20 AWDSGL

```
#define AWDSGL 9
```

Definition at line 173 of file [ADC_private.h](#).

7.60.1.21 SCAN

```
#define SCAN 8
```

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

7.60.1.22 JEOCIE

```
#define JEOCIE 7
```

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

7.60.1.23 AWDIE

```
#define AWDIE 6
```

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

7.60.1.24 EOCIE

```
#define EOCIE 5
```

Definition at line 177 of file [ADC_private.h](#).

7.60.1.25 AWDCH0

```
#define AWDCH0 0
```

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

7.60.1.26 AWDCH1

```
#define AWDCH1 1
```

Definition at line 179 of file [ADC_private.h](#).

7.60.1.27 AWDCH2

```
#define AWDCH2 2
```

Definition at line 180 of file [ADC_private.h](#).

7.60.1.28 AWDCH3

```
#define AWDCH3 3
```

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

7.60.1.29 AWDCH4

```
#define AWDCH4 4
```

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

7.60.1.30 SWSTART

```
#define SWSTART 30
```

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

7.60.1.31 EXTEN0

```
#define EXTEN0 28
```

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

7.60.1.32 EXTEN1

```
#define EXTEN1 29
```

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

7.60.1.33 EXTSEL0

```
#define EXTSEL0 24
```

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

7.60.1.34 EXTSEL1

```
#define EXTSEL1 25
```

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

7.60.1.35 EXTSEL2

```
#define EXTSEL2 26
```

Definition at line 193 of file [ADC_private.h](#).

7.60.1.36 EXTSEL3

```
#define EXTSEL3 27
```

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

7.60.1.37 JSWSTART

```
#define JSWSTART 22
```

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

7.60.1.38 JEXTEN0

```
#define JEXTEN0 20
```

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

7.60.1.39 JEXTEN1

```
#define JEXTEN1 21
```

Definition at line 197 of file [ADC_private.h](#).

7.60.1.40 JEXTSEL0

```
#define JEXTSEL0 16
```

Definition at line 198 of file [ADC_private.h](#).

7.60.1.41 JEXTSEL1

```
#define JEXTSEL1 17
```

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

7.60.1.42 JEXTSEL2

```
#define JEXTSEL2 18
```

Definition at line 200 of file [ADC_private.h](#).

7.60.1.43 JEXTSEL3

```
#define JEXTSEL3 19
```

Definition at line 201 of file [ADC_private.h](#).

7.60.1.44 ALIGN

```
#define ALIGN 11
```

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

7.60.1.45 EOCS

```
#define EOCS 10
```

Definition at line 203 of file [ADC_private.h](#).

7.60.1.46 DDS

```
#define DDS 9
```

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

7.60.1.47 DMA

```
#define DMA 8
```

Definition at line 205 of file [ADC_private.h](#).

7.60.1.48 CONT

```
#define CONT 1
```

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

7.60.1.49 ADON

```
#define ADON 0
```

Definition at line 207 of file [ADC_private.h](#).

7.60.1.50 SMP10_b0

```
#define SMP10_b0 0
```

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

7.60.1.51 SMP10_b1

```
#define SMP10_b1 1
```

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

7.60.1.52 SMP10_b2

```
#define SMP10_b2 2
```

Definition at line 215 of file [ADC_private.h](#).

7.60.1.53 SMP11_b0

```
#define SMP11_b0 3
```

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

7.60.1.54 SMP11_b1

```
#define SMP11_b1 4
```

Definition at line 218 of file [ADC_private.h](#).

7.60.1.55 SMP11_b2

```
#define SMP11_b2 5
```

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

7.60.1.56 SMP12_b0

```
#define SMP12_b0 6
```

Definition at line 221 of file [ADC_private.h](#).

7.60.1.57 SMP12_b1

```
#define SMP12_b1 7
```

Definition at line [222](#) of file [ADC_private.h](#).

7.60.1.58 SMP12_b2

```
#define SMP12_b2 8
```

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

7.60.1.59 SMP13_b0

```
#define SMP13_b0 9
```

Definition at line [225](#) of file [ADC_private.h](#).

7.60.1.60 SMP13_b1

```
#define SMP13_b1 10
```

Definition at line [226](#) of file [ADC_private.h](#).

7.60.1.61 SMP13_b2

```
#define SMP13_b2 11
```

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

7.60.1.62 SMP14_b0

```
#define SMP14_b0 12
```

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

7.60.1.63 SMP14_b1

```
#define SMP14_b1 13
```

Definition at line 230 of file [ADC_private.h](#).

7.60.1.64 SMP14_b2

```
#define SMP14_b2 14
```

Definition at line 231 of file [ADC_private.h](#).

7.60.1.65 SMP15_b0 [1/2]

```
#define SMP15_b0 15
```

Definition at line 237 of file [ADC_private.h](#).

7.60.1.66 SMP15_b1 [1/2]

```
#define SMP15_b1 16
```

Definition at line 238 of file [ADC_private.h](#).

7.60.1.67 SMP15_b2 [1/2]

```
#define SMP15_b2 17
```

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

7.60.1.68 SMP15_b0 [2/2]

```
#define SMP15_b0 15
```

Definition at line 237 of file [ADC_private.h](#).

7.60.1.69 SMP15_b1 [2/2]

```
#define SMP15_b1 16
```

Definition at line [238](#) of file [ADC_private.h](#).

7.60.1.70 SMP15_b2 [2/2]

```
#define SMP15_b2 17
```

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

7.60.1.71 SMP16_b0

```
#define SMP16_b0 18
```

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

7.60.1.72 SMP16_b1

```
#define SMP16_b1 19
```

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

7.60.1.73 SMP16_b2

```
#define SMP16_b2 20
```

Definition at line [243](#) of file [ADC_private.h](#).

7.60.1.74 SMP17_b0

```
#define SMP17_b0 21
```

Definition at line [245](#) of file [ADC_private.h](#).

7.60.1.75 SMP17_b1

```
#define SMP17_b1 22
```

Definition at line [246](#) of file [ADC_private.h](#).

7.60.1.76 SMP17_b2

```
#define SMP17_b2 23
```

Definition at line [247](#) of file [ADC_private.h](#).

7.60.1.77 SMP18_b0

```
#define SMP18_b0 24
```

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

7.60.1.78 SMP18_b1

```
#define SMP18_b1 25
```

Definition at line [250](#) of file [ADC_private.h](#).

7.60.1.79 SMP18_b2

```
#define SMP18_b2 26
```

Definition at line [251](#) of file [ADC_private.h](#).

7.60.1.80 SMP0_b0

```
#define SMP0_b0 0
```

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

7.60.1.81 SMP0_b1

```
#define SMP0_b1 1
```

Definition at line [258](#) of file [ADC_private.h](#).

7.60.1.82 SMP0_b2

```
#define SMP0_b2 2
```

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

7.60.1.83 SMP1_b0

```
#define SMP1_b0 3
```

Definition at line [261](#) of file [ADC_private.h](#).

7.60.1.84 SMP1_b1

```
#define SMP1_b1 4
```

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

7.60.1.85 SMP1_b2

```
#define SMP1_b2 5
```

Definition at line [263](#) of file [ADC_private.h](#).

7.60.1.86 SMP2_b0

```
#define SMP2_b0 6
```

Definition at line [265](#) of file [ADC_private.h](#).

7.60.1.87 SMP2_b1

```
#define SMP2_b1 7
```

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

7.60.1.88 SMP2_b2

```
#define SMP2_b2 8
```

Definition at line [267](#) of file [ADC_private.h](#).

7.60.1.89 SMP3_b0

```
#define SMP3_b0 9
```

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

7.60.1.90 SMP3_b1

```
#define SMP3_b1 10
```

Definition at line [270](#) of file [ADC_private.h](#).

7.60.1.91 SMP3_b2

```
#define SMP3_b2 11
```

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

7.60.1.92 SMP4_b0

```
#define SMP4_b0 12
```

Definition at line [273](#) of file [ADC_private.h](#).

7.60.1.93 SMP4_b1

```
#define SMP4_b1 13
```

Definition at line [274](#) of file [ADC_private.h](#).

7.60.1.94 SMP4_b2

```
#define SMP4_b2 14
```

Definition at line [275](#) of file [ADC_private.h](#).

7.60.1.95 SMP5_b0 [1/2]

```
#define SMP5_b0 15
```

Definition at line [281](#) of file [ADC_private.h](#).

7.60.1.96 SMP5_b1 [1/2]

```
#define SMP5_b1 16
```

Definition at line [282](#) of file [ADC_private.h](#).

7.60.1.97 SMP5_b2 [1/2]

```
#define SMP5_b2 17
```

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

7.60.1.98 SMP5_b0 [2/2]

```
#define SMP5_b0 15
```

Definition at line [281](#) of file [ADC_private.h](#).

7.60.1.99 SMP5_b1 [2/2]

```
#define SMP5_b1 16
```

Definition at line [282](#) of file [ADC_private.h](#).

7.60.1.100 SMP5_b2 [2/2]

```
#define SMP5_b2 17
```

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

7.60.1.101 SMP6_b0

```
#define SMP6_b0 18
```

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

7.60.1.102 SMP6_b1

```
#define SMP6_b1 19
```

Definition at line [286](#) of file [ADC_private.h](#).

7.60.1.103 SMP6_b2

```
#define SMP6_b2 20
```

Definition at line [287](#) of file [ADC_private.h](#).

7.60.1.104 SMP7_b0

```
#define SMP7_b0 21
```

Definition at line [289](#) of file [ADC_private.h](#).

7.60.1.105 SMP7_b1

```
#define SMP7_b1 22
```

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

7.60.1.106 SMP7_b2

```
#define SMP7_b2 23
```

Definition at line [291](#) of file [ADC_private.h](#).

7.60.1.107 SMP8_b0

```
#define SMP8_b0 24
```

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

7.60.1.108 SMP8_b1

```
#define SMP8_b1 25
```

Definition at line [294](#) of file [ADC_private.h](#).

7.60.1.109 SMP8_b2

```
#define SMP8_b2 26
```

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

7.60.1.110 SMP9_b0

```
#define SMP9_b0 27
```

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

7.60.1.111 SMP9_b1

```
#define SMP9_b1 28
```

Definition at line 298 of file [ADC_private.h](#).

7.60.1.112 SMP9_b2

```
#define SMP9_b2 29
```

Definition at line 299 of file [ADC_private.h](#).

7.60.1.113 L0

```
#define L0 20
```

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

7.60.1.114 L1

```
#define L1 21
```

Definition at line 306 of file [ADC_private.h](#).

7.60.1.115 L2

```
#define L2 22
```

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

7.60.1.116 L3

```
#define L3 23
```

Definition at line 308 of file [ADC_private.h](#).

7.60.1.117 SQ16_b0

```
#define SQ16_b0 15
```

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

7.60.1.118 SQ16_b1

```
#define SQ16_b1 16
```

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

7.60.1.119 SQ16_b2

```
#define SQ16_b2 17
```

Definition at line [312](#) of file [ADC_private.h](#).

7.60.1.120 SQ16_b3

```
#define SQ16_b3 18
```

Definition at line [313](#) of file [ADC_private.h](#).

7.60.1.121 SQ16_b4

```
#define SQ16_b4 19
```

Definition at line [314](#) of file [ADC_private.h](#).

7.60.1.122 SQ15_b0

```
#define SQ15_b0 10
```

Definition at line [316](#) of file [ADC_private.h](#).

7.60.1.123 SQ15_b1

```
#define SQ15_b1 11
```

Definition at line 317 of file [ADC_private.h](#).

7.60.1.124 SQ15_b2

```
#define SQ15_b2 12
```

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

7.60.1.125 SQ15_b3

```
#define SQ15_b3 13
```

Definition at line 319 of file [ADC_private.h](#).

7.60.1.126 SQ15_b4

```
#define SQ15_b4 14
```

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

7.60.1.127 SQ14_b0

```
#define SQ14_b0 5
```

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

7.60.1.128 SQ14_b1

```
#define SQ14_b1 6
```

Definition at line 323 of file [ADC_private.h](#).

7.60.1.129 SQ14_b2

```
#define SQ14_b2 7
```

Definition at line [324](#) of file [ADC_private.h](#).

7.60.1.130 SQ14_b3

```
#define SQ14_b3 8
```

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

7.60.1.131 SQ14_b4

```
#define SQ14_b4 9
```

Definition at line [326](#) of file [ADC_private.h](#).

7.60.1.132 SQ13_b0

```
#define SQ13_b0 0
```

Definition at line [328](#) of file [ADC_private.h](#).

7.60.1.133 SQ13_b1

```
#define SQ13_b1 1
```

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

7.60.1.134 SQ13_b2

```
#define SQ13_b2 2
```

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

7.60.1.135 SQ13_b3

```
#define SQ13_b3 3
```

Definition at line 331 of file [ADC_private.h](#).

7.60.1.136 SQ13_b4

```
#define SQ13_b4 4
```

Definition at line 332 of file [ADC_private.h](#).

7.60.1.137 SQ12_b0

```
#define SQ12_b0 25
```

Definition at line 338 of file [ADC_private.h](#).

7.60.1.138 SQ12_b1

```
#define SQ12_b1 26
```

Definition at line 339 of file [ADC_private.h](#).

7.60.1.139 SQ12_b2

```
#define SQ12_b2 27
```

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

7.60.1.140 SQ12_b3

```
#define SQ12_b3 28
```

Definition at line 341 of file [ADC_private.h](#).

7.60.1.141 SQ12_b4

```
#define SQ12_b4 29
```

Definition at line 342 of file [ADC_private.h](#).

7.60.1.142 SQ11_b0

```
#define SQ11_b0 20
```

Definition at line 344 of file [ADC_private.h](#).

7.60.1.143 SQ11_b1

```
#define SQ11_b1 21
```

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

7.60.1.144 SQ11_b2

```
#define SQ11_b2 22
```

Definition at line 346 of file [ADC_private.h](#).

7.60.1.145 SQ11_b3

```
#define SQ11_b3 23
```

Definition at line 347 of file [ADC_private.h](#).

7.60.1.146 SQ11_b4

```
#define SQ11_b4 24
```

Definition at line 348 of file [ADC_private.h](#).

7.60.1.147 SQ10_b0

```
#define SQ10_b0 15
```

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

7.60.1.148 SQ10_b1

```
#define SQ10_b1 16
```

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

7.60.1.149 SQ10_b2

```
#define SQ10_b2 17
```

Definition at line 352 of file [ADC_private.h](#).

7.60.1.150 SQ10_b3

```
#define SQ10_b3 18
```

Definition at line 353 of file [ADC_private.h](#).

7.60.1.151 SQ10_b4

```
#define SQ10_b4 19
```

Definition at line 354 of file [ADC_private.h](#).

7.60.1.152 SQ9_b0

```
#define SQ9_b0 10
```

Definition at line 356 of file [ADC_private.h](#).

7.60.1.153 SQ9_b1

```
#define SQ9_b1 11
```

Definition at line 357 of file [ADC_private.h](#).

7.60.1.154 SQ9_b2

```
#define SQ9_b2 12
```

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

7.60.1.155 SQ9_b3

```
#define SQ9_b3 13
```

Definition at line 359 of file [ADC_private.h](#).

7.60.1.156 SQ9_b4

```
#define SQ9_b4 14
```

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

7.60.1.157 SQ8_b0

```
#define SQ8_b0 5
```

Definition at line 362 of file [ADC_private.h](#).

7.60.1.158 SQ8_b1

```
#define SQ8_b1 6
```

Definition at line 363 of file [ADC_private.h](#).

7.60.1.159 SQ8_b2

```
#define SQ8_b2 7
```

Definition at line 364 of file [ADC_private.h](#).

7.60.1.160 SQ8_b3

```
#define SQ8_b3 8
```

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

7.60.1.161 SQ8_b4

```
#define SQ8_b4 9
```

Definition at line 366 of file [ADC_private.h](#).

7.60.1.162 SQ7_b0

```
#define SQ7_b0 0
```

Definition at line 368 of file [ADC_private.h](#).

7.60.1.163 SQ7_b1

```
#define SQ7_b1 1
```

Definition at line 369 of file [ADC_private.h](#).

7.60.1.164 SQ7_b2

```
#define SQ7_b2 2
```

Definition at line 370 of file [ADC_private.h](#).

7.60.1.165 SQ7_b3

```
#define SQ7_b3 3
```

Definition at line 371 of file [ADC_private.h](#).

7.60.1.166 SQ7_b4

```
#define SQ7_b4 4
```

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

7.60.1.167 SQ6_b0

```
#define SQ6_b0 25
```

Definition at line 378 of file [ADC_private.h](#).

7.60.1.168 SQ6_b1

```
#define SQ6_b1 26
```

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

7.60.1.169 SQ6_b2

```
#define SQ6_b2 27
```

Definition at line 380 of file [ADC_private.h](#).

7.60.1.170 SQ6_b3

```
#define SQ6_b3 28
```

Definition at line 381 of file [ADC_private.h](#).

7.60.1.171 SQ6_b4

```
#define SQ6_b4 29
```

Definition at line 382 of file [ADC_private.h](#).

7.60.1.172 SQ5_b0

```
#define SQ5_b0 20
```

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

7.60.1.173 SQ5_b1

```
#define SQ5_b1 21
```

Definition at line 385 of file [ADC_private.h](#).

7.60.1.174 SQ5_b2

```
#define SQ5_b2 22
```

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

7.60.1.175 SQ5_b3

```
#define SQ5_b3 23
```

Definition at line 387 of file [ADC_private.h](#).

7.60.1.176 SQ5_b4

```
#define SQ5_b4 24
```

Definition at line 388 of file [ADC_private.h](#).

7.60.1.177 SQ4_b0

```
#define SQ4_b0 15
```

Definition at line 390 of file [ADC_private.h](#).

7.60.1.178 SQ4_b1

```
#define SQ4_b1 16
```

Definition at line 391 of file [ADC_private.h](#).

7.60.1.179 SQ4_b2

```
#define SQ4_b2 17
```

Definition at line 392 of file [ADC_private.h](#).

7.60.1.180 SQ4_b3

```
#define SQ4_b3 18
```

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

7.60.1.181 SQ4_b4

```
#define SQ4_b4 19
```

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

7.60.1.182 SQ3_b0

```
#define SQ3_b0 10
```

Definition at line 396 of file [ADC_private.h](#).

7.60.1.183 SQ3_b1

```
#define SQ3_b1 11
```

Definition at line 397 of file [ADC_private.h](#).

7.60.1.184 SQ3_b2

```
#define SQ3_b2 12
```

Definition at line 398 of file [ADC_private.h](#).

7.60.1.185 SQ3_b3

```
#define SQ3_b3 13
```

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

7.60.1.186 SQ3_b4

```
#define SQ3_b4 14
```

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

7.60.1.187 SQ2_b0

```
#define SQ2_b0 5
```

Definition at line 402 of file [ADC_private.h](#).

7.60.1.188 SQ2_b1

```
#define SQ2_b1 6
```

Definition at line 403 of file [ADC_private.h](#).

7.60.1.189 SQ2_b2

```
#define SQ2_b2 7
```

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

7.60.1.190 SQ2_b3

```
#define SQ2_b3 8
```

Definition at line 405 of file [ADC_private.h](#).

7.60.1.191 SQ2_b4

```
#define SQ2_b4 9
```

Definition at line 406 of file [ADC_private.h](#).

7.60.1.192 SQ1_b0

```
#define SQ1_b0 0
```

Definition at line 408 of file [ADC_private.h](#).

7.60.1.193 SQ1_b1

```
#define SQ1_b1 1
```

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

7.60.1.194 SQ1_b2

```
#define SQ1_b2 2
```

Definition at line 410 of file [ADC_private.h](#).

7.60.1.195 SQ1_b3

```
#define SQ1_b3 3
```

Definition at line 411 of file [ADC_private.h](#).

7.60.1.196 SQ1_b4

```
#define SQ1_b4 4
```

Definition at line 412 of file [ADC_private.h](#).

7.60.1.197 TSVREFE

```
#define TSVREFE 23
```

Definition at line 418 of file [ADC_private.h](#).

7.60.1.198 VBATE

```
#define VBATE 22
```

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

7.60.1.199 ADCPRE0

```
#define ADCPRE0 16
```

Definition at line 420 of file [ADC_private.h](#).

7.60.1.200 ADCPRE1

```
#define ADCPRE1 17
```

Definition at line 421 of file [ADC_private.h](#).

7.60.1.201 ENABLE

```
#define ENABLE 1
```

Definition at line [427](#) of file [ADC_private.h](#).

7.60.1.202 DISABLE

```
#define DISABLE 0
```

Definition at line [428](#) of file [ADC_private.h](#).

7.60.1.203 _12_BITS

```
#define _12_BITS 0
```

Definition at line [430](#) of file [ADC_private.h](#).

7.60.1.204 _10_BITS

```
#define _10_BITS 1
```

Definition at line [431](#) of file [ADC_private.h](#).

7.60.1.205 _8_BITS

```
#define _8_BITS 2
```

Definition at line [432](#) of file [ADC_private.h](#).

7.60.1.206 _6_BITS

```
#define _6_BITS 3
```

Definition at line [433](#) of file [ADC_private.h](#).

7.60.1.207 _1_CHANNEL

```
#define _1_CHANNEL 0
```

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

7.60.1.208 _2_CHANNELS

```
#define _2_CHANNELS 1
```

Definition at line [436](#) of file [ADC_private.h](#).

7.60.1.209 _3_CHANNELS

```
#define _3_CHANNELS 2
```

Definition at line [437](#) of file [ADC_private.h](#).

7.60.1.210 _4_CHANNELS

```
#define _4_CHANNELS 3
```

Definition at line [438](#) of file [ADC_private.h](#).

7.60.1.211 _5_CHANNELS

```
#define _5_CHANNELS 4
```

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

7.60.1.212 _6_CHANNELS

```
#define _6_CHANNELS 5
```

Definition at line [440](#) of file [ADC_private.h](#).

7.60.1.213 _7_CHANNELS

```
#define _7_CHANNELS 6
```

Definition at line [441](#) of file [ADC_private.h](#).

7.60.1.214 _8_CHANNELS

```
#define _8_CHANNELS 7
```

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

7.60.1.215 ZERO

```
#define ZERO 0
```

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

7.60.1.216 ONE

```
#define ONE 1
```

Definition at line [445](#) of file [ADC_private.h](#).

7.60.1.217 THREE

```
#define THREE 2
```

Definition at line [446](#) of file [ADC_private.h](#).

7.60.1.218 FOUR

```
#define FOUR 3
```

Definition at line [447](#) of file [ADC_private.h](#).

7.60.1.219 FIVE

```
#define FIVE 4
```

Definition at line 448 of file [ADC_private.h](#).

7.60.1.220 SIX

```
#define SIX 5
```

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

7.60.1.221 SEVEN

```
#define SEVEN 6
```

Definition at line 450 of file [ADC_private.h](#).

7.60.1.222 EIGHT

```
#define EIGHT 7
```

Definition at line 451 of file [ADC_private.h](#).

7.60.1.223 NINE

```
#define NINE 8
```

Definition at line 452 of file [ADC_private.h](#).

7.60.1.224 TEN

```
#define TEN 9
```

Definition at line 453 of file [ADC_private.h](#).

7.60.1.225 ELEVEN

```
#define ELEVEN 10
```

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

7.60.1.226 TWELVE

```
#define TWELVE 11
```

Definition at line [455](#) of file [ADC_private.h](#).

7.60.1.227 THERTEEN

```
#define THERTEEN 12
```

Definition at line [456](#) of file [ADC_private.h](#).

7.60.1.228 FOURTEEN

```
#define FOURTEEN 13
```

Definition at line [457](#) of file [ADC_private.h](#).

7.60.1.229 FIFTEEN

```
#define FIFTEEN 14
```

Definition at line [458](#) of file [ADC_private.h](#).

7.60.1.230 SIXTEEN

```
#define SIXTEEN 15
```

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

7.60.1.231 SEVENTEEN

```
#define SEVENTEEN 16
```

Definition at line 460 of file [ADC_private.h](#).

7.60.1.232 EIGHTEEN

```
#define EIGHTEEN 17
```

Definition at line 461 of file [ADC_private.h](#).

7.60.1.233 T1_CC1

```
#define T1_CC1 0
```

Definition at line 463 of file [ADC_private.h](#).

7.60.1.234 T1_CC2

```
#define T1_CC2 1
```

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

7.60.1.235 T1_CC3

```
#define T1_CC3 2
```

Definition at line 465 of file [ADC_private.h](#).

7.60.1.236 T2_CC2

```
#define T2_CC2 3
```

Definition at line 466 of file [ADC_private.h](#).

7.60.1.237 T2_CC3

```
#define T2_CC3 4
```

Definition at line [467](#) of file [ADC_private.h](#).

7.60.1.238 T2_CC4

```
#define T2_CC4 5
```

Definition at line [468](#) of file [ADC_private.h](#).

7.60.1.239 REG_T2_TRGO

```
#define REG_T2_TRGO 6
```

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

7.60.1.240 T3_CC1

```
#define T3_CC1 7
```

Definition at line [470](#) of file [ADC_private.h](#).

7.60.1.241 T3_TRGO

```
#define T3_TRGO 8
```

Definition at line [471](#) of file [ADC_private.h](#).

7.60.1.242 T4_CC4

```
#define T4_CC4 9
```

Definition at line [472](#) of file [ADC_private.h](#).

7.60.1.243 T5_CC1

```
#define T5_CC1 10
```

Definition at line [473](#) of file [ADC_private.h](#).

7.60.1.244 T5_CC2

```
#define T5_CC2 11
```

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

7.60.1.245 T5_CC3

```
#define T5_CC3 12
```

Definition at line [475](#) of file [ADC_private.h](#).

7.60.1.246 EXTI_LINE11

```
#define EXTI_LINE11 15
```

Definition at line [476](#) of file [ADC_private.h](#).

7.60.1.247 RISING

```
#define RISING 1
```

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

7.60.1.248 FALLING

```
#define FALLING 2
```

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

7.60.1.249 ON_CHANGE

```
#define ON_CHANGE 3
```

Definition at line [480](#) of file [ADC_private.h](#).

7.60.1.250 T1_CC4

```
#define T1_CC4 0
```

Definition at line [482](#) of file [ADC_private.h](#).

7.60.1.251 T1_TRGO

```
#define T1_TRGO 1
```

Definition at line [483](#) of file [ADC_private.h](#).

7.60.1.252 T2_CC1

```
#define T2_CC1 2
```

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

7.60.1.253 INJ_T2_TRGO

```
#define INJ_T2_TRGO 3
```

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

7.60.1.254 T3_CC2

```
#define T3_CC2 4
```

Definition at line [486](#) of file [ADC_private.h](#).

7.60.1.255 T3_CC4

```
#define T3_CC4 5
```

Definition at line [487](#) of file [ADC_private.h](#).

7.60.1.256 T4_CC1

```
#define T4_CC1 6
```

Definition at line [488](#) of file [ADC_private.h](#).

7.60.1.257 T4_CC2

```
#define T4_CC2 7
```

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

7.60.1.258 T4_CC3

```
#define T4_CC3 8
```

Definition at line [490](#) of file [ADC_private.h](#).

7.60.1.259 T4_TRGO

```
#define T4_TRGO 9
```

Definition at line [491](#) of file [ADC_private.h](#).

7.60.1.260 T5_CC4

```
#define T5_CC4 10
```

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

7.60.1.261 T5_TRGO

```
#define T5_TRGO 11
```

Definition at line [493](#) of file [ADC_private.h](#).

7.60.1.262 EXTI_LINE15

```
#define EXTI_LINE15 15
```

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

7.60.1.263 RIGHT

```
#define RIGHT 0
```

Definition at line [496](#) of file [ADC_private.h](#).

7.60.1.264 LEFT

```
#define LEFT 1
```

Definition at line [497](#) of file [ADC_private.h](#).

7.60.1.265 _3_CYCLES

```
#define _3_CYCLES 0
```

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

7.60.1.266 _15_CYCLES

```
#define _15_CYCLES 1
```

Definition at line [500](#) of file [ADC_private.h](#).

7.60.1.267 _28_CYCLEs

```
#define _28_CYCLEs 2
```

Definition at line 501 of file [ADC_private.h](#).

7.60.1.268 _56_CYCLEs

```
#define _56_CYCLEs 3
```

Definition at line 502 of file [ADC_private.h](#).

7.60.1.269 _84_CYCLEs

```
#define _84_CYCLEs 4
```

Definition at line 503 of file [ADC_private.h](#).

7.60.1.270 _112_CYCLEs

```
#define _112_CYCLEs 5
```

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

7.60.1.271 _144_CYCLEs

```
#define _144_CYCLEs 6
```

Definition at line 505 of file [ADC_private.h](#).

7.60.1.272 _480_CYCLEs

```
#define _480_CYCLEs 7
```

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

7.60.1.273 _1_CONV

```
#define _1_CONV 0
```

Definition at line [508](#) of file [ADC_private.h](#).

7.60.1.274 _2_CONV

```
#define _2_CONV 1
```

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

7.60.1.275 _3_CONV

```
#define _3_CONV 2
```

Definition at line [510](#) of file [ADC_private.h](#).

7.60.1.276 _4_CONV

```
#define _4_CONV 3
```

Definition at line [511](#) of file [ADC_private.h](#).

7.60.1.277 _5_CONV

```
#define _5_CONV 4
```

Definition at line [512](#) of file [ADC_private.h](#).

7.60.1.278 _6_CONV

```
#define _6_CONV 5
```

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

7.60.1.279 _7_CONV

```
#define _7_CONV 6
```

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

7.60.1.280 _8_CONV

```
#define _8_CONV 7
```

Definition at line 515 of file [ADC_private.h](#).

7.60.1.281 _9_CONV

```
#define _9_CONV 8
```

Definition at line 516 of file [ADC_private.h](#).

7.60.1.282 _10_CONV

```
#define _10_CONV 9
```

Definition at line 517 of file [ADC_private.h](#).

7.60.1.283 _11_CONV

```
#define _11_CONV 10
```

Definition at line 518 of file [ADC_private.h](#).

7.60.1.284 _15_CONV

```
#define _15_CONV 11
```

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

7.60.1.285 _16_CONV

```
#define _16_CONV 15
```

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

7.60.1.286 DIV_BY_2

```
#define DIV_BY_2 0
```

Definition at line [523](#) of file [ADC_private.h](#).

7.60.1.287 DIV_BY_4

```
#define DIV_BY_4 1
```

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

7.60.1.288 DIV_BY_6

```
#define DIV_BY_6 2
```

Definition at line [525](#) of file [ADC_private.h](#).

7.60.1.289 DIV_BY_8

```
#define DIV_BY_8 3
```

Definition at line [526](#) of file [ADC_private.h](#).

7.61 ADC_private.h

[Go to the documentation of this file.](#)

```

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 /* Peripherals declaration */
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
00137 #define ADC_BASE_ADDRESS 0x40012000
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 /***** *****/
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 /***** *****/

```

```
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 AWDSGL 9  
00174 #define SCAN 8  
00175 #define JEOCIE 7  
00176 #define AWDIE 6  
00177 #define EOCIE 5  
00178 #define AWDCHO 0  
00179 #define AWDCH1 1  
00180 #define AWDCH2 2  
00181 #define AWDCH3 3  
00182 #define AWDCH4 4  
00183  
00184 /*****  
00185 /* ADC_CR2 register bits */  
00186 *****/  
00187  
00188 #define SWSTART 30  
00189 #define EXTENO 28  
00190 #define EXTE1 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 /*****  
00210 /* ADC_SMPR1 register bits */  
00211 *****/  
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 /*****  
00254 /* ADC_SMPR2 register bits */  
00255 *****/  
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 /*****  
00302 /* ADC_SQR1 register bits */  
00303 *****/  
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 *****/
```

```

00335 /* ADC_SQR2 register bits */
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 /***** */
00375 /* ADC_SQR3 register bits */
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 /***** */
00415 /* ADC_CCR register bits */
00416 /***** */
00417
00418 #define TSVREFE 23
00419 #define VBATE 22
00420 #define ADCPRE0 16
00421 #define ADCPRE1 17

```

```
00422
00423 /***** Config.h Macros *****/
00424 /* */
00425 /***** */
00426
00427 #define ENABLE 1
00428 #define DISABLE 0
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

```

7.62 COTS/MCAL/ADC/ADC_program.c File Reference

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../RCC/MRCC_interface.h"
#include "ADC_interface.h"
#include "ADC_private.h"
#include "ADC_config.h"

```

Functions

- void **MADC_vInit** (void)
- void **MADC_vSelectChannel** (**u8_t** A_u8ChannelNum)
- **u16_t MADC_u16ConvertToDigital** (**u8_t** A_u8ChannelNum)
- void **MADC_vDisable** (void)
- void **MADC_vEnable** (void)
- void **MADC_vRegINT_Disable** (void)
- void **MADC_vRegINTEnable** (void)
- void **MADC_vSetCallBack** (void(*MEXTI_vpPointerTo_ISR_function))(void))
- void **ADC_IRQHandler** (void)

Variables

- void(* **MADC_vpPointerToFunction**)(void) = ((void*)0)

7.62.1 Function Documentation

7.62.1.1 MADC_vInit()

```
void MADC_vInit (
    void )
```

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_vEnablePeripheralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00035
00036     // EN/DIS power for ADC to be turned on/off:
00037 #if ADC_ON == ENABLE
00038     SET_BIT( ADC->CR2, ADON ) ;
00040
00041 #elif ADC_ON == DISABLE
00042     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
00107     // EN/DIS Injected INT:
00108 #if JEOCIE_MODE == ENABLE
```

```
00109
00110     SET_BIT( ADC->CR1, JEOCIE ) ;
00111
00112 #elif JEOCIE_MODE == DISABLE
00113     CLR_BIT( ADC->CR1, JEOCIE ) ;
00114
00115 #endif
00116
00117 // EN/DIS Regular INT:
00118 #if EOCIE_MODE == ENABLE
00119     SET_BIT( ADC->CR1, EOCIE ) ;
00120
00121 #elif EOCIE_MODE == DISABLE
00122     CLR_BIT( ADC->CR1, EOCIE ) ;
00123
00124 #endif
00125
00126 #endif
00127
00128 // EN/DIS Overrun interrupt:
00129 #if OVERRUN_INT == ENABLE
00130     SET_BIT( ADC->CR1, OVRIE ) ;
00131
00132 #elif OVERRUN_INT == DISABLE
00133     CLR_BIT( ADC->CR1, OVRIE ) ;
00134
00135 #endif
00136
00137 // Select Regular channel sequence length(1/2/3/.../16):
00138 #if REG_SQ_LENGTH == _1_CONV
00139
00140     CLR_BIT( ADC->SQR1, L0 ) ;
00141     CLR_BIT( ADC->SQR1, L1 ) ;
00142     CLR_BIT( ADC->SQR1, L2 ) ;
00143     CLR_BIT( ADC->SQR1, L3 ) ;
00144
00145 #elif REG_SQ_LENGTH == _2_CONV
00146
00147     SET_BIT( ADC->SQR1, L0 ) ;
00148     CLR_BIT( ADC->SQR1, L1 ) ;
00149     CLR_BIT( ADC->SQR1, L2 ) ;
00150     CLR_BIT( ADC->SQR1, L3 ) ;
00151
00152 #elif REG_SQ_LENGTH == _3_CONV
00153
00154     CLR_BIT( ADC->SQR1, L0 ) ;
00155     SET_BIT( ADC->SQR1, L1 ) ;
00156     CLR_BIT( ADC->SQR1, L2 ) ;
00157     CLR_BIT( ADC->SQR1, L3 ) ;
00158
00159 #elif REG_SQ_LENGTH == _4_CONV
00160
00161     SET_BIT( ADC->SQR1, L0 ) ;
00162     SET_BIT( ADC->SQR1, L1 ) ;
00163     CLR_BIT( ADC->SQR1, L2 ) ;
00164     CLR_BIT( ADC->SQR1, L3 ) ;
00165
00166 #elif REG_SQ_LENGTH == _5_CONV
00167
00168     CLR_BIT( ADC->SQR1, L0 ) ;
00169     CLR_BIT( ADC->SQR1, L1 ) ;
00170     SET_BIT( ADC->SQR1, L2 ) ;
00171     CLR_BIT( ADC->SQR1, L3 ) ;
00172
00173 #elif REG_SQ_LENGTH == _6_CONV
00174
00175     SET_BIT( ADC->SQR1, L0 ) ;
00176     CLR_BIT( ADC->SQR1, L1 ) ;
00177     SET_BIT( ADC->SQR1, L2 ) ;
00178     CLR_BIT( ADC->SQR1, L3 ) ;
00179
00180 #elif REG_SQ_LENGTH == _7_CONV
00181
00182     CLR_BIT( ADC->SQR1, L0 ) ;
00183     SET_BIT( ADC->SQR1, L1 ) ;
00184     SET_BIT( ADC->SQR1, L2 ) ;
00185     CLR_BIT( ADC->SQR1, L3 ) ;
00186
00187 #elif REG_SQ_LENGTH == _8_CONV
00188
00189     SET_BIT( ADC->SQR1, L0 ) ;
00190     SET_BIT( ADC->SQR1, L1 ) ;
00191     SET_BIT( ADC->SQR1, L2 ) ;
00192     CLR_BIT( ADC->SQR1, L3 ) ;
```

```

00196
00197 #elif REG_SQ_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_SQ_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_SQ_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_SQ_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_SQ_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_SQ_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_SQ_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_SQ_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 [ADC](#), [ADCPRE0](#), [ADON](#), [ALIGN](#), [APB2ENR_ADC1EN](#), [CLR_BIT](#), [CONT](#), [DISCEN](#), [DISCNUM0](#), [DISCNUM1](#), [DISCNUM2](#), [EOCIE](#), [JEOCIE](#), [L0](#), [L1](#), [L2](#), [L3](#), [MRCC_vEnablePeriphralCLK\(\)](#), [OVRIE](#), [RCC_APB2](#), [RES0](#), [RES1](#), [SCAN](#), and [SET_BIT](#).

Referenced by [HLDR_vInit\(\)](#).

7.62.1.2 MADC_vSelectChannel()

```

void MADC_vSelectChannel (
    u8_t A_u8ChannelNum )

```

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 [ADC](#), and [SQ1_BIT_MANIPULATION](#).

7.62.1.3 MADC_u16ConvertToDigital()

```
u16_t MADC_u16ConvertToDigital (
    u8_t A_u8ChannelNum )
```

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 [ADC](#), [CLR_BIT](#), [EOC](#), [FLAG_SET](#), [GET_BIT](#), [SET_BIT](#), [SQ1_BIT_MANIPULATION](#), and [SWSTART](#).

7.62.1.4 MADC_vDisable()

```
void MADC_vDisable (
    void )
```

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 [ADC](#), [ADON](#), [APB2ENR_ADC1EN](#), [CLR_BIT](#), [MRCC_vDisablePeriphralCLK\(\)](#), and [RCC_APB2](#).

7.62.1.5 MADC_vEnable()

```
void MADC_vEnable (
    void )
```

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 [ADC](#), [ADON](#), [APB2ENR_ADC1EN](#), [MRCC_vEnablePeriphralCLK\(\)](#), [RCC_APB2](#), and [SET_BIT](#).

7.62.1.6 MADC_vRegINT_Disable()

```
void MADC_vRegINT_Disable (
    void )
```

Definition at line 407 of file [ADC_program.c](#).

```
00408 {  
00409     CLR_BIT( ADC->CR1, EOCIE ) ;  
00410 }
```

References [ADC](#), [CLR_BIT](#), and [EOCIE](#).

7.62.1.7 MADC_vRegINTEnable()

```
void MADC_vRegINTEnable (
    void )
```

Definition at line 415 of file [ADC_program.c](#).

```
00416 {  
00417     SET_BIT( ADC->CR1, EOCIE ) ;  
00418 }
```

References [ADC](#), [EOCIE](#), and [SET_BIT](#).

7.62.1.8 MADC_vSetCallBack()

```
void MADC_vSetCallBack (
    void(*) (void) MEXTI_vpPointerTo_ISR_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 }
```

References [MADC_vpPointerToFunction](#), and [NULL](#).

7.62.1.9 ADC_IRQHandler()

```
void ADC_IRQHandler (
    void )
```

Definition at line 438 of file [ADC_program.c](#).

```
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 }
```

References [ADC](#), [CLR_BIT](#), [EOC](#), [JEOC](#), [MADC_vpPointerToFunction](#), and [NULL](#).

7.62.2 Variable Documentation

7.62.2.1 MADC_vpPointerToFunction

```
void(* MADC_vpPointerToFunction) (void) (
    void ) = ( (void*)0 )
```

Definition at line 23 of file [ADC_program.c](#).

Referenced by [ADC_IRQHandler\(\)](#), and [MADC_vSetCallBack\(\)](#).

7.63 ADC_program.c

[Go to the documentation of this file.](#)

```
00001 /* FILENAME: ADC_program
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Wed 12/14/2022
00005 */
00006
00007 /***** Include headers *****/
00008 /*           Functions implementations           */
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 /***** Functions implementations           */
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_vEnablePeriphralCLK( RCC_APB2, APB2ENR_ADC1EN ) ;
00035
00036     // EN/DIS power for ADC to be turned on/off:
00037 #if ADC_ON == ENABLE
00038     SET_BIT( ADC->CR2, ADON ) ;
00039
00040 #elif ADC_ON == DISABLE
00041     CLR_BIT( ADC->CR2, ADON ) ;
00042
00043 #endif
00044
00045 // Select ADC prescalers(2/4/6/8):
00046 #if ADC_PRESCALER == DIV_BY_2
00047     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00048     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00049
00050     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00051     CLR_BIT( ADC->CCR, ADCPRE0 ) ;
00052
00053 #elif ADC_PRESCALER == DIV_BY_4
00054     SET_BIT( ADC->CCR, ADCPRE0 ) ;
00055     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
00107     // EN/DIS Injected INT:
00108 #if JEOCIE_MODE == ENABLE
00109
00110     SET_BIT( ADC->CR1, JEOCIE ) ;
00111
00112 #elif JEOCIE_MODE == DISABLE
00113
00114     CLR_BIT( ADC->CR1, JEOCIE ) ;
00115
00116 #endif
00117
00118     // EN/DIS Regular INT:
00119 #if EOCIE_MODE == ENABLE
00120
00121     SET_BIT( ADC->CR1, EOCIE ) ;
00122
00123 #elif EOCIE_MODE == DISABLE
00124
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_SQ_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_SQ_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_SQ_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_SQ_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( VAR(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( VAR(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 coversion 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
00382     // DIS CLK on APB2 bus to be able to operate ADC peripheral:
00383     MRCC_vDisablePeripheralCLK( 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         MADC_vpPointerToFunction = MEXTI_vpPointerTo_ISR_function ;
00429     }
00430
00431 }
00432
00433 }
00434
00435
00436 //*****
00437 //*****
00438 FUNC(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 }
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461

```

7.64 COTS/MCAL/EXTI/EXTI_config.h File Reference

Macros

- #define EXTI_LINE0_EN ENABLE
- #define EXTI_LINE0_TRIGGER EXTI_FallingEdge

- #define EXTI_LINE0_EN ENABLE
- #define EXTI_LINE0_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE1_EN ENABLE
- #define EXTI_LINE1_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE2_EN ENABLE
- #define EXTI_LINE2_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE3_EN ENABLE
- #define EXTI_LINE3_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE4_EN ENABLE
- #define EXTI_LINE4_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE5_EN ENABLE
- #define EXTI_LINE5_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE6_EN ENABLE
- #define EXTI_LINE6_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE7_EN ENABLE
- #define EXTI_LINE7_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE8_EN ENABLE
- #define EXTI_LINE8_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE9_EN ENABLE
- #define EXTI_LINE9_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE10_EN ENABLE
- #define EXTI_LINE10_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE11_EN ENABLE
- #define EXTI_LINE11_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE12_EN ENABLE
- #define EXTI_LINE12_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE13_EN ENABLE
- #define EXTI_LINE13_TRIGGER EXTI_RisingEdge
- #define EXTI_LINE14_EN ENABLE
- #define EXTI_LINE14_TRIGGER EXTI_OnChange
- #define EXTI_LINE15_EN ENABLE
- #define EXTI_LINE15_TRIGGER EXTI_RisingEdge

7.64.1 Macro Definition Documentation

7.64.1.1 EXTI_LINE0_EN

```
#define EXTI_LINE0_EN ENABLE
```

Definition at line 18 of file [EXTI_config.h](#).

7.64.1.2 EXTI_LINE0_TRIGGER

```
#define EXTI_LINE0_TRIGGER EXTI_FallingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line 28 of file [EXTI_config.h](#).

7.64.1.3 EXTI_LINE1_EN

```
#define EXTI_LINE1_EN ENABLE
```

Definition at line [36](#) of file [EXTI_config.h](#).

7.64.1.4 EXTI_LINE1_TRIGGER

```
#define EXTI_LINE1_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line [46](#) of file [EXTI_config.h](#).

7.64.1.5 EXTI_LINE2_EN

```
#define EXTI_LINE2_EN ENABLE
```

Definition at line [54](#) of file [EXTI_config.h](#).

7.64.1.6 EXTI_LINE2_TRIGGER

```
#define EXTI_LINE2_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line [64](#) of file [EXTI_config.h](#).

7.64.1.7 EXTI_LINE3_EN

```
#define EXTI_LINE3_EN ENABLE
```

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

7.64.1.8 EXTI_LINE3_TRIGGER

```
#define EXTI_LINE3_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.9 EXTI_LINE4_EN

```
#define EXTI_LINE4_EN ENABLE
```

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

7.64.1.10 EXTI_LINE4_TRIGGER

```
#define EXTI_LINE4_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.11 EXTI_LINE5_EN

```
#define EXTI_LINE5_EN ENABLE
```

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

7.64.1.12 EXTI_LINE5_TRIGGER

```
#define EXTI_LINE5_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.13 EXTI_LINE6_EN

```
#define EXTI_LINE6_EN ENABLE
```

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

7.64.1.14 EXTI_LINE6_TRIGGER

```
#define EXTI_LINE6_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.15 EXTI_LINE7_EN

```
#define EXTI_LINE7_EN ENABLE
```

Definition at line 144 of file [EXTI_config.h](#).

7.64.1.16 EXTI_LINE7_TRIGGER

```
#define EXTI_LINE7_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.17 EXTI_LINE8_EN

```
#define EXTI_LINE8_EN ENABLE
```

Definition at line 162 of file [EXTI_config.h](#).

7.64.1.18 EXTI_LINE8_TRIGGER

```
#define EXTI_LINE8_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.19 EXTI_LINE9_EN

```
#define EXTI_LINE9_EN ENABLE
```

Definition at line 180 of file [EXTI_config.h](#).

7.64.1.20 EXTI_LINE9_TRIGGER

```
#define EXTI_LINE9_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.21 EXTI_LINE10_EN

```
#define EXTI_LINE10_EN ENABLE
```

Definition at line 198 of file [EXTI_config.h](#).

7.64.1.22 EXTI_LINE10_TRIGGER

```
#define EXTI_LINE10_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line 208 of file [EXTI_config.h](#).

7.64.1.23 EXTI_LINE11_EN

```
#define EXTI_LINE11_EN ENABLE
```

Definition at line 216 of file [EXTI_config.h](#).

7.64.1.24 EXTI_LINE11_TRIGGER

```
#define EXTI_LINE11_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line 226 of file [EXTI_config.h](#).

7.64.1.25 EXTI_LINE12_EN

```
#define EXTI_LINE12_EN ENABLE
```

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

7.64.1.26 EXTI_LINE12_TRIGGER

```
#define EXTI_LINE12_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.64.1.27 EXTI_LINE13_EN

```
#define EXTI_LINE13_EN ENABLE
```

Definition at line 251 of file [EXTI_config.h](#).

7.64.1.28 EXTI_LINE13_TRIGGER

```
#define EXTI_LINE13_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line 261 of file [EXTI_config.h](#).

7.64.1.29 EXTI_LINE14_EN

```
#define EXTI_LINE14_EN ENABLE
```

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

7.64.1.30 EXTI_LINE14_TRIGGER

```
#define EXTI_LINE14_TRIGGER EXTI_OnChange
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

Definition at line 279 of file [EXTI_config.h](#).

7.64.1.31 EXTI_LINE15_EN

```
#define EXTI_LINE15_EN ENABLE
```

Definition at line 287 of file [EXTI_config.h](#).

7.64.1.32 EXTI_LINE15_TRIGGER

```
#define EXTI_LINE15_TRIGGER EXTI_RisingEdge
```

Options: EXTI_FallingEdge , EXTI_RisingEdge , EXTI_OnChange

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

7.65 EXTI_config.h

[Go to the documentation of this file.](#)

```
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 /*options:
00035 *ENABLE
00036 *DISABLE
00037 */
00038 #define EXTI_LINE1_TRIGGER EXTI_RisingEdge
00039
00040 /***** *****/
00041
00042 /*options:
00043 *ENABLE
00044 *DISABLE
00045 */
00046 #define EXTI_LINE2_EN ENABLE
00047
00048 /***** *****/
00049
00050 /*options:
00051 *ENABLE
00052 *DISABLE
00053 */
00054 #define EXTI_LINE2_TRIGGER EXTI_RisingEdge
00055
00056 /***** *****/
00057
00058 #define EXTI_LINE2_TRIGGER EXTI_RisingEdge
00059
00060 /***** *****/
00061
00062 /*options:
00063 *ENABLE
00064 *DISABLE
00065 */
00066 /***** *****/
00067
00068 /*options:
00069 *ENABLE
00070 *DISABLE
00071 */
00072 #define EXTI_LINE3_EN ENABLE
00073
00074 /***** *****/
00075
00076 #define EXTI_LINE3_TRIGGER EXTI_RisingEdge
00077
00078 /***** *****/
00079
00080 /*options:
00081 *ENABLE
00082 *DISABLE
00083 */
00084 #define EXTI_LINE4_EN ENABLE
00085
00086 /***** *****/
00087
00088 /*options:
00089 *ENABLE
00090 *DISABLE
00091 */
00092 /***** *****/
00093
00094 #define EXTI_LINE4_TRIGGER EXTI_RisingEdge
00095
00096 /***** *****/
00097
00098 /*options:
00099 *ENABLE
00100 *DISABLE
00101 */
00102 /***** *****/
00103
00104 /*options:
00105 *ENABLE
00106 *DISABLE
00107 */
00108 #define EXTI_LINE5_EN ENABLE
00109
00110 /***** *****/
00111
00112 #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
00130 #define EXTI_LINE6_TRIGGER EXTI_RisingEdge
00131
00132 /****** */
00133
00134 /*options:
00135 *ENABLE
00136 *DISABLE
00137 */
00138 #define EXTI_LINE7_EN ENABLE
00139
00140 /****** */
00141
00142 /*options:
00143 *ENABLE
00144 *DISABLE
00145 */
00146 #define EXTI_LINE7_TRIGGER EXTI_RisingEdge
00147
00148 /****** */
00149
00150 /*options:
00151 *ENABLE
00152 *DISABLE
00153 */
00154 #define EXTI_LINE8_EN ENABLE
00155
00156 /****** */
00157
00158 /*options:
00159 *ENABLE
00160 *DISABLE
00161 */
00162 #define EXTI_LINE8_TRIGGER EXTI_RisingEdge
00163
00164 /****** */
00165
00166 #define EXTI_LINE8_TRIGGER EXTI_RisingEdge
00167
00168 /****** */
00169
00170 /*options:
00171 *ENABLE
00172 *DISABLE
00173 */
00174 #define EXTI_LINE9_EN ENABLE
00175
00176 /****** */
00177
00178 /*options:
00179 */
00180 #define EXTI_LINE9_TRIGGER EXTI_RisingEdge
00181
00182 /****** */
00183
00184 #define EXTI_LINE9_TRIGGER EXTI_RisingEdge
00185
00186 /****** */
00187
00188 /*options:
00189 *ENABLE
00190 *DISABLE
00191 */
00192 #define EXTI_LINE10_EN ENABLE
00193
00194 /****** */
00195
00196 /*options:
00197 */
00198 #define EXTI_LINE10_TRIGGER EXTI_RisingEdge
00199
00200 /****** */
00201
00202 #define EXTI_LINE10_TRIGGER EXTI_RisingEdge
00203
00204 /****** */
00205
00206 /*options:
00207 *ENABLE
00208 *DISABLE
00209 */
00210 /****** */
00211
00212 /*options:
00213 *ENABLE
00214 *DISABLE
00215 */
00216 #define EXTI_LINE11_EN ENABLE
00217
00218 /****** */
00219
00220 #define EXTI_LINE11_TRIGGER EXTI_RisingEdge
00221
00222 /****** */
00223
00224 /*options:
00225 *ENABLE
00226 *DISABLE
00227 */
00228 /****** */
00229
00230 /*options:
00231 *ENABLE
00232 *DISABLE
00233 */
00234 #define EXTI_LINE12_EN ENABLE
00235
00236 /****** */
00237
00238 #define EXTI_LINE12_TRIGGER EXTI_RisingEdge
00239
00240 /****** */
00241
00242 /*options:
00243 *ENABLE
00244 *DISABLE
00245 */
00246 /****** */
00247 /*options:
```

```

00248 *ENABLE
00249 *DISABLE
00250 */
00251 #define EXTI_LINE13_EN ENABLE
00252
00253 /*****
00254
00255 #define EXTI_LINE13_TRIGGER EXTI_RisingEdge
00256
00257 /*****
00258
00259 /*options:
00260 *ENABLE
00261 *DISABLE
00262 */
00263 #define EXTI_LINE14_EN ENABLE
00264
00265 /*****
00266
00267 /*options:
00268 *ENABLE
00269 *DISABLE
00270 */
00271 /*****
00272
00273 #define EXTI_LINE14_TRIGGER EXTI_OnChange
00274
00275 /*****
00276
00277 /*options:
00278 *ENABLE
00279 *DISABLE
00280 */
00281 /*****
00282
00283 /*options:
00284 *ENABLE
00285 *DISABLE
00286 */
00287 #define EXTI_LINE15_EN ENABLE
00288
00289 /*****
00290
00291 #define EXTI_LINE15_TRIGGER EXTI_RisingEdge
00292
00293 /*****
00294
00295
00296
00297
00298
00299 /*****
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315 #endif // _EXTI_config_H

```

7.66 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

- `MEXTI_vInit (void)`
Initialize the EXTI module.
- `MEXTI_vInit_WithStruct (P2VAR(EXTI_ConfigType) A_xINTConfig)`
Initialize the EXTI module with a certain configuration.
- `MEXTI_vEnableLine (VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus)`
Enable the interrupt on a certain line ID with a certain trigger status.
- `MEXTI_vDisableLine (VAR(u8_t) A_u8LineID)`
Disable the interrupt on a certain line ID.
- `MEXTI_vSWITrigger (VAR(u8_t) A_u8LineID)`
perform a software event interrupt on a certain line
- `MEXTI_vSetTrigger (VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus)`
Set the trigger status interrupt on a certain line ID.
- `MEXTI_vSetCallback (VAR(u8_t) A_u8LineID, void(*A_vFptr)(void))`
Set the callback when an interrupt occurs on a certain line ID.
- `MSYSCFG_vSetEXTIPort (VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8PortID)`
Enable the interrupt on a certain line in a certain port.

7.66.1 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](#).

7.66.2 Function Documentation

7.66.2.1 MEXTI_vInit()

```
void MEXTI_vInit (
    void )
```

Initialize the EXTI module.

Definition at line 42 of file [EXTI_program.c](#).

```
00043 {
00044     VAR(u8_t) L_u8ShiftedOffset = INITIAL_ZERO;
00046
00047 #if EXTI_LINE0_EN == ENABLE
00048     L_u8ShiftedOffset = 0;
00049 #elif EXTI_LINE1_EN == ENABLE
00050     L_u8ShiftedOffset = 1;
00051 #elif EXTI_LINE2_EN == ENABLE
00052     L_u8ShiftedOffset = 2;
00053 #elif EXTI_LINE3_EN == ENABLE
00054     L_u8ShiftedOffset = 3;
00055 #elif EXTI_LINE4_EN == ENABLE
00056     L_u8ShiftedOffset = 4;
00057 #elif EXTI_LINE5_EN == ENABLE
00058     L_u8ShiftedOffset = 5;
00059 #elif EXTI_LINE6_EN == ENABLE
00060     L_u8ShiftedOffset = 6;
00061 #elif EXTI_LINE7_EN == ENABLE
00062     L_u8ShiftedOffset = 7;
00063 #elif EXTI_LINE8_EN == ENABLE
00064     L_u8ShiftedOffset = 8;
00065 #elif EXTI_LINE9_EN == ENABLE
00066     L_u8ShiftedOffset = 9;
00067 #elif EXTI_LINE10_EN == ENABLE
00068     L_u8ShiftedOffset = 10;
00069 #elif EXTI_LINE11_EN == ENABLE
00070     L_u8ShiftedOffset = 11;
00071 #elif EXTI_LINE12_EN == ENABLE
00072     L_u8ShiftedOffset = 12;
00073 #elif EXTI_LINE13_EN == ENABLE
00074     L_u8ShiftedOffset = 13;
00075 #elif EXTI_LINE14_EN == ENABLE
```

```

00076     L_u8ShiftedOffset = 14;
00077 #elif EXTI_LINE15_EN == ENABLE
00078     L_u8ShiftedOffset = 15;
00079 #endif
00080
00081     // Get the enabled line ID.
00082     // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
00083
00084     // Clear all flags.
00085     MEXTI->PR = 0xffffffff;
00086
00087     // Enable EXTI on a line.
00088     MEXTI->IMR = 0;
00089
00090     MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00091
00092 // Set EXTI Triggered status on a line.
00093 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00094
00095     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00096
00097 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00098     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00099
00100 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00101     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00102     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00103
00104 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00105     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00106
00107 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00108     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00109
00110 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00111     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00112     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00113
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
00124 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00125     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00126
00127 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00128     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00129
00130 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00131     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00132     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00133
00134 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00135     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00136
00137 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00138     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00139
00140 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00141     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00142     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00143
00144 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00145     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00146     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00147
00148 #elif EXTI_LINE5_TRIGGER == EXTI_FallingEdge
00149     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00150
00151 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00152     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00153     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00154
00155 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00156     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00157     SET_BIT(MEXTI->FTSR, 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_LINE6_TRIGGER == EXTI_OnChange
00167     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00168     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);

```

```
00169
00171 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00172     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00173
00174 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00175     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00176
00177 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00178     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00179     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00180
00182 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge
00183     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00184
00185 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00186     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00187
00188 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00189     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00190     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00191
00193 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00194     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00195
00196 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00197     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00198
00199 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00200     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00201     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00202
00204 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00205     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00206
00207 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00208     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00209
00210 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00211     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00212     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00213
00215 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
00216     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00217
00218 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00219     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00220
00221 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00222     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00223     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00224
00226 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00227     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00228
00229 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00230     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00231
00232 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00233     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00234     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00235
00237 #elif EXTI_LINE13_TRIGGER == EXTI_RisingEdge
00238     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00239
00240 #elif EXTI_LINE13_TRIGGER == EXTI_FallingEdge
00241     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00242
00243 #elif EXTI_LINE13_TRIGGER == EXTI_OnChange
00244     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00245     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00246
00248 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00249     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00250
00251 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge
00252     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00253
00254 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00255     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00256     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00257
00259 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00260     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00261
00262 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00263     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00264
```

```

00265 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00266     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00267     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00268
00269 #endif
00270
00271 }

```

References [ENABLE](#), [INITIAL_ZERO](#), [MEXTI](#), [SET_BIT](#), and [VAR](#).

7.66.2.2 MEXTI_vInit_WithStruct()

```
void MEXTI_vInit_WithStruct (
    P2VAR(EXTI_ConfigType) A_xINTConfig )
```

Initialize the EXTI module with a certain configuration.

Parameters

in	A_xINTConfig	The configuration structure to initialize the interrupt with
----	--------------	--

7.66.2.3 MEXTI_vEnableLine()

```
void MEXTI_vEnableLine (
    VAR(u8_t) A_u8LineID,
    VAR(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

7.66.2.4 MEXTI_vDisableLine()

```
void MEXTI_vDisableLine (
    VAR(u8_t) A_u8LineID )
```

Disable the interrupt on a certain line ID.

Parameters

in	A_u8LineID	Which line to disable the interrupt on
----	------------	--

7.66.2.5 MEXTI_vSWITrigger()

```
void MEXTI_vSWITrigger (
    VAR(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
----	------------	--

7.66.2.6 MEXTI_vSetTrigger()

```
void MEXTI_vSetTrigger (
    VAR(u8_t) A_u8LineID,
    VAR(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

7.66.2.7 MEXTI_vSetCallback()

```
void MEXTI_vSetCallback (
    VAR(u8_t) A_u8LineID,
    void(*)(void) A_vFptr )
```

Set the callback when an interrupt occurs on a certain line ID.

Parameters

in	A_u8LineID	The line ID to set the callback on
in	A_vFptr	The callback to call when the interrupt occurs on a certain line

7.66.2.8 MSYSCFG_vSetEXTIPort()

```
void MSYSCFG_vSetEXTIPort (
```

```
VAR(u8_t) A_u8LineID,
VAR(u8_t) A_u8PortID )
```

Enable the interrupt on a certain line in a certain port.

Parameters

in	A_u8LineID	The line ID to enable the interrupt onto
in	A_u8PortID	Port that the line belongs to

7.67 EXTI_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 /* 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 *          Functions prototypes
00042 */
00046 void MEXTI_vInit(void);
00051 void MEXTI_vInit_WithStruct(P2VAR(EXTI_ConfigType) A_xINTConfig);
00052
00058 void MEXTI_vEnableLine(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus);
00059
00064 void MEXTI_vDisableLine(VAR(u8_t) A_u8LineID);
00065
00070 void MEXTI_vSWITrigger(VAR(u8_t) A_u8LineID);
00071
00077 void MEXTI_vSetTrigger(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus);
00078
00084 void MEXTI_vSetCallback(VAR(u8_t) A_u8LineID, void (*A_vFptr)(void));
00085
00091 void MSYSCFG_vSetEXTIPort(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8PortID);
00092
00104 #define EXTI_LINE0 (0)
00110 #define EXTI_LINE1 (1)
00116 #define EXTI_LINE2 (2)
00122 #define EXTI_LINE3 (3)
00128 #define EXTI_LINE4 (4)
00134 #define EXTI_LINE5 (5)
00140 #define EXTI_LINE6 (6)
00146 #define EXTI_LINE7 (7)
00152 #define EXTI_LINE8 (8)
00158 #define EXTI_LINE9 (9)
00164 #define EXTI_LINE10 (10)
00170 #define EXTI_LINE11 (11)
00176 #define EXTI_LINE12 (12)
00182 #define EXTI_LINE13 (13)
00188 #define EXTI_LINE14 (14)
00194 #define EXTI_LINE15 (15)
00195
00212 #define EXTI_FallingEdge (1)
00218 #define EXTI_RisingEdge (2)
00224 #define EXTI_OnChange (3)
00230 #endif // _EXTI_interface_H
```

7.68 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

7.68.1 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](#).

7.68.2 Macro Definition Documentation

7.68.2.1 EXTI IRQs

```
#define EXTI IRQs 23
```

Definition at line 109 of file [EXTI_private.h](#).

7.69 EXTI_private.h

[Go to the documentation of this file.](#)

```

00001 /* 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 MBXTI ((volatile P2VAR(EXTI_Type))(EXTI_BASE_ADDRESS))
00085 #define ENABLE (1)
00091 #define DISABLE (2)
00105 #define EXTI_MAX_EXTI_NUM (22)
00109 #define EXTI IRQs 23
00110
00111 #endif // _EXTI_private_H

```

7.70 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)
- void [MEXTI_vInit](#) (void)

Initialize the EXTI module.
- void [MEXTI_vInit_WithStruct](#) (EXTI_ConfigType *A_xINTConfig)
- void [MEXTI_vEnableLine](#) (u8_t A_u8LineID, u8_t A_u8TriggerStatus)
- void [MEXTI_vDisableLine](#) (u8_t A_u8LineID)
- void [MEXTI_vSWITrigger](#) (u8_t A_u8LineID)
- void [MEXTI_vSetTrigger](#) (u8_t A_u8LineID, u8_t A_u8TriggerStatus)
- void [MEXTI_vSetCallback](#) (u8_t A_u8LineID, void(*A_vFptr)(void))
- void [EXTI0_IRQHandler](#) (void)
- void [EXTI1_IRQHandler](#) (void)
- void [EXTI2_IRQHandler](#) (void)
- void [EXTI3_IRQHandler](#) (void)
- void [EXTI4_IRQHandler](#) (void)
- void [EXTI9_5_IRQHandler](#) (void)
- void [EXTI15_10_IRQHandler](#) (void)

7.70.1 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](#).

7.70.2 Function Documentation

7.70.2.1 MSYSCFG_vSetEXTIPort()

```
void MSYSCFG_vSetEXTIPort (
    u8_t A_u8LineID,
    u8_t A_u8PortID )
```

Definition at line 30 of file [EXTI_program.c](#).

```
00031 {
00032     VAR(u8_t) L_u8Index = A_u8LineID / 4;
00033     VAR(u8_t) L_u8ShiftAmount = A_u8LineID % 4;
00034
00035     CLR_BITS(MSYSCFG->EXTICR[L_u8Index], 0b1111, L_u8ShiftAmount, 4);
00036     SET_BITS(MSYSCFG->EXTICR[L_u8Index], A_u8PortID, L_u8ShiftAmount, 4);
00037 }
```

References [CLR_BITS](#), [MSYSCFG](#), [SET_BITS](#), and [VAR](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.70.2.2 MEXTI_vInit()

```
void MEXTI_vInit (
    void )
```

Initialize the EXTI module.

Definition at line 42 of file [EXTI_program.c](#).

```
00043 {
00044     VAR(u8_t) L_u8ShiftedOffset = INITIAL_ZERO;
00046
00047 #if EXTI_LINE0_EN == ENABLE
00048     L_u8ShiftedOffset = 0;
00049 #elif EXTI_LINE1_EN == ENABLE
00050     L_u8ShiftedOffset = 1;
00051 #elif EXTI_LINE2_EN == ENABLE
00052     L_u8ShiftedOffset = 2;
00053 #elif EXTI_LINE3_EN == ENABLE
00054     L_u8ShiftedOffset = 3;
00055 #elif EXTI_LINE4_EN == ENABLE
00056     L_u8ShiftedOffset = 4;
00057 #elif EXTI_LINE5_EN == ENABLE
00058     L_u8ShiftedOffset = 5;
00059 #elif EXTI_LINE6_EN == ENABLE
00060     L_u8ShiftedOffset = 6;
00061 #elif EXTI_LINE7_EN == ENABLE
00062     L_u8ShiftedOffset = 7;
00063 #elif EXTI_LINE8_EN == ENABLE
00064     L_u8ShiftedOffset = 8;
00065 #elif EXTI_LINE9_EN == ENABLE
00066     L_u8ShiftedOffset = 9;
00067 #elif EXTI_LINE10_EN == ENABLE
00068     L_u8ShiftedOffset = 10;
00069 #elif EXTI_LINE11_EN == ENABLE
00070     L_u8ShiftedOffset = 11;
00071 #elif EXTI_LINE12_EN == ENABLE
00072     L_u8ShiftedOffset = 12;
00073 #elif EXTI_LINE13_EN == ENABLE
00074     L_u8ShiftedOffset = 13;
00075 #elif EXTI_LINE14_EN == ENABLE
00076     L_u8ShiftedOffset = 14;
00077 #elif EXTI_LINE15_EN == ENABLE
00078     L_u8ShiftedOffset = 15;
00079 #endif
00080
00081     // Get the enabled line ID.
00082     // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
00083
00084     // Clear all flags.
00085     MEXTI->PR = 0xffffffff;
00086
00087     // Enable EXTI on a line.
00088     MEXTI->IMR = 0;
00089
00090     MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00091
00092 // Set EXTI Triggered status on a line.
00093 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00094
00095     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00096
00097 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00098     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00099
00100 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00101     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00102     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00103
00105 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00106     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00107
00108 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00109     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00110
00111 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00112     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00113     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00114
00116 #elif EXTI_LINE2_TRIGGER == EXTI_RisingEdge
00117     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00118
00119 #elif EXTI_LINE2_TRIGGER == EXTI_FallingEdge
```

```
00120     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00121
00122 #elif EXTI_LINE2_TRIGGER == EXTI_OnChange
00123     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00124     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00125
00127 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00128     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00129
00130 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00131     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00132
00133 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00134     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00135     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00136
00138 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00139     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00140
00141 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00142     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00143
00144 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00145     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00146     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00147
00149 #elif EXTI_LINES_TRIGGER == EXTI_RisingEdge
00150     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00151
00152 #elif EXTI_LINES_TRIGGER == EXTI_FallingEdge
00153     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00154
00155 #elif EXTI_LINES_TRIGGER == EXTI_OnChange
00156     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00157     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00158
00160 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00161     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00162
00163 #elif EXTI_LINE6_TRIGGER == EXTI_FallingEdge
00164     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00165
00166 #elif EXTI_LINE6_TRIGGER == EXTI_OnChange
00167     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00168     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00169
00171 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00172     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00173
00174 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00175     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00176
00177 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00178     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00179     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00180
00182 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge
00183     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00184
00185 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00186     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00187
00188 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00189     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00190     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00191
00193 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00194     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00195
00196 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00197     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00198
00199 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00200     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00201     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00202
00204 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00205     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00206
00207 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00208     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00209
00210 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00211     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00212     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00213
00215 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
```

```

00216     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00217
00218 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00219     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00220
00221 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00222     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00223     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00224
00225 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00226     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00227
00228 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00229     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00230
00231 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00232     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00233     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00234
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
00245 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00246     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00247
00248 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge
00249     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00250
00251 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00252     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00253
00254 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00255     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00256     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00257
00258 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00259     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00260
00261 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00262     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00263
00264 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00265     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00266     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00267
00268 #endif
00270
00271 }

```

References [ENABLE](#), [INITIAL_ZERO](#), [MEXTI](#), [SET_BIT](#), and [VAR](#).

7.70.2.3 MEXTI_vInit_WithStruct()

```

void MEXTI_vInit_WithStruct (
    EXTI_ConfigType * A_xINTConfig )

```

Definition at line 276 of file [EXTI_program.c](#).

```

00277 {
00278
00279     MEXTI_vEnableLine( A_xINTConfig->LineNum, A_xINTConfig->TriggerStatus ) ;
00280
00281     MSYSCFG_vSetEXTIPort( A_xINTConfig->LineNum, A_xINTConfig->PortNum ) ;
00282
00283 }

```

References [EXTI_ConfigType::LineNum](#), [MEXTI_vEnableLine\(\)](#), [MSYSCFG_vSetEXTIPort\(\)](#), [EXTI_ConfigType::PortNum](#), and [EXTI_ConfigType::TriggerStatus](#).

7.70.2.4 MEXTI_vEnableLine()

```
void MEXTI_vEnableLine (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )
```

Definition at line 288 of file [EXTI_program.c](#).

```
00289 {
00290
00291     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00292     {
00293         SET_BIT(MEXTI->IMR, A_u8LineID);
00295         MEXTI_vSetTrigger(A_u8LineID, A_u8TriggerStatus);
00297     }
00298 }
00299
00300 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), [MEXTI_vSetTrigger\(\)](#), and [SET_BIT](#).

Referenced by [MEXTI_vInit_WithStruct\(\)](#).

7.70.2.5 MEXTI_vDisableLine()

```
void MEXTI_vDisableLine (
    u8_t A_u8LineID )
```

Definition at line 305 of file [EXTI_program.c](#).

```
00306 {
00307
00308     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00309     {
00310         CLR_BIT(MEXTI->IMR, A_u8LineID);
00311     }
00312
00313 }
```

References [CLR_BIT](#), [EXTI_MAX_EXTI_NUM](#), and [MEXTI](#).

7.70.2.6 MEXTI_vSWITrigger()

```
void MEXTI_vSWITrigger (
    u8_t A_u8LineID )
```

Definition at line 318 of file [EXTI_program.c](#).

```
00319 {
00320
00321     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00322     {
00323         SET_BIT(MEXTI->SWIER, A_u8LineID);
00324     }
00325
00326 }
```

References [EXTI_MAX_EXTI_NUM](#), [MEXTI](#), and [SET_BIT](#).

7.70.2.7 MEXTI_vSetTrigger()

```
void MEXTI_vSetTrigger (
    u8_t A_u8LineID,
    u8_t A_u8TriggerStatus )
```

Definition at line 331 of file [EXTI_program.c](#).

```
00332 {
00333
00334     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00335     {
00336
00337         switch (A_u8TriggerStatus)
00338         {
00339             case EXTI_RisingEdge:
00340                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00341
00342                 CLR_BIT(MEXTI->FTSR, A_u8LineID);
00343                 break;
00344
00345             case EXTI_FallingEdge:
00346                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00347
00348                 CLR_BIT(MEXTI->RTSR, A_u8LineID);
00349                 break;
00350
00351             case EXTI_OnChange:
00352                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00353
00354                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00355                 break;
00356         }
00357     }
00358 }
```

References [CLR_BIT](#), [EXTI_FallingEdge](#), [EXTI_MAX_EXTI_NUM](#), [EXTI_OnChange](#), [EXTI_RisingEdge](#), [MEXTI](#), and [SET_BIT](#).

Referenced by [MEXTI_vEnableLine\(\)](#).

7.70.2.8 MEXTI_vSetCallback()

```
void MEXTI_vSetCallback (
    u8_t A_u8LineID,
    void(*)(void) A_vFptr )
```

Definition at line 364 of file [EXTI_program.c](#).

```
00365 {
00366
00367     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00368     {
00369         GS_vEXTI_Callback[A_u8LineID] = A_vFptr;
00370     }
00371 }
```

References [EXTI_MAX_EXTI_NUM](#).

7.70.2.9 EXTI0_IRQHandler()

```
void EXTI0_IRQHandler (
    void )
```

Definition at line 377 of file [EXTI_program.c](#).

```
00378 {
00379
00380     if (GS_vEXTI_Callback[EXTI_LINE0] != NULL)
00381     {
00382         GS_vEXTI_Callback[EXTI_LINE0]();
00383     }
00384
00385     // Clear the flag.
00386     SET_BIT(MEXTI->PR, EXTI_LINE0);
00387
00388 }
```

References [EXTI_LINE0](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.70.2.10 EXTI1_IRQHandler()

```
void EXTI1_IRQHandler (
    void )
```

Definition at line 393 of file [EXTI_program.c](#).

```
00394 {
00395
00396     if (GS_vEXTI_Callback[EXTI_LINE1] != NULL)
00397     {
00398         GS_vEXTI_Callback[EXTI_LINE1]();
00399     }
00400
00401     // Clear the flag.
00402     SET_BIT(MEXTI->PR, EXTI_LINE1);
00403
00404 }
```

References [EXTI_LINE1](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.70.2.11 EXTI2_IRQHandler()

```
void EXTI2_IRQHandler (
    void )
```

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
00420 }
```

References [EXTI_LINE2](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.70.2.12 EXTI3_IRQHandler()

```
void EXTI3_IRQHandler (
    void )
```

Definition at line 425 of file [EXTI_program.c](#).

```
00426 {
00427     if (GS_vEXTI_Callback[EXTI_LINE3] != NULL)
00428     {
00429         GS_vEXTI_Callback[EXTI_LINE3]();
00430     }
00431
00432
00433     // Clear the flag.
00434     SET_BIT(MEXTI->PR, EXTI_LINE3);
00435
00436 }
```

References [EXTI_LINE3](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.70.2.13 EXTI4_IRQHandler()

```
void EXTI4_IRQHandler (
    void )
```

Definition at line 441 of file [EXTI_program.c](#).

```
00442 {
00443
00444     if (GS_vEXTI_Callback[EXTI_LINE4] != NULL)
00445     {
00446         GS_vEXTI_Callback[EXTI_LINE4]();
00447     }
00448
00449     // Clear the flag.
00450     SET_BIT(MEXTI->PR, EXTI_LINE4);
00451
00452 }
```

References [EXTI_LINE4](#), [MEXTI](#), [NULL](#), and [SET_BIT](#).

7.70.2.14 EXTI9_5_IRQHandler()

```
void EXTI9_5_IRQHandler (
    void )
```

Definition at line 457 of file [EXTI_program.c](#).

```
00458 {
00459
00460 }
```

7.70.2.15 EXTI15_10_IRQHandler()

```
void EXTI15_10_IRQHandler (
    void )
```

Definition at line 465 of file [EXTI_program.c](#).

```
00466 {
00467
00468 }
```

7.71 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
0023 static void( *GS_vEXTI_Callback[EXTI IRQs] ) (void) = {NULL} ;
0024
0025
0026 /***** Functions implementations *****/
0027 /*           Functions implementations           */
0028 /***** Functions implementations *****/
0029
0030 FUNC(void) MSYSCFG_vSetEXTIPort(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8PortID)
0031 {
0032     VAR(u8_t) L_u8Index = A_u8LineID / 4;
0033     VAR(u8_t) L_u8ShiftAmount = A_u8LineID % 4;
0034
0035     CLR_BITs(MSYSCFG->EXTICR[L_u8Index], 0b1111, L_u8ShiftAmount, 4);
0036     SET_BITs(MSYSCFG->EXTICR[L_u8Index], A_u8PortID, L_u8ShiftAmount, 4);
0037 }
0038
0039
0040
0041
0042 FUNC(void) MEXTI_vInit(void)
0043 {
0044
0045     VAR(u8_t) L_u8ShiftedOffset = INITIAL_ZERO;
0046
0047 #if EXTI_LINE0_EN == ENABLE
0048     L_u8ShiftedOffset = 0;
0049 #elif EXTI_LINE1_EN == ENABLE
0050     L_u8ShiftedOffset = 1;
0051 #elif EXTI_LINE2_EN == ENABLE
0052     L_u8ShiftedOffset = 2;
0053 #elif EXTI_LINE3_EN == ENABLE
0054     L_u8ShiftedOffset = 3;
0055 #elif EXTI_LINE4_EN == ENABLE
0056     L_u8ShiftedOffset = 4;
0057 #elif EXTI_LINE5_EN == ENABLE
0058     L_u8ShiftedOffset = 5;
0059 #elif EXTI_LINE6_EN == ENABLE
0060     L_u8ShiftedOffset = 6;
0061 #elif EXTI_LINE7_EN == ENABLE
0062     L_u8ShiftedOffset = 7;
0063 #elif EXTI_LINE8_EN == ENABLE
0064     L_u8ShiftedOffset = 8;
0065 #elif EXTI_LINE9_EN == ENABLE
0066     L_u8ShiftedOffset = 9;
0067 #elif EXTI_LINE10_EN == ENABLE
0068     L_u8ShiftedOffset = 10;
0069 #elif EXTI_LINE11_EN == ENABLE
0070     L_u8ShiftedOffset = 11;
0071 #elif EXTI_LINE12_EN == ENABLE
0072     L_u8ShiftedOffset = 12;
0073 #elif EXTI_LINE13_EN == ENABLE
0074     L_u8ShiftedOffset = 13;
0075 #elif EXTI_LINE14_EN == ENABLE
0076     L_u8ShiftedOffset = 14;
0077 #elif EXTI_LINE15_EN == ENABLE
0078     L_u8ShiftedOffset = 15;
0079 #endif
0080
0081 // Get the enabled line ID.
0082 // GS_u8EXTI_EnabledLine_ID = L_u8ShiftedOffset;
0083
0084 // Clear all flags.
0085 MEXTI->PR = 0xffffffff;
0086
0087 // Enable EXTI on a line.

```

```

00088     MEXTI->IMR = 0;
00089
00090     MEXTI->IMR |= (ENABLE << L_u8ShiftedOffset);
00091
00092 // Set EXTI Triggered status on a line.
00093 #if EXTI_LINE0_TRIGGER == EXTI_RisingEdge
00094
00095     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00096
00097 #elif EXTI_LINE0_TRIGGER == EXTI_FallingEdge
00098     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00099
00100 #elif EXTI_LINE0_TRIGGER == EXTI_OnChange
00101     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00102     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00103
00105 #elif EXTI_LINE1_TRIGGER == EXTI_RisingEdge
00106     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00107
00108 #elif EXTI_LINE1_TRIGGER == EXTI_FallingEdge
00109     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00110
00111 #elif EXTI_LINE1_TRIGGER == EXTI_OnChange
00112     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00113     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00114
00116 #elif EXTI_LINE2_TRIGGER == EXTI_RisingEdge
00117     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00118
00119 #elif EXTI_LINE2_TRIGGER == EXTI_FallingEdge
00120     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00121
00122 #elif EXTI_LINE2_TRIGGER == EXTI_OnChange
00123     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00124     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00125
00127 #elif EXTI_LINE3_TRIGGER == EXTI_RisingEdge
00128     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00129
00130 #elif EXTI_LINE3_TRIGGER == EXTI_FallingEdge
00131     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00132
00133 #elif EXTI_LINE3_TRIGGER == EXTI_OnChange
00134     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00135     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00136
00138 #elif EXTI_LINE4_TRIGGER == EXTI_RisingEdge
00139     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00140
00141 #elif EXTI_LINE4_TRIGGER == EXTI_FallingEdge
00142     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00143
00144 #elif EXTI_LINE4_TRIGGER == EXTI_OnChange
00145     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00146     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00147
00149 #elif EXTI_LINE5_TRIGGER == EXTI_RisingEdge
00150     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00151
00152 #elif EXTI_LINE5_TRIGGER == EXTI_FallingEdge
00153     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00154
00155 #elif EXTI_LINE5_TRIGGER == EXTI_OnChange
00156     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00157     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00158
00160 #elif EXTI_LINE6_TRIGGER == EXTI_RisingEdge
00161     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00162
00163 #elif EXTI_LINE6_TRIGGER == EXTI_FallingEdge
00164     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00165
00166 #elif EXTI_LINE6_TRIGGER == EXTI_OnChange
00167     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00168     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00169
00171 #elif EXTI_LINE7_TRIGGER == EXTI_RisingEdge
00172     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00173
00174 #elif EXTI_LINE7_TRIGGER == EXTI_FallingEdge
00175     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00176
00177 #elif EXTI_LINE7_TRIGGER == EXTI_OnChange
00178     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00179     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00180
00182 #elif EXTI_LINE8_TRIGGER == EXTI_RisingEdge

```

```
00183     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00184
00185 #elif EXTI_LINE8_TRIGGER == EXTI_FallingEdge
00186     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00187
00188 #elif EXTI_LINE8_TRIGGER == EXTI_OnChange
00189     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00190     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00191
00193 #elif EXTI_LINE9_TRIGGER == EXTI_RisingEdge
00194     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00195
00196 #elif EXTI_LINE9_TRIGGER == EXTI_FallingEdge
00197     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00198
00199 #elif EXTI_LINE9_TRIGGER == EXTI_OnChange
00200     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00201     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00202
00204 #elif EXTI_LINE10_TRIGGER == EXTI_RisingEdge
00205     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00206
00207 #elif EXTI_LINE10_TRIGGER == EXTI_FallingEdge
00208     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00209
00210 #elif EXTI_LINE10_TRIGGER == EXTI_OnChange
00211     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00212     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00213
00215 #elif EXTI_LINE11_TRIGGER == EXTI_RisingEdge
00216     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00217
00218 #elif EXTI_LINE11_TRIGGER == EXTI_FallingEdge
00219     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00220
00221 #elif EXTI_LINE11_TRIGGER == EXTI_OnChange
00222     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00223     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00224
00226 #elif EXTI_LINE12_TRIGGER == EXTI_RisingEdge
00227     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00228
00229 #elif EXTI_LINE12_TRIGGER == EXTI_FallingEdge
00230     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00231
00232 #elif EXTI_LINE12_TRIGGER == EXTI_OnChange
00233     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00234     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00235
00237 #elif EXTI_LINE13_TRIGGER == EXTI_RisingEdge
00238     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00239
00240 #elif EXTI_LINE13_TRIGGER == EXTI_FallingEdge
00241     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00242
00243 #elif EXTI_LINE13_TRIGGER == EXTI_OnChange
00244     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00245     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00246
00248 #elif EXTI_LINE14_TRIGGER == EXTI_RisingEdge
00249     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00250
00251 #elif EXTI_LINE14_TRIGGER == EXTI_FallingEdge
00252     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00253
00254 #elif EXTI_LINE14_TRIGGER == EXTI_OnChange
00255     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00256     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00257
00259 #elif EXTI_LINE15_TRIGGER == EXTI_RisingEdge
00260     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00261
00262 #elif EXTI_LINE15_TRIGGER == EXTI_FallingEdge
00263     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00264
00265 #elif EXTI_LINE15_TRIGGER == EXTI_OnChange
00266     SET_BIT(MEXTI->RTSR, L_u8ShiftedOffset);
00267     SET_BIT(MEXTI->FTSR, L_u8ShiftedOffset);
00268
00269 #endif
00270
00271 }
00272
00273
00274 //*****
00275 //*****
```

```

00275
00276 FUNC(void) MEXTI_vInit_WithStruct(P2VAR(EXTI_ConfigType) A_xINTConfig)
00277 {
00278
00279     MEXTI_vEnableLine( A_xINTConfig->LineNum, A_xINTConfig->TriggerStatus ) ;
00280
00281     MSYSCFG_vSetEXTIPort( A_xINTConfig->LineNum, A_xINTConfig->PortNum ) ;
00282
00283 }
00284
00285
00286 /*****
00287 *****/
00288 FUNC(void) MEXTI_vEnableLine(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus)
00289 {
00290
00291     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00292     {
00293
00294         SET_BIT(MEXTI->IMR, A_u8LineID);
00295
00296         MEXTI_vSetTrigger(A_u8LineID, A_u8TriggerStatus);
00297
00298     }
00299
00300 }
00301
00302
00303 /*****
00304 *****/
00305 FUNC(void) MEXTI_vDisableLine(VAR(u8_t) A_u8LineID)
00306 {
00307
00308     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00309     {
00310         CLR_BIT(MEXTI->IMR, A_u8LineID);
00311     }
00312
00313 }
00314
00315
00316 /*****
00317 *****/
00318 FUNC(void) MEXTI_vSWITrigger(VAR(u8_t) A_u8LineID)
00319 {
00320
00321     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00322     {
00323         SET_BIT(MEXTI->SWIER, A_u8LineID);
00324     }
00325
00326 }
00327
00328
00329 /*****
00330 *****/
00331 FUNC(void) MEXTI_vSetTrigger(VAR(u8_t) A_u8LineID, VAR(u8_t) A_u8TriggerStatus)
00332 {
00333
00334     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00335     {
00336
00337         switch (A_u8TriggerStatus)
00338         {
00339             case EXTI_RisingEdge:
00340                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00341
00342                 CLR_BIT(MEXTI->FTSR, A_u8LineID);
00343                 break;
00344
00345             case EXTI_FallingEdge:
00346                 SET_BIT(MEXTI->FTSR, A_u8LineID);
00347
00348                 CLR_BIT(MEXTI->RTSR, A_u8LineID);
00349                 break;
00350
00351             case EXTI_OnChange:
00352                 SET_BIT(MEXTI->RTSR, A_u8LineID);
00353

```

```
00354     SET_BIT(MEXTI->FTSR, A_u8LineID);
00355     break;
00356 }
00357 }
00358 }
00359 }
00360
00361 //*****
00362 //*****
00363
00364 FUNC(void) MEXTI_vSetCallback(VAR(u8_t) A_u8LineID, P2FUNC(VAR(void), A_vFptr) (void))
00365 {
00366
00367     if (A_u8LineID < EXTI_MAX_EXTI_NUM)
00368     {
00369         GS_vEXTI_Callback[A_u8LineID] = A_vFptr;
00370     }
00371
00372 }
00373
00374
00375 //*****
00376 //*****
00377 FUNC(void) EXTIO_IRQHandler(void)
00378 {
00379
00380     if (GS_vEXTI_Callback[EXTI_LINE0] != NULL)
00381     {
00382         GS_vEXTI_Callback[EXTI_LINE0]();
00383     }
00384
00385     // Clear the flag.
00386     SET_BIT(MEXTI->PR, EXTI_LINE0);
00387
00388 }
00389
00390
00391 //*****
00392 //*****
00393 FUNC(void) EXTI1_IRQHandler(void)
00394 {
00395
00396     if (GS_vEXTI_Callback[EXTI_LINE1] != NULL)
00397     {
00398         GS_vEXTI_Callback[EXTI_LINE1]();
00399     }
00400
00401     // Clear the flag.
00402     SET_BIT(MEXTI->PR, EXTI_LINE1);
00403
00404 }
00405
00406
00407 //*****
00408 //*****
00409 FUNC(void) EXTI2_IRQHandler(void)
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
00420 }
00421
00422
00423 //*****
00424 //*****
00425 FUNC(void) EXTI3_IRQHandler(void)
00426 {
00427
00428     if (GS_vEXTI_Callback[EXTI_LINE3] != NULL)
00429     {
00430         GS_vEXTI_Callback[EXTI_LINE3]();
```

```

00431     }
00432
00433     // Clear the flag.
00434     SET_BIT(MEXTI->PR, EXTI_LINE3);
00435
00436 }
00437
00438
00439 /******
00440 *****/
00441 FUNC(void) EXTI4_IRQHandler(void)
00442 {
00443
00444     if (GS_vEXTI_Callback[EXTI_LINE4] != NULL)
00445     {
00446         GS_vEXTI_Callback[EXTI_LINE4]();
00447     }
00448
00449     // Clear the flag.
00450     SET_BIT(MEXTI->PR, EXTI_LINE4);
00451
00452 }
00453
00454
00455 /******
00456 *****/
00457 FUNC(void) EXTI9_5_IRQHandler(void)
00458 {
00459
00460 }
00461
00462
00463 /******
00464 *****/
00465 FUNC(void) EXTI15_10_IRQHandler(void)
00466 {
00467
00468 }
00469
00470
00471 /******
00472 *****/
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491

```

7.72 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))

7.72.1 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](#).

7.73 SYSCFG_private.h

[Go to the documentation of this file.](#)

```
00001
00010 /* Header file guard */
00011 #ifndef MCAL_EXTI_SYSCFG_PRV_H_
00012 #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_ */
```

7.74 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`
- `#define PORTB_BIT_MANIPULATION 0x001C`

7.74.1 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](#).

7.74.2 Macro Definition Documentation

7.74.2.1 GPIOA_PIN_POS

```
#define GPIOA_PIN_POS (0b1110000000000000)
lock PA13, PA14 and PA15
```

Definition at line [22](#) of file [GPIO_config.h](#).

7.74.2.2 GPIOB_PIN_POS

```
#define GPIOB_PIN_POS (0b0000100000011100)
lock PB2, PB3 and PB4
```

Definition at line [29](#) of file [GPIO_config.h](#).

7.74.2.3 LCKK_BIT_POS

```
#define LCKK_BIT_POS (16U)
```

Position of LCKK bit.

Definition at line 36 of file [GPIO_config.h](#).

7.74.2.4 PORTA_BIT_MANIPULATION

```
#define PORTA_BIT_MANIPULATION 0xE000
```

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

7.74.2.5 PORTB_BIT_MANIPULATION

```
#define PORTB_BIT_MANIPULATION 0x001C
```

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

7.75 GPIO_config.h

[Go to the documentation of this file.](#)

```
00001
00009 #ifndef _GPIO_config_H
00010 #define _GPIO_config_H
00011
00012
00013 /*****// *****
00014 // GPIOx configurations //
00015 /*****// *****
00016
00022 #define GPIOA_PIN_POS (0b1110000000000000) //I want to lock PA13,14,15.
00023
00029 #define GPIOB_PIN_POS (0b0000100000011100) //I want to lock PB2,3,4,11.
00030
00036 #define LCKK_BIT_POS (16U) //Position of LCKK bit.
00037
00038
00039 #define PORTA_BIT_MANIPULATION 0xE000
00040
00041 #define PORTB_BIT_MANIPULATION 0x001C
00042
00043
00044 #endif // _GPIO_config_H
```

7.76 COTS/MCAL/GPIO/GPIO_interface.h File Reference

This file contains the interfacing information for the GPIO module.

Data Structures

- struct **MGPIOp_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)

- `#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.
- `#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
- #define GPIOA_LOW_NIBBLE_LOW 0x00

Functions

- void MGPIOx_vLockedPins (void)
Locks the prohibited GPIO PINs.
- void MGPIOx_vSetPinMode (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8Mode)
Sets a certain pin's mode on a specific port.
- void MGPIOx_vSetPinOutputType (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8OutputType)
Sets a certain pin's output type on a specific port.
- void MGPIOx_vSetPinOutputSpeed (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8OutputSpeed)
Sets a certain pin's output speed on a specific port.
- void MGPIOx_vSetPinInputPullType (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8InputPullType)
Sets a certain pin's input pull type on a specific port.
- u8_t MGPIOx_u8GetPinValue (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID)
Gets the value currently on a certain pin.
- void MGPIOx_vSetPinValue (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8PinValue)
Sets a certain pin's output value on a specific port.
- void MGPIOx_vSetResetAtomic (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8SetResetPinValue)
Resets a certain pin's output value on a specific port.
- void MGPIOx_vSetAlternateFunctionON (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8AFID)
Applies an alternative function on a certain pin.
- void MGPIOx_vSetPortConfigLock (VAR(u8_t) A_u8PortID)
Updates a port's configuration lock.
- void MGPIOx_vInit (P2VAR(MGPIOx_ConfigType) A_xPinConfig)
Initialize the GPIO with a certain configuration.
- void MGPIOx_vTogglePinValue (VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID)
Toggles a certain's pin's value on a certain port.
- void GPIO_vSetPortValue (VAR(u8_t) A_u8PortID, VAR(u16_t) A_u16PortValue)
Sets a certain port's output value.

7.76.1 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](#).

7.76.2 Macro Definition Documentation

7.76.2.1 GPIOA_LOW_NIBBLE_HIGH

```
#define GPIOA_LOW_NIBBLE_HIGH 0xFF
```

Definition at line [570](#) of file [GPIO_interface.h](#).

7.76.2.2 GPIOA_LOW_NIBBLE_LOW

```
#define GPIOA_LOW_NIBBLE_LOW 0x00
```

Definition at line [572](#) of file [GPIO_interface.h](#).

7.76.3 Function Documentation

7.76.3.1 MGPIox_vLockedPins()

```
void MGPIox_vLockedPins (
    void )
```

Locks the prohibited GPIO PINs.

Definition at line [24](#) of file [GPIO_program.c](#).

```
00025 {
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] = *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] should not change*/
00056 GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00057
00058 /* WR LCKR[16] = '1' + LCKR[2,3,4] should not change*/
00059 GPIOB->LCKRx = L_u32LockGPIOB ;
00060
00061 /* RD LCKR */
00062 L_u32LockGPIOB = GPIOB->LCKRx ;
00063
00064
00065 }

```

References [GPIOA](#), [GPIOA_PIN_POS](#), [GPIOB](#), [GPIOB_PIN_POS](#), [INITIAL_ZERO](#), [LCKK_BIT_POS](#), and [VAR](#).

7.76.3.2 MGPIox_vSetPinMode()

```

void MGPIox_vSetPinMode (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

7.76.3.3 MGPIox_vSetPinOutputType()

```

void MGPIox_vSetPinOutputType (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

7.76.3.4 MGPIOX_vSetPinOutputSpeed()

```
void MGPIOX_vSetPinOutputSpeed (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(u8_t) A_u8OutputSpeed )
```

Sets a certain pin's output speed 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_u8OutputSpeed	The output speed to apply on the pin

7.76.3.5 MGPIOX_vSetPinInputPullType()

```
void MGPIOX_vSetPinInputPullType (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

7.76.3.6 MGPIOX_u8GetPinValue()

```
u8_t MGPIOX_u8GetPinValue (
    VAR(u8_t) A_u8PortID,
    VAR(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

7.76.3.7 MGPIOx_vSetPinValue()

```
void MGPIOx_vSetPinValue (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

Referenced by [HDCM_vMoveBackward\(\)](#), [HDCM_vMoveForward\(\)](#), [HDCM_vStopMotor\(\)](#), [HLCD_vSendCommand\(\)](#), and [HLCD_vSendData\(\)](#).

7.76.3.8 MGPIOx_vSetResetAtomic()

```
void MGPIOx_vSetResetAtomic (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

7.76.3.9 MGPIOx_vSetAlternateFunctionON()

```
void MGPIOx_vSetAlternateFunctionON (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID,
    VAR(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

7.76.3.10 MGPIox_vSetPortConfigLock()

```
void MGPIox_vSetPortConfigLock (
    VAR(u8_t) A_u8PortID )
```

Updates a port's configuration lock.

Parameters

in	<i>A_u8PortID</i>	The port to update the pin's mode
----	-------------------	-----------------------------------

7.76.3.11 MGPIox_vInit()

```
void MGPIox_vInit (
    P2VAR(MGPIox_ConfigType) A_xPinConfig )
```

Initialize the GPIO with a certain configuration.

Parameters

in	<i>A_xPinConfig</i>	The initialization configuration for the GPIO
----	---------------------	---

Referenced by [HDCM_vInitMotor\(\)](#), [HLCD_vInit\(\)](#), [HLDR_vInit\(\)](#), [MSPI_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.76.3.12 MGPIox_vTogglePinValue()

```
void MGPIox_vTogglePinValue (
    VAR(u8_t) A_u8PortID,
    VAR(u8_t) A_u8PinID )
```

Toggles a certain's pin's value on a certain port.

Parameters

in	<i>A_u8PortID</i>	The port that the pin belongs to
in	<i>A_u8PinID</i>	The pin to toggle its value

7.76.3.13 GPIO_vSetValue()

```
void GPIO_vSetValue (
    VAR(u8_t) A_u8PortID,
    VAR(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

Referenced by [HLCR_vSendCommand\(\)](#), and [HLCR_vSendData\(\)](#).

7.77 GPIO_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
00010 #ifndef _GPIO_interface_H
00011 #define _GPIO_interface_H
00012
00018 typedef struct
00019 {
00023     u8_t Port;
00027     u8_t Pin;
00031     u8_t Mode;
00035     u8_t OutputType;
00039     u8_t OutputSpeed;
00043     u8_t InputType;
00047     u8_t AF_Type;
00048
00049 } MGPIOx_ConfigType;
00050
00051 /***** Functions prototypes *****/
00052 /* Functions prototypes */
00053 /***** Functions prototypes *****/
00054
00055 void MGPIOx_vLockedPins(void);
00056
00057 void MGPIOx_vSetPinMode(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8Mode);
00058
00059 void MGPIOx_vSetPinOutputType(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8OutputType);
00060
00061 void MGPIOx_vSetPinOutputSpeed(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8OutputSpeed);
00062
00063 void MGPIOx_vSetPinInputPullType(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
A_u8InputPullType);
00064
00065 u8_t MGPIOx_u8GetPinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID);
00066
00067 void MGPIOx_vSetPinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8PinValue);
00068
00069 void MGPIOx_vSetResetAtomic(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
A_u8SetResetPinValue);
00070
00071 void MGPIOx_vSetAlternateFunctionON(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8AFID);
00072
00073 void MGPIOx_vSetPortConfigLock(VAR(u8_t) A_u8PortID);
00074
00075 void MGPIOx_vInit(P2VAR(MGPIOx_ConfigType) A_xPinConfig);
00076
00077 void MGPIOx_vTogglePinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID);
00078
00079 void GPIO_vSetValue( VAR(u8_t) A_u8PortID, VAR(u16_t) A_u16PortValue );
```

```
00149
00150
00151 /****** Interfacing macros *****/
00152
00153 /***** */
00154
00166 #define GPIOx_MODE_INPUT (0b00)
00167
00173 #define GPIOx_MODE_OUTPUT (0b01)
00174
00180 #define GPIOx_MODE_AF (0b10)
00181
00187 #define GPIOx_MODE_ANALOG (0b11)
00188
00202 #define GPIO_PORTA (0)
00203
00209 #define GPIO_PORTB (1)
00210
00216 #define GPIO_PORTC (2)
00217
00231 #define GPIOx_OPENDRAIN (1)
00232
00238 #define GPIOx_PUSH_PULL (2)
00239
00253 #define GPIOx_LowSpeed (0b00)
00254
00260 #define GPIOx_MediumSpeed (0b01)
00261
00267 #define GPIOx_HighSpeed (0b10)
00268
00274 #define GPIOx_VeryHighSpeed (0b11)
00275
00289 #define GPIOx_NoPull (0b00)
00290
00296 #define GPIOx_PullUp (0b01)
00297
00303 #define GPIOx_PullDown (0b10)
00304
00318 #define GPIOx_HIGH (1)
00319
00325 #define GPIOx_LOW (2)
00326
00340 #define GPIOx_PIN0 (0)
00341
00347 #define GPIOx_PIN1 (1)
00348
00354 #define GPIOx_PIN2 (2)
00355
00361 #define GPIOx_PIN3 (3)
00362
00368 #define GPIOx_PIN4 (4)
00369
00375 #define GPIOx_PIN5 (5)
00376
00382 #define GPIOx_PIN6 (6)
00383
00389 #define GPIOx_PIN7 (7)
00390
00396 #define GPIOx_PIN8 (8)
00397
00403 #define GPIOx_PIN9 (9)
00404
00410 #define GPIOx_PIN10 (10)
00411
00417 #define GPIOx_PIN11 (11)
00418
00424 #define GPIOx_PIN12 (12)
00425
00431 #define GPIOx_PIN13 (13)
00432
00438 #define GPIOx_PIN14 (14)
00439
00445 #define GPIOx_PIN15 (15)
00446
00460 #define GPIOx_AF0 (0)
00461
00467 #define GPIOx_AF1 (1)
00468
00474 #define GPIOx_AF2 (2)
00475
00481 #define GPIOx_AF3 (3)
00482
00488 #define GPIOx_AF4 (4)
00489
00495 #define GPIOx_AF5 (5)
00496
00502 #define GPIOx_AF6 (6)
```

```

00503
00509 #define GPIOx_AF7 (7)
00510
00516 #define GPIOx_AF8 (8)
00517
00523 #define GPIOx_AF9 (9)
00524
00530 #define GPIOx_AF10 (10)
00531
00537 #define GPIOx_AF11 (11)
00538
00544 #define GPIOx_AF12 (12)
00545
00551 #define GPIOx_AF13 (13)
00552
00558 #define GPIOx_AF14 (14)
00559
00565 #define GPIOx_AF15 (15)
00566
00570 #define GPIOA_LOW_NIBBLE_HIGH 0xFF
00571
00572 #define GPIOA_LOW_NIBBLE_LOW 0x00
00573
00574
00575
00576
00577 #endif // _GPIO_interface_H

```

7.78 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))

7.78.1 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](#).

7.79 GPIO_private.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
00010 #ifndef _GPIO_private_H
00011 #define _GPIO_private_H
00012
00018 typedef struct
00019 {
00023     u32_t MODERx;
00024
00028     u32_t OTYPERx;
00029
00033     u32_t OSPEEDRx;
00034
00038     u32_t PUPDRx;
00039
00043     u32_t IDRx;
00044
00048     u32_t ODRx;
00049
00053     u32_t BSRRx;
00054
00058     u32_t LCKRx;
00059
00063     u32_t AFRLx;
00064
00068     u32_t AFRHx;
00069
00070 } GPIOx_MemoryMapType;
00071
00083 #define GPIOA_BASE_ADDRESS (0x40020000)
00084
00090 #define GPIOB_BASE_ADDRESS (0x40020400)
00091
00097 #define GPIOC_BASE_ADDRESS (0x40020800)
00098
00112 #define GPIOA ((volatile P2VAR(GPIOx_MemoryMapType)) (GPIOA_BASE_ADDRESS))
00113
00119 #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

```

7.80 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)
- void **MGPIOx_vSetPinOutputType** (**u8_t** A_u8PortID, **u8_t** A_u8PinID, **u8_t** A_u8OutputType)
- void **MGPIOx_vSetPinOutputSpeed** (**u8_t** A_u8PortID, **u8_t** A_u8PinID, **u8_t** A_u8OutputSpeed)
- void **MGPIOx_vSetPinInputPullType** (**u8_t** A_u8PortID, **u8_t** A_u8PinID, **u8_t** A_u8InputPullType)

- `u8_t MGPIox_u8GetPinValue (u8_t A_u8PortID, u8_t A_u8PinID)`
- `void MGPIox_vSetPinValue (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8PinValue)`
- `void MGPIox_vSetResetAtomic (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8SetResetPinValue)`
- `void MGPIox_vSetAlternateFunctionON (u8_t A_u8PortID, u8_t A_u8PinID, u8_t A_u8AFID)`
- `void MGPIox_vInit (MGPIox_ConfigType *A_xPinConfig)`
- `void MGPIox_vTogglePinValue (u8_t A_u8PortID, u8_t A_u8PinID)`
- `void GPIO_vSetPortValue (u8_t A_u8PortID, u16_t A_u16PortValue)`

7.80.1 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](#).

7.80.2 Function Documentation

7.80.2.1 MGPIox_vLockedPins()

```
void MGPIox_vLockedPins (
    void )
```

Locks the prohibited GPIO PINs.

Definition at line 24 of file [GPIO_program.c](#).

```
00025 {
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
00046     VAR(volatile u32_t) L_u32LockGPIOB = INITIAL_ZERO ;
00048
00049     /* Lock key write sequence */
00050
00051     /* WR LCKR[16] = '1' + LCKR[2,3,4] = #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] should not change*/
00057     GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00058
00059     /* WR LCKR[16] = '1' + LCKR[2,3,4] should not change*/
00060     GPIOB->LCKRx = L_u32LockGPIOB ;
00061
00062     /* RD LCKR */
00063     L_u32LockGPIOB = GPIOB->LCKRx ;
00064
00065 }
```

References [GPIOA](#), [GPIOA_PIN_POS](#), [GPIOB](#), [GPIOB_PIN_POS](#), [INITIAL_ZERO](#), [LCKK_BIT_POS](#), and [VAR](#).

7.80.2.2 MGPIox_vSetPinMode()

```

void MGPIox_vSetPinMode (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8Mode )
```

Definition at line 70 of file [GPIO_program.c](#).

```

00071 {
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 }
```

References [CLR_BITS](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITS](#).

Referenced by [MGPIox_vInit\(\)](#).

7.80.2.3 MGPIox_vSetPinOutputType()

```
void MGPIox_vSetPinOutputType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputType )
```

Definition at line 96 of file [GPIO_program.c](#).

```
00097 {
00098
00099     switch (A_u8OutputType)
00100     {
00101         case GPIOx_OPENDRAIN:
00102             switch (A_u8PortID)
00103             {
00104                 case GPIO_PORTA:
00105                     SET_BIT(GPIOA->OTYPERx, A_u8PinID);
00106                     break;
00107
00108                 case GPIO_PORTB:
00109                     SET_BIT(GPIOB->OTYPERx, A_u8PinID);
00110                     break;
00111
00112                 case GPIO_PORTC:
00113                     SET_BIT(GPIOC->OTYPERx, A_u8PinID);
00114                     break;
00115             }
00116             break;
00117
00118         case GPIOx_PUSH_PULL:
00119             switch(A_u8PortID)
00120             {
00121                 case GPIO_PORTA:
00122                     CLR_BIT(GPIOA->OTYPERx, A_u8PinID);
00123                     break;
00124
00125                 case GPIO_PORTB:
00126                     CLR_BIT(GPIOB->OTYPERx, A_u8PinID);
00127                     break;
00128
00129                 case GPIO_PORTC:
00130                     CLR_BIT(GPIOC->OTYPERx, A_u8PinID);
00131                     break;
00132             }
00133             break;
00134     }
00135 }
00136 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_OPENDRAIN](#), [GPIOx_PUSH_PULL](#), and [SET_BIT](#).

Referenced by [MGPIox_vInit\(\)](#).

7.80.2.4 MGPIox_vSetPinOutputSpeed()

```
void MGPIox_vSetPinOutputSpeed (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8OutputSpeed )
```

Definition at line 141 of file [GPIO_program.c](#).

```
00142 {
00143
00144     switch(A_u8PortID)
00145     {
00146         case GPIO_PORTA:
00147             CLR_BITS(GPIOA->OSPEEDR, 0b11, A_u8PinID, 2);
00148             SET_BITS(GPIOA->OSPEEDR, 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 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITs](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.80.2.5 MGPIOx_vSetPinInputPullType()

```

void MGPIOx_vSetPinInputPullType (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8InputPullType )
```

Definition at line 167 of file [GPIO_program.c](#).

```

00168 {
00169
00170     switch (A_u8PortID)
00171     {
00172         case GPIO_PORTA:
00173             CLR_BITS(GPIOA->PUPDRx, 0b11, A_u8PinID, 2);
00174             SET_BITS(GPIOA->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00175             break;
00176
00177         case GPIO_PORTB:
00178             CLR_BITS(GPIOB->PUPDRx, 0b11, A_u8PinID, 2);
00179             SET_BITS(GPIOB->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00180             break;
00181
00182         case GPIO_PORTC:
00183             CLR_BITS(GPIOC->PUPDRx, 0b11, A_u8PinID, 2);
00184             SET_BITS(GPIOC->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00185             break;
00186     }
00187
00188 }
```

References [CLR_BITs](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [SET_BITs](#).

Referenced by [MGPIOx_vInit\(\)](#).

7.80.2.6 MGPIOx_u8GetPinValue()

```

u8_t MGPIOx_u8GetPinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Definition at line 193 of file [GPIO_program.c](#).

```

00194 {
00195
00196     VAR(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     return L_u8PinValue;
00214
00215 }
```

References [GET_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [INITIAL_ZERO](#), and [VAR](#).

7.80.2.7 MGPIox_vSetValue()

```

void MGPIox_vSetValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8PinValue )
```

Definition at line 220 of file [GPIO_program.c](#).

```

00221 {
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         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 }
```

References [CLR_BIT](#), [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), [GPIOx_LOW](#), and [SET_BIT](#).

7.80.2.8 MGPIox_vSetResetAtomic()

```
void MGPIox_vSetResetAtomic (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8SetResetPinValue )
```

Definition at line 265 of file [GPIO_program.c](#).

```
00266 {
00267
00268     switch (A_u8PortID)
00269     {
00270
00271         case GPIO_PORTA:
00272
00273             switch (A_u8SetResetPinValue)
00274             {
00275                 case GPIOx_HIGH:
00276                     GPIOA->BSRRx = (1 << A_u8PinID);
00277                     break;
00278
00279                 case GPIOx_LOW:
00280                     GPIOA->BSRRx = (1 << (A_u8PinID + 16));
00281                     break;
00282             }
00283
00284             break;
00285
00286         case GPIO_PORTB:
00287
00288             switch (A_u8SetResetPinValue)
00289             {
00290                 case GPIOx_HIGH:
00291                     GPIOB->BSRRx = (1 << A_u8PinID);
00292                     break;
00293
00294                 case GPIOx_LOW:
00295                     GPIOB->BSRRx = (1 << (A_u8PinID + 16));
00296                     break;
00297             }
00298
00299             break;
00300
00301     case GPIO_PORTC:
00302
00303         switch (A_u8SetResetPinValue)
00304         {
00305             case GPIOx_HIGH:
00306                 GPIOC->BSRRx = (1 << A_u8PinID);
00307                 break;
00308
00309             case GPIOx_LOW:
00310                 GPIOC->BSRRx = (1 << (A_u8PinID + 16));
00311                 break;
00312         }
00313
00314         break;
00315     }
00316
00317 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [GPIOx_HIGH](#), and [GPIOx_LOW](#).

7.80.2.9 MGPIox_vSetAlternateFunctionON()

```
void MGPIox_vSetAlternateFunctionON (
    u8_t A_u8PortID,
    u8_t A_u8PinID,
    u8_t A_u8AFID )
```

Definition at line 322 of file [GPIO_program.c](#).

```

00323 {
00324     switch (A_u8PortID)
00325     {
00326         case GPIO_PORTA:
00327             switch (A_u8PinID)
00328             {
00329                 case GPIOx_PIN0:
00330                 case GPIOx_PIN1:
00331                 case GPIOx_PIN2:
00332                 case GPIOx_PIN3:
00333                 case GPIOx_PIN4:
00334                 case GPIOx_PIN5:
00335                 case GPIOx_PIN6:
00336                 case GPIOx_PIN7:
00337                     CLR_BITS(GPIOA->AFRLx, 0b1111, A_u8PinID, 4);
00338                     SET_BITS(GPIOA->AFRLx, A_u8AFID, A_u8PinID, 4);
00339                     break;
00340
00341
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, 4);
00351                     SET_BITS(GPIOA->AFRHx, A_u8AFID, A_u8PinID, 4);
00352                     break;
00353             }
00354         break;
00355
00356         case GPIO_PORTB:
00357             switch (A_u8PinID)
00358             {
00359                 case GPIOx_PIN0:
00360                 case GPIOx_PIN1:
00361                 case GPIOx_PIN2:
00362                 case GPIOx_PIN3:
00363                 case GPIOx_PIN4:
00364                 case GPIOx_PIN5:
00365                 case GPIOx_PIN6:
00366                 case GPIOx_PIN7:
00367                     CLR_BITS(GPIOB->AFRLx, 0b1111, A_u8PinID, 4);
00368                     SET_BITS(GPIOB->AFRLx, A_u8AFID, A_u8PinID, 4);
00369                     break;
00370
00371                 case GPIOx_PIN8:
00372                 case GPIOx_PIN9:
00373                 case GPIOx_PIN10:
00374                 case GPIOx_PIN11:
00375                 case GPIOx_PIN12:
00376                 case GPIOx_PIN13:
00377                 case GPIOx_PIN14:
00378                 case GPIOx_PIN15:
00379                     CLR_BITS(GPIOB->AFRHx, 0b1111, A_u8PinID-8, 4);
00380                     SET_BITS(GPIOB->AFRHx, A_u8AFID, A_u8PinID-8, 4);
00381                     break;
00382             }
00383         break;
00384
00385         case GPIO_PORTC:
00386             switch (A_u8PinID)
00387             {
00388                 case GPIOx_PIN0:
00389                 case GPIOx_PIN1:
00390                 case GPIOx_PIN2:
00391                 case GPIOx_PIN3:
00392                 case GPIOx_PIN4:
00393                 case GPIOx_PIN5:
00394                 case GPIOx_PIN6:
00395                 case GPIOx_PIN7:
00396                     CLR_BITS(GPIOC->AFRLx, 0b1111, A_u8PinID, 4);
00397                     SET_BITS(GPIOC->AFRLx, A_u8AFID, A_u8PinID, 4);
00398                     break;
00399
00400                 case GPIOx_PIN8:
00401                 case GPIOx_PIN9:
00402                 case GPIOx_PIN10:
00403                 case GPIOx_PIN11:
00404                 case GPIOx_PIN12:
00405                 case GPIOx_PIN13:
00406                 case GPIOx_PIN14:
00407                 case GPIOx_PIN15:
00408                     CLR_BITS(GPIOC->AFRHx, 0b1111, A_u8PinID, 4);
00409                     SET_BITS(GPIOC->AFRHx, A_u8AFID, A_u8PinID, 4);

```

```

00410         break;
00411     }
00412     break;
00413 }
00414 }
00415 }
```

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\(\)](#).

7.80.2.10 MGPIOx_vInit()

```
void MGPIOx_vInit (
    MGPIOx_ConfigType * A_xPinConfig )
```

Definition at line 420 of file [GPIO_program.c](#).

```

00421 {
00422
00423     MGPIOx_vSetPinMode          (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->Mode
00424     );
00425     MGPIOx_vSetPinOutputType   (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputType
00426     );
00427     MGPIOx_vSetPinOutputSpeed  (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputSpeed
00428     );
00429     MGPIOx_vSetPinInputPullType (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->InputType
00430     );
00431     MGPIOx_vSetAlternateFunctionON (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->AF_Type
00432     );
00433 }
```

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](#).

7.80.2.11 MGPIOx_vTogglePinValue()

```
void MGPIOx_vTogglePinValue (
    u8_t A_u8PortID,
    u8_t A_u8PinID )
```

Definition at line 434 of file [GPIO_program.c](#).

```

00435 {
00436
00437     switch (A_u8PortID)
00438     {
00439         case GPIO_PORTA:
00440             TOGGLE_BIT(GPIOA->ODRx, A_u8PinID);
00441             break;
00442
00443         case GPIO_PORTB:
00444             TOGGLE_BIT(GPIOB->ODRx, A_u8PinID);
00445             break;
00446
00447         case GPIO_PORTC:
00448             TOGGLE_BIT(GPIOC->ODRx, A_u8PinID);
00449             break;
00450     }
00451 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), and [TOGGLE_BIT](#).

7.80.2.12 GPIO_vSetValue()

```
void GPIO_vSetPortValue (
    u8_t A_u8PortID,
    u16_t A_u16PortValue )
```

Definition at line 457 of file [GPIO_program.c](#).

```
00458 {
00459     switch(A_u8PortID)
00460     {
00461
00462         case GPIO_PORTA : GPIOA->ODRx = ( ((GPIOA->ODRx & PORTA_BIT_MANIPULATION) | (A_u16PortValue))
00463             « 1 ) ; break; //Start writing from PIN1 to PIN8.
00464
00465         case GPIO_PORTB : GPIOB->ODRx = ( ((GPIOB->ODRx & PORTB_BIT_MANIPULATION) | (A_u16PortValue))
00466             « 5 ) ; break; //Start writing from PIN5 to PIN12.
00467         case GPIO_PORTC : GPIOC->ODRx ; break; // You shouldn't write on this port.
00468
00469     default : break;
00470 }
00471
00472 }
```

References [GPIO_PORTA](#), [GPIO_PORTB](#), [GPIO_PORTC](#), [GPIOA](#), [GPIOB](#), [GPIOC](#), [PORTA_BIT_MANIPULATION](#), and [PORTB_BIT_MANIPULATION](#).

7.81 GPIO_program.c

[Go to the documentation of this file.](#)

```
00001
00009 /***** Include headers *****/
00010 /*           Include headers           */
00011 /***** Include headers *****/
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 "GPIO_interface.h"
00017 #include "GPIO_private.h"
00018 #include "GPIO_config.h"
00019
00020 /***** Functions' implementations *****/
00021 /*           Functions' implementations           */
00022 /***** Functions' implementations *****/
00023
00024 FUNC(void) MGPIOx_vLockedPins(void)
00025 {
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] = ♦1♦ */
00051     L_u32LockGPIOB = ( (1UL « LCKK_BIT_POS) | (GPIOB_PIN_POS) ) ;
00052
00053
```

```

00054     GPIOB->LCKRx = L_u32LockGPIOB ;
00055
00056     /* WR LCKR[16] = '0' + LCKR[2,3,4] should not change*/
00057     GPIOB->LCKRx = (GPIOB_PIN_POS) ;
00058
00059     /* WR LCKR[16] = '1' + LCKR[2,3,4] should not change*/
00060     GPIOB->LCKRx = L_u32LockGPIOB ;
00061
00062     /* RD LCKR */
00063     L_u32LockGPIOB = GPIOB->LCKRx ;
00064
00065 }
00066
00067 /*****
00068 ****/
00069
00070 FUNC(void) MGPIox_vSetPinMode(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8Mode)
00071 {
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
00093 /*****
00094 ****/
00095
00096 FUNC(void) MGPIox_vSetPinOutputType(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
00097     A_u8OutputType)
00098
00099     switch (A_u8OutputType)
00100     {
00101         case GPIOX_OPENDRAIN:
00102             switch (A_u8PortID)
00103             {
00104                 case GPIO_PORTA:
00105                     SET_BIT(GPIOA->OTYPERx, A_u8PinID);
00106                     break;
00107
00108                 case GPIO_PORTB:
00109                     SET_BIT(GPIOB->OTYPERx, A_u8PinID);
00110                     break;
00111
00112                 case GPIO_PORTC:
00113                     SET_BIT(GPIOC->OTYPERx, A_u8PinID);
00114                     break;
00115             }
00116             break;
00117
00118         case GPIOX_PUSH_PULL:
00119             switch(A_u8PortID)
00120             {
00121                 case GPIO_PORTA:
00122                     CLR_BIT(GPIOA->OTYPERx, A_u8PinID);
00123                     break;
00124
00125                 case GPIO_PORTB:
00126                     CLR_BIT(GPIOB->OTYPERx, A_u8PinID);
00127                     break;
00128
00129                 case GPIO_PORTC:
00130                     CLR_BIT(GPIOC->OTYPERx, A_u8PinID);
00131                     break;
00132             }
00133             break;
00134     }
00135
00136 }
00137
00138 /*****
00139 ****/

```

```

00140
00141 FUNC(void) MGPIox_vSetPinOutputSpeed(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
00142     A_u8OutputSpeed)
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
00164 /*****
00165 *****/
00166
00167 FUNC(void) MGPIox_vSetPinInputPullType(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
00168     A_u8InputPullType)
00169 {
00170     switch (A_u8PortID)
00171     {
00172         case GPIO_PORTA:
00173             CLR_BITS(GPIOA->PUPDRx, 0b11, A_u8PinID, 2);
00174             SET_BITS(GPIOA->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00175             break;
00176
00177         case GPIO_PORTB:
00178             CLR_BITS(GPIOB->PUPDRx, 0b11, A_u8PinID, 2);
00179             SET_BITS(GPIOB->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00180             break;
00181
00182         case GPIO_PORTC:
00183             CLR_BITS(GPIOC->PUPDRx, 0b11, A_u8PinID, 2);
00184             SET_BITS(GPIOC->PUPDRx, A_u8InputPullType, A_u8PinID, 2);
00185             break;
00186     }
00187
00188 }
00189
00190 /*****
00191 *****/
00192
00193 FUNC(u8_t) MGPIox_u8GetPinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID)
00194 {
00195
00196     VAR(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     return L_u8PinValue;
00214
00215 }
00216
00217 /*****
00218 *****/
00219
00220 FUNC(void) MGPIox_vSetPinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t) A_u8PinValue)
00221 {
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     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
00262 /*****
00263 *****/
00264
00265 FUNC(void) MGPIOx_vSetResetAtomic(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
A_u8SetResetPinValue)
00266 {
00267
00268     switch (A_u8PortID)
00269     {
00270
00271         case GPIO_PORTA:
00272
00273             switch (A_u8SetResetPinValue)
00274             {
00275                 case GPIOx_HIGH:
00276                     GPIOA->BSRRx = (1 << A_u8PinID);
00277                     break;
00278
00279                 case GPIOx_LOW:
00280                     GPIOA->BSRRx = (1 << (A_u8PinID + 16));
00281                     break;
00282             }
00283
00284             break;
00285
00286         case GPIO_PORTB:
00287
00288             switch (A_u8SetResetPinValue)
00289             {
00290                 case GPIOx_HIGH:
00291                     GPIOB->BSRRx = (1 << A_u8PinID);
00292                     break;
00293
00294                 case GPIOx_LOW:
00295                     GPIOB->BSRRx = (1 << (A_u8PinID + 16));
00296                     break;
00297             }
00298
00299             break;
00300
00301         case GPIO_PORTC:
00302
00303             switch (A_u8SetResetPinValue)
00304             {
00305                 case GPIOx_HIGH:
00306                     GPIOC->BSRRx = (1 << A_u8PinID);
00307                     break;
00308
00309                 case GPIOx_LOW:
00310                     GPIOC->BSRRx = (1 << (A_u8PinID + 16));
00311             }
00312
00313     }
00314
00315 }
```

```

00311             break;
00312         }
00313     }
00314     break;
00315 }
00316
00317 }
00318
00319 /*****
00320 *****/
00321
00322 FUNC(void) MGPIOx_vSetAlternateFunctionON(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID, VAR(u8_t)
A_u8AFID)
00323 {
00324
00325     switch (A_u8PortID)
00326     {
00327         case GPIO_PORTA:
00328             switch (A_u8PinID)
00329             {
00330                 case GPIOx_PIN0:
00331                 case GPIOx_PIN1:
00332                 case GPIOx_PIN2:
00333                 case GPIOx_PIN3:
00334                 case GPIOx_PIN4:
00335                 case GPIOx_PIN5:
00336                 case GPIOx_PIN6:
00337                 case GPIOx_PIN7:
00338                     CLR_BITS(GPIOA->AFRLx, 0b1111, A_u8PinID, 4);
00339                     SET_BITS(GPIOA->AFRLx, A_u8AFID, A_u8PinID, 4);
00340                     break;
00341
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, 4);
00351                     SET_BITS(GPIOA->AFRHx, A_u8AFID, A_u8PinID, 4);
00352                     break;
00353             }
00354             break;
00355
00356         case GPIO_PORTB:
00357             switch (A_u8PinID)
00358             {
00359                 case GPIOx_PIN0:
00360                 case GPIOx_PIN1:
00361                 case GPIOx_PIN2:
00362                 case GPIOx_PIN3:
00363                 case GPIOx_PIN4:
00364                 case GPIOx_PIN5:
00365                 case GPIOx_PIN6:
00366                 case GPIOx_PIN7:
00367                     CLR_BITS(GPIOB->AFRLx, 0b1111, A_u8PinID, 4);
00368                     SET_BITS(GPIOB->AFRLx, A_u8AFID, A_u8PinID, 4);
00369                     break;
00370
00371                 case GPIOx_PIN8:
00372                 case GPIOx_PIN9:
00373                 case GPIOx_PIN10:
00374                 case GPIOx_PIN11:
00375                 case GPIOx_PIN12:
00376                 case GPIOx_PIN13:
00377                 case GPIOx_PIN14:
00378                 case GPIOx_PIN15:
00379                     CLR_BITS(GPIOB->AFRHx, 0b1111, A_u8PinID-8, 4);
00380                     SET_BITS(GPIOB->AFRHx, A_u8AFID, A_u8PinID-8, 4);
00381                     break;
00382             }
00383             break;
00384
00385         case GPIO_PORTC:
00386             switch (A_u8PinID)
00387             {
00388                 case GPIOx_PIN0:
00389                 case GPIOx_PIN1:
00390                 case GPIOx_PIN2:
00391                 case GPIOx_PIN3:
00392                 case GPIOx_PIN4:
00393                 case GPIOx_PIN5:
00394                 case GPIOx_PIN6:
00395                 case GPIOx_PIN7:
00396                     CLR_BITS(GPIOC->AFRLx, 0b1111, A_u8PinID, 4);

```

```

00397             SET_BITS(GPIOC->AFRLx, A_u8AFID, A_u8PinID, 4);
00398             break;
00399
00400         case GPIOx_PIN8:
00401         case GPIOx_PIN9:
00402         case GPIOx_PIN10:
00403         case GPIOx_PIN11:
00404         case GPIOx_PIN12:
00405         case GPIOx_PIN13:
00406         case GPIOx_PIN14:
00407         case GPIOx_PIN15:
00408             CLR_BITS(GPIOC->AFRHx, 0b1111, A_u8PinID, 4);
00409             SET_BITS(GPIOC->AFRHx, A_u8AFID, A_u8PinID, 4);
00410             break;
00411     }
00412     break;
00413 }
00414
00415 }
00416
00417 /*****
00418 *****/
00419
00420 FUNC(void) MGPIOx_vInit(P2VAR(MGPIOx_ConfigType) A_xPinConfig)
00421 {
00422
00423     MGPIOx_vSetPinMode          (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->Mode
00424 );
00425     MGPIOx_vSetPinOutputType    (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputType
00426 );
00427     MGPIOx_vSetPinOutputSpeed   (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->OutputSpeed
00428 );
00429     MGPIOx_vSetPinInputPullType (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->InputType
00430 );
00431     MGPIOx_vSetAlternateFunctionON (A_xPinConfig->Port, A_xPinConfig->Pin, A_xPinConfig->AF_Type
00432 );
00433
00434
00435 FUNC(void) MGPIOx_vTogglePinValue(VAR(u8_t) A_u8PortID, VAR(u8_t) A_u8PinID)
00436 {
00437     switch (A_u8PortID)
00438     {
00439         case GPIO_PORTA:
00440             TOGGLE_BIT(GPIOA->ODRx, A_u8PinID);
00441             break;
00442
00443         case GPIO_PORTB:
00444             TOGGLE_BIT(GPIOB->ODRx, A_u8PinID);
00445             break;
00446
00447         case GPIO_PORTC:
00448             TOGGLE_BIT(GPIOC->ODRx, A_u8PinID);
00449             break;
00450     }
00451
00452 }
00453
00454
00455
00456 FUNC(void) GPIO_vSetValue( VAR(u8_t) A_u8PortID, VAR(u16_t) A_u16PortValue )
00457 {
00458
00459     switch(A_u8PortID)
00460     {
00461
00462         case GPIO_PORTA : GPIOA->ODRx = ( ((GPIOA->ODRx & PORTA_BIT_MANIPULATION) | (A_u16PortValue))
00463             << 1 ) ; break; //Start writing from PIN1 to PIN8.
00464
00465         case GPIO_PORTB : GPIOB->ODRx = ( ((GPIOB->ODRx & PORTB_BIT_MANIPULATION) | (A_u16PortValue))
00466             << 5 ) ; break; //Start writing from PIN5 to PIN12.
00467
00468         case GPIO_PORTC : GPIOC->ODRx ; break; // You shouldn't write on this port.
00469
00470         default : break;
00471     }
00472 }
00473
00474
00475
00476

```

00477
00478

7.82 COTS/MCAL/NVIC/NVIC_config.h File Reference

7.83 NVIC_config.h

[Go to the documentation of this file.](#)

```
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
```

7.84 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)`
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
- `#define NO_GROUP_PRIORITY (0)`
No group priority.
- `#define GROUP_PRIORITY_0 (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)
- `#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.
- #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 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) 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) Steam 2 global interrupt.
• #define DMA2_STREAM3 (59)
• #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 3 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)
• #define FPU (81)
- FPU (Floating Point Unit) global interrupt.
• #define SPI4 (84)
- SPI4 (Serial Peripheral Interface 4) global interrupt.

Functions

- void MNVIC_vEnablePeripheral (VAR(u8_t) A_u8INTID)
Enable the interrupt of a certain peripheral in NVIC.
- void MNVIC_vDisablePeripheral (VAR(u8_t) A_u8INTID)
Disable the interrupt of a certain peripheral in NVIC.
- void MNVIC_vSetPendingFlag (VAR(u8_t) A_u8INTID)
Set the pending flag of the peripheral.
- void MNVIC_vClearPendingFlag (VAR(u8_t) A_u8INTID)
Clear the pending flag of the peripheral.
- u8_t MNVIC_u8GetActive (VAR(u8_t) A_u8INTID)
Get the current state of the interrupt active flag of a certain peripheral in NVIC.
- void MNVIC_vSetPriorityConfig (VAR(u8_t) A_u8PriorityOption)
Configures the group and sub-group priority configuration.
- void MNVIC_vSetPriority (VAR(s8_t) A_s8INTID, VAR(u8_t) A_u8GroupPriority, VAR(u8_t) A_u8SubPriority)
Set the group and the sub-group priorities for the required interrupt.
- u32_t NVIC_GetPriority (VAR(s8_t) A_s8INTID)
Get a certain interrupt's priority.

7.84.1 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](#).

7.84.2 Function Documentation

7.84.2.1 MNVIC_vEnablePeriphral()

```
void MNVIC_vEnablePeriphral (
    VAR(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](#)

7.84.2.2 MNVIC_vDisablePeriphral()

```
void MNVIC_vDisablePeriphral (
    VAR(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](#)

7.84.2.3 MNVIC_vSetPendingFlag()

```
void MNVIC_vSetPendingFlag (
    VAR(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](#)

7.84.2.4 MNVIC_vClearPendingFlag()

```
void MNVIC_vClearPendingFlag (
    VAR(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](#)

7.84.2.5 MNVIC_u8GetActive()

```
u8_t MNVIC_u8GetActive (
    VAR(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

7.84.2.6 MNVIC_vSetPriorityConfig()

```
void MNVIC_vSetPriorityConfig (
    VAR(u8_t) A_u8PriorityOption )
```

Configures the group and sub-group priority configuration.

Parameters

in	<i>A_u8PriorityOption</i>	The peripheral's interrupt ID
----	---------------------------	-------------------------------

7.84.2.7 MNVIC_vSetPriority()

```
void MNVIC_vSetPriority (
    VAR(s8_t) A_s8INTID,
    VAR(u8_t) A_u8GroupPriority,
    VAR(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](#)

7.84.2.8 NVIC_GetPriority()

```
u32_t NVIC_GetPriority (
    VAR(s8_t) A_s8INTID )
```

Get a certain interrupt's priority.

Parameters

in	A_s8INTID	
----	-----------	--

Returns

The priority for the corresponding interrupt

7.85 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(VAR(u8_t) A_u8INTID);
00023
00029 void MNVIC_vDisablePeriphral(VAR(u8_t) A_u8INTID);
00030
00037 void MNVIC_vSetPendingFlag(VAR(u8_t) A_u8INTID);
00038
00045 void MNVIC_vClearPendingFlag(VAR(u8_t) A_u8INTID);
00046
00052 u8_t MNVIC_u8GetActive(VAR(u8_t) A_u8INTID);
00053
00058 void MNVIC_vSetPriorityConfig(VAR(u8_t) A_u8PriorityOption);
00059
00068 void MNVIC_vSetPriority(VAR(s8_t) A_s8INTID, VAR(u8_t) A_u8GroupPriority, VAR(u8_t) A_u8SubPriority);
00069
00075 u32_t NVIC_GetPriority(VAR(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
```

7.86 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)

7.86.1 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](#).

7.86.2 Macro Definition Documentation

7.86.2.1 MNVIC_STIR

```
#define MNVIC_STIR *((volatile 32 *) (0xE000EF00))
```

Definition at line 243 of file [NVIC_private.h](#).

7.86.2.2 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](#).

7.87 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

```

7.88 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_vEnablePeripheral (u8_t A_u8INTID)`
- `void MNVIC_vDisablePeripheral (u8_t A_u8INTID)`
- `void MNVIC_vSetPendingFlag (u8_t A_u8INTID)`
- `void MNVIC_vClearPendingFlag (u8_t A_u8INTID)`
- `u8_t MNVIC_u8GetActive (u8_t A_u8INTID)`
- `void MNVIC_vSetPriorityConfig (u8_t A_u8PriorityOption)`
- `void MNVIC_vSetPriority (s8_t A_s8INTID, u8_t A_u8GroupPriority, u8_t A_u8SubPriority)`
- `u32_t NVIC_GetPriority (s8_t A_s8INTID)`

7.88.1 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](#).

7.88.2 Macro Definition Documentation

7.88.2.1 REGISTER_SIZE

```
#define REGISTER_SIZE (32)
```

Definition at line 27 of file [NVIC_program.c](#).

7.88.3 Function Documentation

7.88.3.1 MNVIC_vEnablePeriphral()

```
void MNVIC_vEnablePeriphral (
    u8_t A_u8INTID )
```

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](#), and [REGISTER_SIZE](#).

7.88.3.2 MNVIC_vDisablePeriphral()

```
void MNVIC_vDisablePeriphral (
    u8_t A_u8INTID )
```

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](#), and [REGISTER_SIZE](#).

7.88.3.3 MNVIC_vSetPendingFlag()

```
void MNVIC_vSetPendingFlag (
    u8_t A_u8INTID )
```

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](#), and [REGISTER_SIZE](#).

7.88.3.4 MNVIC_vClearPendingFlag()

```
void MNVIC_vClearPendingFlag (
    u8_t A_u8INTID )
```

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](#), and [REGISTER_SIZE](#).

7.88.3.5 MNVIC_u8GetActive()

```
u8_t MNVIC_u8GetActive (
    u8_t A_u8INTID )
```

Definition at line 65 of file [NVIC_program.c](#).

```
00066 {
00067     VAR(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](#), [MNVIC](#), [REGISTER_SIZE](#), and [VAR](#).

7.88.3.6 MNVIC_vSetPriorityConfig()

```
void MNVIC_vSetPriorityConfig (
    u8_t A_u8PriorityOption )
```

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](#).

7.88.3.7 MNVIC_vSetPriority()

```
void MNVIC_vSetPriority (
    s8_t A_s8INTID,
    u8_t A_u8GroupPriority,
    u8_t A_u8SubPriority )
```

Definition at line 84 of file [NVIC_program.c](#).

```
00085 {
00086     VAR(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](#), [USAGE_FAULT](#), and [VAR](#).

7.88.3.8 NVIC_GetPriority()

```
u32_t NVIC_GetPriority (
    s8_t A_s8INTID )
```

Definition at line 126 of file [NVIC_program.c](#).

```
00127 {
00128     VAR(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](#), [USAGE_FAULT](#), and [VAR](#).

7.89 NVIC_program.c

[Go to the documentation of this file.](#)

```

00001 /*****
00009 ****/
00010 /*           Include headers           */
00011 ****/
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 #include "NVIC_interface.h"
00018 #include "NVIC_private.h"
00019 #include "NVIC_config.h"
00020
00021 ****/
00022 /*           Global functions           */
00023 ****/
00024
00025 STATIC VAR(u32_t) GS_u32GroupConf = INITIAL_ZERO;
00026
00027 #define REGISTER_SIZE (32)
00028
00029 ****/
00030 /*           Functions' implementations   */
00031 ****/
00032
00033 FUNC(void) MNVIC_vEnablePeriphral(VAR(u8_t) A_u8INTID)
00034 {
00035     MNVIC->ISERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00036 }
00037
00038
00039 ****/
00040
00041 FUNC(void) MNVIC_vDisablePeriphral(VAR(u8_t) A_u8INTID)
00042 {
00043     MNVIC->ICERx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00044 }
00045
00046
00047 ****/
00048
00049 FUNC(void) MNVIC_vSetPendingFlag(VAR(u8_t) A_u8INTID)
00050 {
00051     MNVIC->ISPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00052 }
00053
00054

```

```

00055
00056     /*****
00057     FUNC(void) MNVIC_vClearPendingFlag(VAR(u8_t) A_u8INTID)
00058 {
00059     MNVIC->ICPRx[A_u8INTID / REGISTER_SIZE] = (1 << (A_u8INTID % REGISTER_SIZE));
00060 }
00061
00062     *****/
00063
00064     /*****
00065     FUNC(u8_t) MNVIC_u8GetActive(VAR(u8_t) A_u8INTID)
00066 {
00067     VAR(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 }
00071
00072     *****/
00073
00074     /*****
00075     FUNC(void) MNVIC_vSetPriorityConfig(VAR(u8_t) A_u8PriorityOption)
00076 {
00077     GS_u32GroupConf = (VECTKEY_PASSWORD | (A_u8PriorityOption << 8));
00078     MSCB->AIRCR = GS_u32GroupConf;
00079 }
00080
00081     *****/
00082
00083     /*****
00084     FUNC(void) MNVIC_vSetPriority(VAR(s8_t) A_s8INTID, VAR(u8_t) A_u8GroupPriority, VAR(u8_t)
00085     A_u8SubPriority)
00086 {
00087     VAR(u8_t) L_u8Priority = INITIAL_ZERO;
00088     L_u8Priority = A_u8SubPriority | (A_u8GroupPriority << ((GS_u32GroupConf - 0x05FA0300) / 0x100));
00089
00090     // Core Peripheral
00091     if (A_s8INTID < 0)
00092     {
00093         if ((A_s8INTID == MEMORY_MANAGE) || (A_s8INTID == BUS_FAULT) || (A_s8INTID == USAGE_FAULT))
00094         {
00095             A_s8INTID += 3;
00096             MSCB->SHPR1 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00097         }
00098         else if (A_s8INTID == SV_CALL)
00099         {
00100             A_s8INTID += 7;
00101             MSCB->SHPR2 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00102         }
00103         else if (A_s8INTID == PEND_SV || A_s8INTID == SYSTICK)
00104         {
00105             A_s8INTID += 8;
00106             MSCB->SHPR3 = (L_u8Priority) << ((8 * A_s8INTID) + 4);
00107         }
00108         else
00109         {
00110             /* Do nothing */
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     VAR(u32_t) NVIC_GetPriority(VAR(s8_t) A_s8INTID)
00127 {
00128     VAR(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 }

/***** ****
00167 *****/

```

7.90 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 PPREG2 NoDivision
- #define PPREG1 NoDivision
- #define HPREG 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

7.90.1 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](#).

7.90.2 Macro Definition Documentation

7.90.2.1 PLLI2S

```
#define PLLI2S DISABLE
```

Definition at line 21 of file [MRCC_config.h](#).

7.90.2.2 PLL

```
#define PLL DISABLE
```

Definition at line 29 of file [MRCC_config.h](#).

7.90.2.3 CSS

```
#define CSS DISABLE
```

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

7.90.2.4 HSEBYP

```
#define HSEBYP NOTBYBASED
```

Definition at line 45 of file [MRCC_config.h](#).

7.90.2.5 HSE_EN

```
#define HSE_EN DISABLE
```

Definition at line 53 of file [MRCC_config.h](#).

7.90.2.6 HSI_EN

```
#define HSI_EN ENABLE
```

Definition at line 61 of file [MRCC_config.h](#).

7.90.2.7 PLLQ

```
#define PLLQ Equal_2
```

Definition at line 84 of file [MRCC_config.h](#).

7.90.2.8 PLLSRC

```
#define PLLSRC HSE
```

Definition at line 92 of file [MRCC_config.h](#).

7.90.2.9 PLLP

```
#define PLLP PLLCLK
```

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

7.90.2.10 PLLN

```
#define PLLN Equal_200
```

Definition at line 173 of file [MRCC_config.h](#).

7.90.2.11 PLLM

```
#define PLLM Equal_2
```

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

7.90.2.12 MCO2

```
#define MCO2 SYSCLK
```

Definition at line [253](#) of file [MRCC_config.h](#).

7.90.2.13 MCO2PRE

```
#define MCO2PRE NoDivision
```

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

7.90.2.14 MCO1PRE

```
#define MCO1PRE NoDivision
```

Definition at line [275](#) of file [MRCC_config.h](#).

7.90.2.15 I2SSRC

```
#define I2SSRC PLLI2SCLK
```

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

7.90.2.16 MCO1

```
#define MCO1 HSE
```

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

7.90.2.17 RTCPRE

```
#define RTCPRE HSEby2
```

Definition at line [332](#) of file [MRCC_config.h](#).

7.90.2.18 PPREG

```
#define PPREG NoDivision
```

Definition at line 343 of file [MRCC_config.h](#).

7.90.2.19 PPREF

```
#define PPREF NoDivision
```

Definition at line 354 of file [MRCC_config.h](#).

7.90.2.20 HPREG

```
#define HPREG SYSCLKby2
```

Definition at line 369 of file [MRCC_config.h](#).

7.90.2.21 SWS

```
#define SWS HSI
```

Definition at line 378 of file [MRCC_config.h](#).

7.90.2.22 SW

```
#define SW HSI
```

Definition at line 387 of file [MRCC_config.h](#).

7.90.2.23 DMA2CLK

```
#define DMA2CLK DISABLE
```

Definition at line 398 of file [MRCC_config.h](#).

7.90.2.24 DMA1CLK

```
#define DMA1CLK DISABLE
```

Definition at line 406 of file [MRCC_config.h](#).

7.90.2.25 CRCCLK

```
#define CRCCLK DISABLE
```

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

7.90.2.26 GPIOHCLK

```
#define GPIOHCLK DISABLE
```

Definition at line 422 of file [MRCC_config.h](#).

7.90.2.27 GPIOECLK

```
#define GPIOECLK DISABLE
```

Definition at line 430 of file [MRCC_config.h](#).

7.90.2.28 GPIODCLK

```
#define GPIODCLK DISABLE
```

Definition at line 438 of file [MRCC_config.h](#).

7.90.2.29 GPIOCCLK

```
#define GPIOCCLK DISABLE
```

Definition at line 446 of file [MRCC_config.h](#).

7.90.2.30 GPIOBCLK

```
#define GPIOBCLK DISABLE
```

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

7.90.2.31 GPIOACLK

```
#define GPIOACLK DISABLE
```

Definition at line [462](#) of file [MRCC_config.h](#).

7.90.2.32 OTGFS

```
#define OTGFS DISABLE
```

Definition at line [473](#) of file [MRCC_config.h](#).

7.90.2.33 PWREN

```
#define PWREN DISABLE
```

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

7.90.2.34 I2C3EN

```
#define I2C3EN DISABLE
```

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

7.90.2.35 I2C2EN

```
#define I2C2EN DISABLE
```

Definition at line [500](#) of file [MRCC_config.h](#).

7.90.2.36 I2C1EN

```
#define I2C1EN DISABLE
```

Definition at line 508 of file [MRCC_config.h](#).

7.90.2.37 USART2EN

```
#define USART2EN DISABLE
```

Definition at line 516 of file [MRCC_config.h](#).

7.90.2.38 SPI3EN

```
#define SPI3EN DISABLE
```

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

7.90.2.39 SPI2EN

```
#define SPI2EN DISABLE
```

Definition at line 532 of file [MRCC_config.h](#).

7.90.2.40 WWDGEN

```
#define WWDGEN DISABLE
```

Definition at line 540 of file [MRCC_config.h](#).

7.90.2.41 TIM5EN

```
#define TIM5EN DISABLE
```

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

7.90.2.42 TIM4EN

```
#define TIM4EN DISABLE
```

Definition at line 556 of file [MRCC_config.h](#).

7.90.2.43 TIM3EN

```
#define TIM3EN DISABLE
```

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

7.90.2.44 TIM2EN

```
#define TIM2EN DISABLE
```

Definition at line 572 of file [MRCC_config.h](#).

7.90.2.45 TIM11EN

```
#define TIM11EN DISABLE
```

Definition at line 583 of file [MRCC_config.h](#).

7.90.2.46 TIM10EN

```
#define TIM10EN DISABLE
```

Definition at line 591 of file [MRCC_config.h](#).

7.90.2.47 TIM9EN

```
#define TIM9EN DISABLE
```

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

7.90.2.48 SYSCFGEN

```
#define SYSCFGEN DISABLE
```

Definition at line 607 of file [MRCC_config.h](#).

7.90.2.49 SPI4EN

```
#define SPI4EN DISABLE
```

Definition at line 615 of file [MRCC_config.h](#).

7.90.2.50 SPI1EN

```
#define SPI1EN DISABLE
```

Definition at line 623 of file [MRCC_config.h](#).

7.90.2.51 SDIOEN

```
#define SDIOEN DISABLE
```

Definition at line 631 of file [MRCC_config.h](#).

7.90.2.52 ADC1EN

```
#define ADC1EN DISABLE
```

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

7.90.2.53 USART6EN

```
#define USART6EN DISABLE
```

Definition at line 647 of file [MRCC_config.h](#).

7.90.2.54 USART1EN

```
#define USART1EN DISABLE
```

Definition at line 655 of file [MRCC_config.h](#).

7.90.2.55 TIM1EN

```
#define TIM1EN DISABLE
```

Definition at line 663 of file [MRCC_config.h](#).

7.91 MRCC_config.h

[Go to the documentation of this file.](#)

```
00001
00009 #ifndef MCAL_RCC_MRCC_CONFIG_H_
00010 #define MCAL_RCC_MRCC_CONFIG_H_
00011
00012
00013 /***** RCC_CR configurations *****/
00014 /* Options: *ENABLE *DISABLE */
00015 /***** */
00016
00017 /*options:
00018 *ENABLE
00019 *DISABLE
00020 */
00021 #define PLLI2S DISABLE
00022
00023 /***** */
00024
00025 /*options:
00026 *ENABLE
00027 *DISABLE
00028 */
00029 #define PLL DISABLE
00030
00031 /***** */
00032
00033 /*options:
00034 *ENABLE
00035 *DISABLE
00036 */
00037 #define CSS DISABLE
00038
00039 /***** */
00040
00041 /*options:
00042 *BYBASED
00043 *NOTBYBASED
00044 */
00045 #define HSEBYP NOTBYBASED
00046
00047 /***** */
00048
00049 /*options:
00050 *ENABLE
00051 *DISABLE
00052 */
00053 #define HSE_EN DISABLE
00054
00055 /***** */
00056
00057 /*options:
00058 *ENABLE
00059 *DISABLE
00060 */
00061 #define HSI_EN ENABLE
00062
```

```
00063 /******  
00064 // RCC_PLLCFGR configurations //  
00065 /*****  
00066 /*options:  
00069 *Equal_2  
00070 *Equal_3  
00071 *Equal_4  
00072 *Equal_5  
00073 *Equal_6  
00074 *Equal_7  
00075 *Equal_8  
00076 *Equal_9  
00077 *Equal_10  
00078 *Equal_11  
00079 *Equal_12  
00080 *Equal_13  
00081 *Equal_14  
00082 *Equal_15  
00083 */  
00084 #define PLLQ Equal_2  
00085  
00086 /*****  
00087 /*options:  
00089 *HSI  
00090 *HSE  
00091 */  
00092 #define PLLSRC HSE  
00093  
00094 /*****  
00095 /*options:  
00097 *Equal_2  
00098 *Equal_4  
00099 *Equal_6  
00100 *Equal_8  
00101 */  
00102 #define PLLP PLLCLK  
00103  
00104 /*****  
00105 /*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 /***** // RCC_CFGR configurations *****/
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 *AHBBby8
00341 *AHBby16
00342 */
00343 #define PPREG NoDivision
00344
00345 *****/
00346
00347 /*options:
00348 *NoDivision
00349 *AHBby2
00350 *AHBby4
00351 *AHBby8
00352 *AHBby16
00353 */
00354 #define PPRL 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 SPILEN 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 */
```

7.92 COTS/MCAL/RCC/MRCC_interface.h File Reference

This file contains the interfacing information for the RCC module.

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
- `#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.

Functions

- void MRCC_vInit (void)
Initialize the RCC with a certain configurations.
- void MRCC_vEnablePeriphralCLK (VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeriphralID)
Enabling the CLK on a specific Peripheral.
- void MRCC_vDisablePeriphralCLK (VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeriphralID)
Disabling the CLK on a specific Peripheral.

7.92.1 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](#).

7.92.2 Function Documentation

7.92.2.1 MRCC_vInit()

```
void MRCC_vInit (
    void )
```

Initialize the RCC with a certain configurations.

Definition at line 29 of file [MRCC_program.c](#).

```
00030 {
00031
00032     // PLLI2S (ON/OFF).
00033 #if PLLI2S == ENABLE
00034     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
```

```

00063     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
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->CFGGR, RCC_CFGGR_SW_b0 ) ;
00084     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SW_b1 ) ;
00085
00086 #elif SW == HSE
00087
00088     SET_BIT( RCC->CFGGR, RCC_CFGGR_SW_b0 ) ;
00089     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SW_b1 ) ;
00090
00091 #elif SW == PLLCLK
00092
00093     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SW_b0 ) ;
00094     SET_BIT( RCC->CFGGR, RCC_CFGGR_SW_b1 ) ;
00095
00096 #endif
00097
00098 // Select CLK switch status (HSI/HSE/PLL).
00099 #if SWS == HSI
00100
00101     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b0 ) ;
00102     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b1 ) ;
00103
00104 #elif SWS == HSE
00105
00106     SET_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b0 ) ;
00107     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b1 ) ;
00108
00109 #elif SWS == PLLCLK
00110
00111     CLR_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b0 ) ;
00112     SET_BIT( RCC->CFGGR, RCC_CFGGR_SWS_b1 ) ;
00113
00114 #endif
00115
00116
00117 // MCO2 selection:
00118 #if MCO2 == SYSCLK
00119
00120     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b0 ) ;
00121     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b1 ) ;
00122
00123 #elif MCO2 == PLLI2SCLK
00124
00125     SET_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b0 ) ;
00126     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b1 ) ;
00127
00128 #elif MCO2 == HSE
00129
00130     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b0 ) ;
00131     SET_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b1 ) ;
00132
00133 #elif MCO2 == PLLCLK
00134
00135     SET_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b0 ) ;
00136     SET_BIT( RCC->CFGGR, RCC_CFGGR_MOC2_b1 ) ;
00137
00138 #endif
00139
00140
00141 // MCO2 prescaler:
00142 #if MCO2PRE == NoDivision
00143
00144     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2PRE_b0 ) ;
00145     CLR_BIT( RCC->CFGGR, RCC_CFGGR_MOC2PRE_b1 ) ;
00146     CLR_BIT( RCC->CFGGR, RCC_CFGGR_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 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 }

```

References `CLR_BIT`, `RCC`, `RCC_CFGR_HPREG_b0`, `RCC_CFGR_HPREG_b1`, `RCC_CFGR_HPREG_b2`, `RCC_CFGR_HPREG_b3`, `RCC_CFGR_MOC1_b0`, `RCC_CFGR_MOC1_b1`, `RCC_CFGR_MOC1PRE_b0`, `RCC_CFGR_MOC1PRE_b1`, `RCC_CFGR_MOC1PRE_b2`, `RCC_CFGR_MOC2_b0`, `RCC_CFGR_MOC2_b1`, `RCC_CFGR_MOC2PRE_b0`, `RCC_CFGR_MOC2PRE_b1`, `RCC_CFGR_MOC2PRE_b2`, `RCC_CFGR_PPRE1_b0`, `RCC_CFGR_PPRE1_b1`, `RCC_CFGR_PPRE1_b2`, `RCC_CFGR_PPRE2_b0`, `RCC_CFGR_PPRE2_b1`, `RCC_CFGR_PPRE2_b2`, `RCC_CFGR_SW_b0`, `RCC_CFGR_SW_b1`, `RCC_CFGR_SWS_b0`, `RCC_CFGR_SWS_b1`, `RCC_CR_CSSON`, `RCC_CR_HSEBYP`, `RCC_CR_HSEON`, `RCC_CR_HSION`, `RCC_CR_PLLI2SON`, `RCC_CR_PLLON`, and `SET_BIT`.

7.92.2.2 MRCC_vEnablePeripheralCLK()

```
void MRCC_vEnablePeripheralCLK (
    VAR(u32_t) A_u32BusID,
    VAR(u32_t) A_u32PeripheralID )
```

Enabling the CLK on a specific Peripheral.

Parameters

in	A_u32BusID	Bus ID.
in	A_u32PeripheralID	Peripheral ID.

Referenced by [MADC_vEnable\(\)](#), [MADC_vInit\(\)](#), [MSPI_vInit\(\)](#), and [MUSART_vInit\(\)](#).

7.92.2.3 MRCC_vDisablePeripheralCLK()

```
void MRCC_vDisablePeripheralCLK (
    VAR(u32_t) A_u32BusID,
    VAR(u32_t) A_u32PeripheralID )
```

Disabling the CLK on a specific Peripheral.

Parameters

in	A_u32BusID	Bus ID.
in	A_u32PeripheralID	Peripheral ID.

Referenced by [MADC_vDisable\(\)](#), and [MSPI_vDISABLE\(\)](#).

7.93 MRCC_interface.h

[Go to the documentation of this file.](#)

```
00001
00009 /* Header file guard */
0010 #ifndef MCAL_RCC_MRCC_INTERFACE_H_
0011 #define MCAL_RCC_MRCC_INTERFACE_H_
0012
0013 ****
0014 /* Functions prototypes */
0015 ****
0016
0020 void MRCC_vInit(void);
0021
0027 void MRCC_vEnablePeripheralCLK(VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeripheralID);
0028
0034 void MRCC_vDisablePeripheralCLK(VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeripheralID);
0035
0036 ****
0037 /* Interfacing macros */
0038 ****
0039
0051 #define RCC_AHB1 1
0052
0058 #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_PWRREN 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_ */
```

7.94 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_RTCPPRE_b0 16`
- `#define RCC_CFGR_RTCPPRE_b1 17`

- #define RCC_CFGR_RTCPRE_b2 18
- #define RCC_CFGR_RTCPRE_b3 19
- #define RCC_CFGR_RTCPRE_b4 20
- #define RCC_CFGR_PPRE2_b0 13
- #define RCC_CFGR_PPRE2_b1 14
- #define RCC_CFGR_PPRE2_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 29
- #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

7.94.1 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](#).

7.94.2 Macro Definition Documentation

7.94.2.1 RCC_BASE_ADDRESS

```
#define RCC_BASE_ADDRESS 0x40023800
```

RCC Base Address

Definition at line [211](#) of file [MRCC_private.h](#).

7.94.2.2 RCC

```
#define RCC ( (volatile P2VAR(RCC_MemoryMapType) ) (RCC_BASE_ADDRESS) )
```

RCC register

Definition at line [218](#) of file [MRCC_private.h](#).

7.94.2.3 RCC_CR_PLLI2SRDY

```
#define RCC_CR_PLLI2SRDY 27
```

Definition at line [225](#) of file [MRCC_private.h](#).

7.94.2.4 RCC_CR_PLLI2SON

```
#define RCC_CR_PLLI2SON 26
```

Definition at line [226](#) of file [MRCC_private.h](#).

7.94.2.5 RCC_CR_PLLRDY

```
#define RCC_CR_PLLRDY 25
```

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

7.94.2.6 RCC_CR_PLLON

```
#define RCC_CR_PLLON 24
```

Definition at line [228](#) of file [MRCC_private.h](#).

7.94.2.7 RCC_CR_CSSON

```
#define RCC_CR_CSSON 19
```

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

7.94.2.8 RCC_CR_HSEBYP

```
#define RCC_CR_HSEBYP 18
```

Definition at line 230 of file [MRCC_private.h](#).

7.94.2.9 RCC_CR_HSERDY

```
#define RCC_CR_HSERDY 17
```

Definition at line 231 of file [MRCC_private.h](#).

7.94.2.10 RCC_CR_HSEON

```
#define RCC_CR_HSEON 16
```

Definition at line 232 of file [MRCC_private.h](#).

7.94.2.11 RCC_CR_HSIRDY

```
#define RCC_CR_HSIRDY 1
```

Definition at line 233 of file [MRCC_private.h](#).

7.94.2.12 RCC_CR_HSION

```
#define RCC_CR_HSION 0
```

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

7.94.2.13 RCC_PLLCFGR_PLLQ_b0

```
#define RCC_PLLCFGR_PLLQ_b0 24
```

Definition at line 237 of file [MRCC_private.h](#).

7.94.2.14 RCC_PLLCFGR_PLLQ_b1

```
#define RCC_PLLCFGR_PLLQ_b1 25
```

Definition at line 238 of file [MRCC_private.h](#).

7.94.2.15 RCC_PLLCFGR_PLLQ_b2

```
#define RCC_PLLCFGR_PLLQ_b2 26
```

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

7.94.2.16 RCC_PLLCFGR_PLLQ_b3

```
#define RCC_PLLCFGR_PLLQ_b3 27
```

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

7.94.2.17 RCC_PLLCFGR_PLLSRC

```
#define RCC_PLLCFGR_PLLSRC 22
```

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

7.94.2.18 RCC_PLLCFGR_PLLP_b0

```
#define RCC_PLLCFGR_PLLP_b0 16
```

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

7.94.2.19 RCC_PLLCFGR_PLLP_b1

```
#define RCC_PLLCFGR_PLLP_b1 17
```

Definition at line 243 of file [MRCC_private.h](#).

7.94.2.20 RCC_PLLCFGR_PLLN_b0

```
#define RCC_PLLCFGR_PLLN_b0 6
```

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

7.94.2.21 RCC_PLLCFGR_PLLN_b1

```
#define RCC_PLLCFGR_PLLN_b1 7
```

Definition at line 245 of file [MRCC_private.h](#).

7.94.2.22 RCC_PLLCFGR_PLLN_b2

```
#define RCC_PLLCFGR_PLLN_b2 8
```

Definition at line 246 of file [MRCC_private.h](#).

7.94.2.23 RCC_PLLCFGR_PLLN_b3

```
#define RCC_PLLCFGR_PLLN_b3 9
```

Definition at line 247 of file [MRCC_private.h](#).

7.94.2.24 RCC_PLLCFGR_PLLN_b4

```
#define RCC_PLLCFGR_PLLN_b4 10
```

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

7.94.2.25 RCC_PLLCFGR_PLLN_b5

```
#define RCC_PLLCFGR_PLLN_b5 11
```

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

7.94.2.26 RCC_PLLCFGR_PLLN_b6

```
#define RCC_PLLCFGR_PLLN_b6 12
```

Definition at line 250 of file [MRCC_private.h](#).

7.94.2.27 RCC_PLLCFGR_PLLN_b7

```
#define RCC_PLLCFGR_PLLN_b7 13
```

Definition at line 251 of file [MRCC_private.h](#).

7.94.2.28 RCC_PLLCFGR_PLLN_b8

```
#define RCC_PLLCFGR_PLLN_b8 14
```

Definition at line 252 of file [MRCC_private.h](#).

7.94.2.29 RCC_PLLCFGR_PLLM_b0

```
#define RCC_PLLCFGR_PLLM_b0 0
```

Definition at line 253 of file [MRCC_private.h](#).

7.94.2.30 RCC_PLLCFGR_PLLM_b1

```
#define RCC_PLLCFGR_PLLM_b1 1
```

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

7.94.2.31 RCC_PLLCFGR_PLLM_b2

```
#define RCC_PLLCFGR_PLLM_b2 2
```

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

7.94.2.32 RCC_PLLCFGR_PLLM_b3

```
#define RCC_PLLCFGR_PLLM_b3 3
```

Definition at line 256 of file [MRCC_private.h](#).

7.94.2.33 RCC_PLLCFGR_PLLM_b4

```
#define RCC_PLLCFGR_PLLM_b4 4
```

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

7.94.2.34 RCC_PLLCFGR_PLLM_b5

```
#define RCC_PLLCFGR_PLLM_b5 5
```

Definition at line 258 of file [MRCC_private.h](#).

7.94.2.35 RCC_CFGR_MOC2_b0

```
#define RCC_CFGR_MOC2_b0 30
```

Definition at line 261 of file [MRCC_private.h](#).

7.94.2.36 RCC_CFGR_MOC2_b1

```
#define RCC_CFGR_MOC2_b1 31
```

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

7.94.2.37 RCC_CFGR_MOC2PRE_b0

```
#define RCC_CFGR_MOC2PRE_b0 27
```

Definition at line 263 of file [MRCC_private.h](#).

7.94.2.38 RCC_CFGR_MOC2PRE_b1

```
#define RCC_CFGR_MOC2PRE_b1 28
```

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

7.94.2.39 RCC_CFGR_MOC2PRE_b2

```
#define RCC_CFGR_MOC2PRE_b2 29
```

Definition at line 265 of file [MRCC_private.h](#).

7.94.2.40 RCC_CFGR_MOC1PRE_b0

```
#define RCC_CFGR_MOC1PRE_b0 24
```

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

7.94.2.41 RCC_CFGR_MOC1PRE_b1

```
#define RCC_CFGR_MOC1PRE_b1 25
```

Definition at line 267 of file [MRCC_private.h](#).

7.94.2.42 RCC_CFGR_MOC1PRE_b2

```
#define RCC_CFGR_MOC1PRE_b2 26
```

Definition at line 268 of file [MRCC_private.h](#).

7.94.2.43 RCC_CFGR_I2SSRC

```
#define RCC_CFGR_I2SSRC 23
```

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

7.94.2.44 RCC_CFGR_MOC1_b0

```
#define RCC_CFGR_MOC1_b0 21
```

Definition at line [270](#) of file [MRCC_private.h](#).

7.94.2.45 RCC_CFGR_MOC1_b1

```
#define RCC_CFGR_MOC1_b1 22
```

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

7.94.2.46 RCC_CFGR_RTCPRE_b0

```
#define RCC_CFGR_RTCPRE_b0 16
```

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

7.94.2.47 RCC_CFGR_RTCPRE_b1

```
#define RCC_CFGR_RTCPRE_b1 17
```

Definition at line [273](#) of file [MRCC_private.h](#).

7.94.2.48 RCC_CFGR_RTCPRE_b2

```
#define RCC_CFGR_RTCPRE_b2 18
```

Definition at line [274](#) of file [MRCC_private.h](#).

7.94.2.49 RCC_CFGR_RTCPRE_b3

```
#define RCC_CFGR_RTCPRE_b3 19
```

Definition at line [275](#) of file [MRCC_private.h](#).

7.94.2.50 RCC_CFGR_RTCPRE_b4

```
#define RCC_CFGR_RTCPRE_b4 20
```

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

7.94.2.51 RCC_CFGR_PPRE2_b0

```
#define RCC_CFGR_PPRE2_b0 13
```

Definition at line [277](#) of file [MRCC_private.h](#).

7.94.2.52 RCC_CFGR_PPRE2_b1

```
#define RCC_CFGR_PPRE2_b1 14
```

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

7.94.2.53 RCC_CFGR_PPRE2_b2

```
#define RCC_CFGR_PPRE2_b2 15
```

Definition at line [279](#) of file [MRCC_private.h](#).

7.94.2.54 RCC_CFGR_PPRE1_b0

```
#define RCC_CFGR_PPRE1_b0 10
```

Definition at line [280](#) of file [MRCC_private.h](#).

7.94.2.55 RCC_CFGR_PPRE1_b1

```
#define RCC_CFGR_PPRE1_b1 11
```

Definition at line [281](#) of file [MRCC_private.h](#).

7.94.2.56 RCC_CFGR_PPREG_b2

```
#define RCC_CFGR_PPREG_b2 12
```

Definition at line [282](#) of file [MRCC_private.h](#).

7.94.2.57 RCC_CFGR_HPRE_b0

```
#define RCC_CFGR_HPRE_b0 4
```

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

7.94.2.58 RCC_CFGR_HPRE_b1

```
#define RCC_CFGR_HPRE_b1 5
```

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

7.94.2.59 RCC_CFGR_HPRE_b2

```
#define RCC_CFGR_HPRE_b2 6
```

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

7.94.2.60 RCC_CFGR_HPRE_b3

```
#define RCC_CFGR_HPRE_b3 7
```

Definition at line [286](#) of file [MRCC_private.h](#).

7.94.2.61 RCC_CFGR_SWS_b0

```
#define RCC_CFGR_SWS_b0 2
```

Definition at line [287](#) of file [MRCC_private.h](#).

7.94.2.62 RCC_CFGR_SWS_b1

```
#define RCC_CFGR_SWS_b1 3
```

Definition at line 288 of file [MRCC_private.h](#).

7.94.2.63 RCC_CFGR_SW_b0

```
#define RCC_CFGR_SW_b0 0
```

Definition at line 289 of file [MRCC_private.h](#).

7.94.2.64 RCC_CFGR_SW_b1

```
#define RCC_CFGR_SW_b1 1
```

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

7.94.2.65 RCC_AHB1ENR_DMA2EN

```
#define RCC_AHB1ENR_DMA2EN 22
```

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

7.94.2.66 RCC_AHB1ENR_DMA1EN

```
#define RCC_AHB1ENR_DMA1EN 21
```

Definition at line 294 of file [MRCC_private.h](#).

7.94.2.67 RCC_AHB1ENR_CRCEN

```
#define RCC_AHB1ENR_CRCEN 12
```

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

7.94.2.68 RCC_AHB1ENR_GPIOHEN

```
#define RCC_AHB1ENR_GPIOHEN 7
```

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

7.94.2.69 RCC_AHB1ENR_GPIOEEN

```
#define RCC_AHB1ENR_GPIOEEN 4
```

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

7.94.2.70 RCC_AHB1ENR_GPIODEN

```
#define RCC_AHB1ENR_GPIODEN 3
```

Definition at line [298](#) of file [MRCC_private.h](#).

7.94.2.71 RCC_AHB1ENR_GPIOCEN

```
#define RCC_AHB1ENR_GPIOCEN 2
```

Definition at line [299](#) of file [MRCC_private.h](#).

7.94.2.72 RCC_AHB1ENR_GPIOBEN

```
#define RCC_AHB1ENR_GPIOBEN 1
```

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

7.94.2.73 RCC_AHB1ENR_GPIOAEN

```
#define RCC_AHB1ENR_GPIOAEN 0
```

Definition at line [301](#) of file [MRCC_private.h](#).

7.94.2.74 RCC_AHB2ENR_OTGFSEN

```
#define RCC_AHB2ENR_OTGFSEN 7
```

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

7.94.2.75 RCC_APB1ENR_PWREN

```
#define RCC_APB1ENR_PWREN 28
```

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

7.94.2.76 RCC_APB1ENR_I2C3EN

```
#define RCC_APB1ENR_I2C3EN 23
```

Definition at line 308 of file [MRCC_private.h](#).

7.94.2.77 RCC_APB1ENR_I2C2EN

```
#define RCC_APB1ENR_I2C2EN 22
```

Definition at line 309 of file [MRCC_private.h](#).

7.94.2.78 RCC_APB1ENR_I2C1EN

```
#define RCC_APB1ENR_I2C1EN 21
```

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

7.94.2.79 RCC_APB1ENR_USART2EN

```
#define RCC_APB1ENR_USART2EN 17
```

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

7.94.2.80 RCC_APB1ENR_SPI3EN

```
#define RCC_APB1ENR_SPI3EN 15
```

Definition at line 312 of file [MRCC_private.h](#).

7.94.2.81 RCC_APB1ENR_SPI2EN

```
#define RCC_APB1ENR_SPI2EN 14
```

Definition at line 313 of file [MRCC_private.h](#).

7.94.2.82 RCC_APB1ENR_WWDGEN

```
#define RCC_APB1ENR_WWDGEN 11
```

Definition at line 314 of file [MRCC_private.h](#).

7.94.2.83 RCC_APB1ENR_TIM5EN

```
#define RCC_APB1ENR_TIM5EN 3
```

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

7.94.2.84 RCC_APB1ENR_TIM4EN

```
#define RCC_APB1ENR_TIM4EN 2
```

Definition at line 316 of file [MRCC_private.h](#).

7.94.2.85 RCC_APB1ENR_TIM3EN

```
#define RCC_APB1ENR_TIM3EN 1
```

Definition at line 317 of file [MRCC_private.h](#).

7.94.2.86 RCC_APB1ENR_TIM2EN

```
#define RCC_APB1ENR_TIM2EN 0
```

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

7.94.2.87 RCC_APB2ENR_TIM11EN

```
#define RCC_APB2ENR_TIM11EN 18
```

Definition at line 321 of file [MRCC_private.h](#).

7.94.2.88 RCC_APB2ENR_TIM10EN

```
#define RCC_APB2ENR_TIM10EN 17
```

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

7.94.2.89 RCC_APB2ENR_TIM9EN

```
#define RCC_APB2ENR_TIM9EN 16
```

Definition at line 323 of file [MRCC_private.h](#).

7.94.2.90 RCC_APB2ENR_SYSCFGEN

```
#define RCC_APB2ENR_SYSCFGEN 14
```

Definition at line 324 of file [MRCC_private.h](#).

7.94.2.91 RCC_APB2ENR_SPI4EN

```
#define RCC_APB2ENR_SPI4EN 13
```

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

7.94.2.92 RCC_APB2ENR_SPI1EN

```
#define RCC_APB2ENR_SPI1EN 12
```

Definition at line [326](#) of file [MRCC_private.h](#).

7.94.2.93 RCC_APB2ENR_SDIOEN

```
#define RCC_APB2ENR_SDIOEN 11
```

Definition at line [327](#) of file [MRCC_private.h](#).

7.94.2.94 RCC_APB2ENR_ADC1EN

```
#define RCC_APB2ENR_ADC1EN 8
```

Definition at line [328](#) of file [MRCC_private.h](#).

7.94.2.95 RCC_APB2ENR_USART6EN

```
#define RCC_APB2ENR_USART6EN 5
```

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

7.94.2.96 RCC_APB2ENR_USART1EN

```
#define RCC_APB2ENR_USART1EN 4
```

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

7.94.2.97 RCC_APB2ENR_TIM1EN

```
#define RCC_APB2ENR_TIM1EN 0
```

Definition at line [331](#) of file [MRCC_private.h](#).

7.94.2.98 RCC_AHB1LPENR_FLITFLPEN

```
#define RCC_AHB1LPENR_FLITFLPEN 15
```

Definition at line 333 of file [MRCC_private.h](#).

7.94.2.99 ENABLE

```
#define ENABLE 1
```

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

7.94.2.100 DISABLE

```
#define DISABLE 2
```

Definition at line 341 of file [MRCC_private.h](#).

7.94.2.101 BYBASED

```
#define BYBASED 1
```

Definition at line 343 of file [MRCC_private.h](#).

7.94.2.102 NOTBYBASED

```
#define NOTBYBASED 2
```

Definition at line 344 of file [MRCC_private.h](#).

7.94.2.103 SYSCLK

```
#define SYSCLK 1
```

Definition at line 346 of file [MRCC_private.h](#).

7.94.2.104 PLLI2SCLK

```
#define PLLI2SCLK 5
```

Definition at line [347](#) of file [MRCC_private.h](#).

7.94.2.105 HSE

```
#define HSE 3
```

Definition at line [348](#) of file [MRCC_private.h](#).

7.94.2.106 PLLCLK [1/2]

```
#define PLLCLK 4
```

Definition at line [362](#) of file [MRCC_private.h](#).

7.94.2.107 NoDivision

```
#define NoDivision 1
```

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

7.94.2.108 DivisionBy2

```
#define DivisionBy2 2
```

Definition at line [352](#) of file [MRCC_private.h](#).

7.94.2.109 DivisionBy3

```
#define DivisionBy3 3
```

Definition at line [353](#) of file [MRCC_private.h](#).

7.94.2.110 DivisionBy4

```
#define DivisionBy4 4
```

Definition at line 354 of file [MRCC_private.h](#).

7.94.2.111 DivisionBy5

```
#define DivisionBy5 5
```

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

7.94.2.112 I2S_CKIN

```
#define I2S_CKIN 2
```

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

7.94.2.113 HSI

```
#define HSI 1
```

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

7.94.2.114 LSE

```
#define LSE 2
```

Definition at line 361 of file [MRCC_private.h](#).

7.94.2.115 PLLCLK [2/2]

```
#define PLLCLK 4
```

Definition at line 362 of file [MRCC_private.h](#).

7.94.2.116 SYSCLKby2

```
#define SYSCLKby2 2
```

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

7.94.2.117 SYSCLKby4

```
#define SYSCLKby4 3
```

Definition at line 366 of file [MRCC_private.h](#).

7.94.2.118 SYSCLKby8

```
#define SYSCLKby8 4
```

Definition at line 367 of file [MRCC_private.h](#).

7.94.2.119 SYSCLKby16

```
#define SYSCLKby16 5
```

Definition at line 368 of file [MRCC_private.h](#).

7.94.2.120 SYSCLKby64

```
#define SYSCLKby64 6
```

Definition at line 369 of file [MRCC_private.h](#).

7.94.2.121 SYSCLKby128

```
#define SYSCLKby128 7
```

Definition at line 370 of file [MRCC_private.h](#).

7.94.2.122 SYSCLKby256

```
#define SYSCLKby256 8
```

Definition at line 371 of file [MRCC_private.h](#).

7.94.2.123 SYSCLKby512

```
#define SYSCLKby512 9
```

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

7.94.2.124 NoCLK0

```
#define NoCLK0 1
```

Definition at line 375 of file [MRCC_private.h](#).

7.94.2.125 NoCLK1

```
#define NoCLK1 2
```

Definition at line 376 of file [MRCC_private.h](#).

7.94.2.126 HSEby2

```
#define HSEby2 3
```

Definition at line 377 of file [MRCC_private.h](#).

7.94.2.127 HSEby3

```
#define HSEby3 4
```

Definition at line 378 of file [MRCC_private.h](#).

7.94.2.128 HSEby4

```
#define HSEby4 5
```

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

7.94.2.129 HSEby5

```
#define HSEby5 6
```

Definition at line [380](#) of file [MRCC_private.h](#).

7.94.2.130 HSEby6

```
#define HSEby6 7
```

Definition at line [381](#) of file [MRCC_private.h](#).

7.94.2.131 HSEby7

```
#define HSEby7 8
```

Definition at line [382](#) of file [MRCC_private.h](#).

7.94.2.132 HSEby8

```
#define HSEby8 9
```

Definition at line [383](#) of file [MRCC_private.h](#).

7.94.2.133 HSEby9

```
#define HSEby9 10
```

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

7.94.2.134 HSEby10

```
#define HSEby10 11
```

Definition at line 385 of file [MRCC_private.h](#).

7.94.2.135 HSEby11

```
#define HSEby11 12
```

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

7.94.2.136 HSEby12

```
#define HSEby12 13
```

Definition at line 387 of file [MRCC_private.h](#).

7.94.2.137 HSEby13

```
#define HSEby13 14
```

Definition at line 388 of file [MRCC_private.h](#).

7.94.2.138 HSEby14

```
#define HSEby14 15
```

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

7.94.2.139 HSEby15

```
#define HSEby15 16
```

Definition at line 390 of file [MRCC_private.h](#).

7.94.2.140 HSEby16

```
#define HSEby16 17
```

Definition at line 391 of file [MRCC_private.h](#).

7.94.2.141 HSEby17

```
#define HSEby17 18
```

Definition at line 392 of file [MRCC_private.h](#).

7.94.2.142 HSEby18

```
#define HSEby18 19
```

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

7.94.2.143 HSEby19

```
#define HSEby19 20
```

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

7.94.2.144 HSEby20

```
#define HSEby20 21
```

Definition at line 395 of file [MRCC_private.h](#).

7.94.2.145 HSEby21

```
#define HSEby21 22
```

Definition at line 396 of file [MRCC_private.h](#).

7.94.2.146 HSEby22

```
#define HSEby22 23
```

Definition at line 397 of file [MRCC_private.h](#).

7.94.2.147 HSEby23

```
#define HSEby23 24
```

Definition at line 398 of file [MRCC_private.h](#).

7.94.2.148 HSEby24

```
#define HSEby24 25
```

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

7.94.2.149 HSEby25

```
#define HSEby25 26
```

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

7.94.2.150 HSEby26

```
#define HSEby26 27
```

Definition at line 401 of file [MRCC_private.h](#).

7.94.2.151 HSEby27

```
#define HSEby27 28
```

Definition at line 402 of file [MRCC_private.h](#).

7.94.2.152 HSEby28

```
#define HSEby28 39
```

Definition at line [403](#) of file [MRCC_private.h](#).

7.94.2.153 HSEby29

```
#define HSEby29 30
```

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

7.94.2.154 HSEby30

```
#define HSEby30 31
```

Definition at line [405](#) of file [MRCC_private.h](#).

7.94.2.155 HSEby31

```
#define HSEby31 32
```

Definition at line [406](#) of file [MRCC_private.h](#).

7.94.2.156 AHBby2

```
#define AHBby2 1
```

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

7.94.2.157 AHBby4

```
#define AHBby4 2
```

Definition at line [410](#) of file [MRCC_private.h](#).

7.94.2.158 AHBby8

```
#define AHBby8 3
```

Definition at line 411 of file [MRCC_private.h](#).

7.94.2.159 AHBby16

```
#define AHBby16 4
```

Definition at line 412 of file [MRCC_private.h](#).

7.94.2.160 Equal_0

```
#define Equal_0 0
```

Definition at line 415 of file [MRCC_private.h](#).

7.94.2.161 Equal_1

```
#define Equal_1 1
```

Definition at line 416 of file [MRCC_private.h](#).

7.94.2.162 Equal_2

```
#define Equal_2 2
```

Definition at line 417 of file [MRCC_private.h](#).

7.94.2.163 Equal_3

```
#define Equal_3 3
```

Definition at line 418 of file [MRCC_private.h](#).

7.94.2.164 Equal_4

```
#define Equal_4 4
```

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

7.94.2.165 Equal_5

```
#define Equal_5 5
```

Definition at line [420](#) of file [MRCC_private.h](#).

7.94.2.166 Equal_6

```
#define Equal_6 6
```

Definition at line [421](#) of file [MRCC_private.h](#).

7.94.2.167 Equal_7

```
#define Equal_7 7
```

Definition at line [422](#) of file [MRCC_private.h](#).

7.94.2.168 Equal_8

```
#define Equal_8 8
```

Definition at line [423](#) of file [MRCC_private.h](#).

7.94.2.169 Equal_9

```
#define Equal_9 9
```

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

7.94.2.170 Equal_10

```
#define Equal_10 10
```

Definition at line [425](#) of file [MRCC_private.h](#).

7.94.2.171 Equal_11

```
#define Equal_11 11
```

Definition at line [426](#) of file [MRCC_private.h](#).

7.94.2.172 Equal_12

```
#define Equal_12 12
```

Definition at line [427](#) of file [MRCC_private.h](#).

7.94.2.173 Equal_13

```
#define Equal_13 13
```

Definition at line [428](#) of file [MRCC_private.h](#).

7.94.2.174 Equal_14

```
#define Equal_14 14
```

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

7.94.2.175 Equal_15

```
#define Equal_15 15
```

Definition at line [430](#) of file [MRCC_private.h](#).

7.94.2.176 Equal_16

```
#define Equal_16 16
```

Definition at line [431](#) of file [MRCC_private.h](#).

7.94.2.177 Equal_17

```
#define Equal_17 17
```

Definition at line [432](#) of file [MRCC_private.h](#).

7.94.2.178 Equal_18

```
#define Equal_18 18
```

Definition at line [433](#) of file [MRCC_private.h](#).

7.94.2.179 Equal_19

```
#define Equal_19 19
```

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

7.94.2.180 Equal_20

```
#define Equal_20 20
```

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

7.94.2.181 Equal_21

```
#define Equal_21 21
```

Definition at line [436](#) of file [MRCC_private.h](#).

7.94.2.182 Equal_22

```
#define Equal_22 22
```

Definition at line [437](#) of file [MRCC_private.h](#).

7.94.2.183 Equal_23

```
#define Equal_23 23
```

Definition at line [438](#) of file [MRCC_private.h](#).

7.94.2.184 Equal_24

```
#define Equal_24 24
```

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

7.94.2.185 Equal_25

```
#define Equal_25 25
```

Definition at line [440](#) of file [MRCC_private.h](#).

7.94.2.186 Equal_26

```
#define Equal_26 26
```

Definition at line [441](#) of file [MRCC_private.h](#).

7.94.2.187 Equal_27

```
#define Equal_27 27
```

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

7.94.2.188 Equal_28

```
#define Equal_28 28
```

Definition at line [443](#) of file [MRCC_private.h](#).

7.94.2.189 Equal_29

```
#define Equal_29 29
```

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

7.94.2.190 Equal_30

```
#define Equal_30 30
```

Definition at line [445](#) of file [MRCC_private.h](#).

7.94.2.191 Equal_31

```
#define Equal_31 31
```

Definition at line [446](#) of file [MRCC_private.h](#).

7.94.2.192 Equal_32

```
#define Equal_32 32
```

Definition at line [447](#) of file [MRCC_private.h](#).

7.94.2.193 Equal_33

```
#define Equal_33 33
```

Definition at line [448](#) of file [MRCC_private.h](#).

7.94.2.194 Equal_34

```
#define Equal_34 34
```

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

7.94.2.195 Equal_35

```
#define Equal_35 35
```

Definition at line [450](#) of file [MRCC_private.h](#).

7.94.2.196 Equal_36

```
#define Equal_36 36
```

Definition at line [451](#) of file [MRCC_private.h](#).

7.94.2.197 Equal_37

```
#define Equal_37 37
```

Definition at line [452](#) of file [MRCC_private.h](#).

7.94.2.198 Equal_38

```
#define Equal_38 38
```

Definition at line [453](#) of file [MRCC_private.h](#).

7.94.2.199 Equal_39

```
#define Equal_39 39
```

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

7.94.2.200 Equal_40

```
#define Equal_40 40
```

Definition at line [455](#) of file [MRCC_private.h](#).

7.94.2.201 Equal_41

```
#define Equal_41 41
```

Definition at line [456](#) of file [MRCC_private.h](#).

7.94.2.202 Equal_42

```
#define Equal_42 42
```

Definition at line [457](#) of file [MRCC_private.h](#).

7.94.2.203 Equal_43

```
#define Equal_43 43
```

Definition at line [458](#) of file [MRCC_private.h](#).

7.94.2.204 Equal_44

```
#define Equal_44 44
```

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

7.94.2.205 Equal_45

```
#define Equal_45 45
```

Definition at line [460](#) of file [MRCC_private.h](#).

7.94.2.206 Equal_46

```
#define Equal_46 46
```

Definition at line [461](#) of file [MRCC_private.h](#).

7.94.2.207 Equal_47

```
#define Equal_47 47
```

Definition at line [462](#) of file [MRCC_private.h](#).

7.94.2.208 Equal_48

```
#define Equal_48 48
```

Definition at line [463](#) of file [MRCC_private.h](#).

7.94.2.209 Equal_49

```
#define Equal_49 49
```

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

7.94.2.210 Equal_50

```
#define Equal_50 50
```

Definition at line [465](#) of file [MRCC_private.h](#).

7.94.2.211 Equal_51

```
#define Equal_51 51
```

Definition at line [466](#) of file [MRCC_private.h](#).

7.94.2.212 Equal_52

```
#define Equal_52 52
```

Definition at line [467](#) of file [MRCC_private.h](#).

7.94.2.213 Equal_53

```
#define Equal_53 53
```

Definition at line [468](#) of file [MRCC_private.h](#).

7.94.2.214 Equal_54

```
#define Equal_54 54
```

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

7.94.2.215 Equal_55

```
#define Equal_55 55
```

Definition at line [470](#) of file [MRCC_private.h](#).

7.94.2.216 Equal_56

```
#define Equal_56 56
```

Definition at line [471](#) of file [MRCC_private.h](#).

7.94.2.217 Equal_57

```
#define Equal_57 57
```

Definition at line [472](#) of file [MRCC_private.h](#).

7.94.2.218 Equal_58

```
#define Equal_58 58
```

Definition at line [473](#) of file [MRCC_private.h](#).

7.94.2.219 Equal_59

```
#define Equal_59 59
```

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

7.94.2.220 Equal_60

```
#define Equal_60 60
```

Definition at line [475](#) of file [MRCC_private.h](#).

7.94.2.221 Equal_61

```
#define Equal_61 61
```

Definition at line [476](#) of file [MRCC_private.h](#).

7.94.2.222 Equal_62

```
#define Equal_62 62
```

Definition at line [477](#) of file [MRCC_private.h](#).

7.94.2.223 Equal_63

```
#define Equal_63 63
```

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

7.95 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 */

```

7.96 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_private.h"
#include "MRCC_interface.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)
- void [MRCC_vDisablePeriphralCLK](#) ([u32_t](#) A_u32BusID, [u32_t](#) A_u32PeriphralID)

7.96.1 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](#).

7.96.2 Function Documentation

7.96.2.1 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, RCC_CFGR_HPRE_b0, RCC_CFGR_HPRE_b1, RCC_CFGR_HPRE_b2, RCC_CFGR_HPRE_b3, RCC_CFGR_MOC1_b0, RCC_CFGR_MOC1_b1, RCC_CFGR_MOC1PRE_b0, RCC_CFGR_MOC1PRE_b1, RCC_CFGR_MOC1PRE_b2, RCC_CFGR_MOC2_b0, RCC_CFGR_MOC2_b1, RCC_CFGR_MOC2PRE_b0, RCC_CFGR_MOC2PRE_b1, RCC_CFGR_MOC2PRE_b2, RCC_CFGR_PPREG1_b0, RCC_CFGR_PPREG1_b1, RCC_CFGR_PPREG1_b2, RCC_CFGR_PPREG2_b0, RCC_CFGR_PPREG2_b1, RCC_CFGR_PPREG2_b2, RCC_CFGR_SW_b0, RCC_CFGR_SW_b1, RCC_CFGR_SWS_b0, RCC_CFGR_SWS_b1, RCC_CR_CSSON, RCC_CR_HSEBYP, RCC_CR_HSEON, RCC_CR_HSION, RCC_CR_PLLI2SON, RCC_CR_PLLON, and SET_BIT.

7.96.2.2 MRCC_vEnablePeriphralCLK()

```

void MRCC_vEnablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )

```

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

```

```

00418     break ;
00419
00420
00421     case RCC_APB1 :
00422         SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00423
00424     break ;
00425
00426
00427     case RCC_APB2 :
00428         SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00429
00430     break ;
00431
00432     case RCC_AHB1LPENR:
00433
00434         SET_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00435
00436     break ;
00437
00438     default:
00439
00440         // Error wrong Bus ID
00441
00442     break ;
00443
00444 }
00445
00446 }
00447
00448 }
```

References [RCC](#), [RCC_AHB1](#), [RCC_AHB1LPENR](#), [RCC_AHB2](#), [RCC_APB1](#), [RCC_APB2](#), and [SET_BIT](#).

7.96.2.3 MRCC_vDisablePeriphralCLK()

```

void MRCC_vDisablePeriphralCLK (
    u32_t A_u32BusID,
    u32_t A_u32PeriphralID )
```

Definition at line 453 of file [MRCC_program.c](#).

```

00454 {
00455
00456     switch( A_u32BusID )
00457     {
00458
00459         case RCC_AHB1 :
00460             CLR_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00461
00462         break ;
00463
00464
00465         case RCC_AHB2 :
00466
00467             CLR_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00468
00469
00470         break ;
00471
00472
00473         case RCC_APB1 :
00474
00475             CLR_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00476
00477         break ;
00478
00479
00480         case RCC_APB2 :
00481
00482             CLR_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00483
00484         break ;
00485
00486
00487         case RCC_AHB1LPENR:
```

```

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](#).

7.97 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_private.h"
00020 #include "MRCC_interface.h"
00021 #include "MRCC_config.h"
00022
00023
00024
00025 /*****
00026 /*           Functions implementations      */
00027 *****/
00028
00029 FUNC(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     CLR_BIT( RCC->CR, RCC_CR_HSEBYP ) ;
00075
00076 #endif
00077
00078
00079 // Select CLK switch (HSI/HSE/PLL).
00080 #if SW == HSI
00081
00082     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00083     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00084
00085 #elif SW == HSE
00086
00087     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00088     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00089
00090 #elif SW == PLLCLK
00091
00092     CLR_BIT( RCC->CFGR, RCC_CFGR_SW_b0 ) ;
00093     SET_BIT( RCC->CFGR, RCC_CFGR_SW_b1 ) ;
00094
00095 #endif
00096
00097 // Select CLK switch status (HSI/HSE/PLL).
00098 #if SWS == HSI
00099
00100     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00101     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00102
00103 #elif SWS == HSE
00104
00105     SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00106     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00107
00108 #elif SWS == PLLCLK
00109
00110     CLR_BIT( RCC->CFGR, RCC_CFGR_SWS_b0 ) ;
00111     SET_BIT( RCC->CFGR, RCC_CFGR_SWS_b1 ) ;
00112
00113 #endif
00114
00115
00116 // MCO2 selection:
00117 #if MCO2 == SYSCLK
00118
00119     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00120     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00121
00122 #elif MCO2 == PLLI2SCLK
00123
00124     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00125     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00126
00127 #elif MCO2 == HSE
00128
00129     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00130     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00131
00132 #elif MCO2 == PLLCLK
00133
00134     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b0 ) ;
00135     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2_b1 ) ;
00136
00137 #endif
00138
00139
00140 // MCO2 prescaler:
00141 #if MCO2PRE == NoDivision
00142
00143     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00144     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00145     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00146
00147 #elif MCO2PRE == DivisionBy2
00148
00149     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00150     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00151     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00152
00153 #elif MCO2PRE == DivisionBy3
00154
00155     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     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00162     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00163     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00164
00165 #elif MCO2PRE == DivisionBy5
00166     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b0 ) ;
00167     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b1 ) ;
00168     SET_BIT( RCC->CFGR, RCC_CFGR_MOC2PRE_b2 ) ;
00169
00170 #endif
00171
00172 // MCO1 selection:
00173
00174 #if MCO1 == HSI
00175     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00176     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00177
00178 #elif MCO1 == LSE
00179     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00180     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00181
00182 #elif MCO1 == HSE
00183     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00184     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00185
00186 #elif MCO1 == PLLCLK
00187     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b0 ) ;
00188     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1_b1 ) ;
00189
00190 #endif
00191
00192 // MCO1 prescaler:
00193 #if MCO1PRE == NoDivision
00194     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00195     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00196     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00197
00198 #elif MCO1PRE == DivisionBy2
00199     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00200     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00201     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00202
00203 #elif MCO1PRE == DivisionBy3
00204     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00205     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00206     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00207
00208 #elif MCO1PRE == DivisionBy4
00209     CLR_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00210     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00211     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00212
00213 #elif MCO1PRE == DivisionBy5
00214     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b0 ) ;
00215     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b1 ) ;
00216     SET_BIT( RCC->CFGR, RCC_CFGR_MOC1PRE_b2 ) ;
00217
00218 #endif
00219
00220 // AHB prescalers:
00221 #if HPRE == NoDivision
00222     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00223     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00224     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00225
00226 #elif HPRE == DivisionBy2
00227     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00228     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b1 ) ;
00229     SET_BIT( RCC->CFGR, RCC_CFGR_HPRE_b2 ) ;
00230
00231 #endif
00232
00233 // SYSCLKby2
00234 #if HPRE == SYSCLKby2
00235     CLR_BIT( RCC->CFGR, RCC_CFGR_HPRE_b0 ) ;
00236
00237 #endif

```

```

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 PPRE1 == NoDivision
00302
00303     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b0 ) ;
00304     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b1 ) ;
00305     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b2 ) ;
00306
00307 #elif PPREG1 == AHBy2
00308
00309     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b0 ) ;
00310     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b1 ) ;
00311     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b2 ) ;
00312
00313 #elif PPREG1 == AHBy4
00314
00315     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b0 ) ;
00316     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b1 ) ;
00317     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b2 ) ;
00318
00319 #elif PPREG1 == AHBy8
00320
00321     CLR_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b0 ) ;
00322     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b1 ) ;
00323     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b2 ) ;
00324
00325 #elif PPREG1 == AHBy16
00326
00327     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b0 ) ;
00328     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_b1 ) ;
00329     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG1_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 PPREG == AHBby16
00359
00360     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b0 ) ;
00361     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b1 ) ;
00362     SET_BIT( RCC->CFGR, RCC_CFGR_PPREG_b2 ) ;
00363
00364 #endif
00365
00366 // PLL configurations:
00367
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 FUNC(void) MRCC_vEnablePeripheralCLK( VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeripheralID )
00401 {
00402
00403     switch( A_u32BusID )
00404     {
00405
00406         case RCC_AHB1 :
00407
00408             SET_BIT( RCC->AHB1ENR, A_u32PeripheralID ) ;
00409
00410         break ;
00411
00412
00413         case RCC_AHB2 :
00414
00415             SET_BIT( RCC->AHB2ENR, A_u32PeripheralID ) ;
00416
00417

```

```
00418     break ;
00419
00420     case RCC_APB1 :
00421         SET_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00422         break ;
00423
00424     case RCC_APB2 :
00425         SET_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00426         break ;
00427
00428     case RCC_AHB1LPENR:
00429         SET_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00430         break ;
00431
00432     default:
00433         // Error wrong Bus ID
00434         break ;
00435     }
00436 }
```

```
00450 /*****
00451 *****/
00452
00453 FUNC(void) MRCC_vDisablePeriphralCLK( VAR(u32_t) A_u32BusID, VAR(u32_t) A_u32PeriphralID )
00454 {
00455     switch( A_u32BusID )
00456     {
00457
00458         case RCC_AHB1 :
00459             CLR_BIT( RCC->AHB1ENR, A_u32PeriphralID ) ;
00460             break ;
00461
00462         case RCC_AHB2 :
00463             CLR_BIT( RCC->AHB2ENR, A_u32PeriphralID ) ;
00464             break ;
00465
00466         case RCC_APB1 :
00467             CLR_BIT( RCC->APB1ENR, A_u32PeriphralID ) ;
00468             break ;
00469
00470         case RCC_APB2 :
00471             CLR_BIT( RCC->APB2ENR, A_u32PeriphralID ) ;
00472             break ;
00473
00474         case RCC_AHB1LPENR:
00475             CLR_BIT( RCC->AHB1LPENR, A_u32PeriphralID ) ;
00476             break ;
00477
00478         default:
00479             // Error wrong Bus ID
00480             break ;
00481     }
00482 }
```

```
00505 /******  
00506 *****/  
00507  
00508  
00509  
00510  
00511  
00512  
00513  
00514  
00515
```

7.98 COTS/MCAL/SPI/SPI_config.h File Reference

Macros

- #define CPHA_MODE FIRST_CLK_Captured
- #define CPOL_MODE CLK_IdleAt0
- #define MSTR_MODE MASTER
- #define BR_MODE CLK_By8
- #define SPE_MODE ENABLE
- #define LSBFIRST_MODE MSB_First
- #define SSM_MODE DISABLE
- #define RXONLY_MODE FULL_DUPLEX
- #define DFF_MODE EIGHT_BITS
- #define CRCNEXT_MODE NO_CRC_PHASE
- #define CRCEN_MODE DISABLE
- #define BIDIOE_MODE DISABLE
- #define BIDMODE_MODE OneLine
- #define HALF_DUPLEX_MODE HALF_DUPLEX_TX
- #define SIMPLEX_MODE SIMPLEX_TX
- #define RXDMAEN_MODE DISABLE
- #define TXDMAEN_MODE DISABLE
- #define MULTI_MSTR_MODE DISABLE
- #define FRF_MODE
- #define ERRIE_MODE DISABLE
- #define RXNEIE_MODE DISABLE
- #define TXNEIE_MODE DISABLE
- #define FRE_MODE DISABLE
- #define BSY_MODE DISABLE
- #define OVR_MODE DISABLE
- #define MODF_MODE DISABLE
- #define UDR_MODE
- #define CRCERR_MODE DISABLE
- #define RXNEIE_MODE DISABLE
- #define TXNEIE_MODE DISABLE

7.98.1 Macro Definition Documentation

7.98.1.1 CPHA_MODE

```
#define CPHA_MODE FIRST_CLK_Captured
```

Definition at line 18 of file [SPI_config.h](#).

7.98.1.2 CPOL_MODE

```
#define CPOL_MODE CLK_IdleAt0
```

Definition at line 26 of file [SPI_config.h](#).

7.98.1.3 MSTR_MODE

```
#define MSTR_MODE MASTER
```

Definition at line 34 of file [SPI_config.h](#).

7.98.1.4 BR_MODE

```
#define BR_MODE CLK_By8
```

Definition at line 48 of file [SPI_config.h](#).

7.98.1.5 SPE_MODE

```
#define SPE_MODE ENABLE
```

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

7.98.1.6 LSBFIRST_MODE

```
#define LSBFIRST_MODE MSB_First
```

Definition at line 64 of file [SPI_config.h](#).

7.98.1.7 SSM_MODE

```
#define SSM_MODE DISABLE
```

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

7.98.1.8 RXONLY_MODE

```
#define RXONLY_MODE FULL_DUPLEX
```

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

7.98.1.9 DFF_MODE

```
#define DFF_MODE EIGHT_BITS
```

Definition at line 88 of file [SPI_config.h](#).

7.98.1.10 CRCNEXT_MODE

```
#define CRCNEXT_MODE NO_CRC_PHASE
```

Definition at line 96 of file [SPI_config.h](#).

7.98.1.11 CRCEN_MODE

```
#define CRCEN_MODE DISABLE
```

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

7.98.1.12 BIDIOE_MODE

```
#define BIDIOE_MODE DISABLE
```

Definition at line 112 of file [SPI_config.h](#).

7.98.1.13 BIDMODE_MODE

```
#define BIDMODE_MODE OneLine
```

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

7.98.1.14 HALF_DUPLEX_MODE

```
#define HALF_DUPLEX_MODE HALF_DUPLEX_TX
```

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

7.98.1.15 SIMPLEX_MODE

```
#define SIMPLEX_MODE SIMPLEX_TX
```

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

7.98.1.16 RXDMAEN_MODE

```
#define RXDMAEN_MODE DISABLE
```

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

7.98.1.17 TXDMAEN_MODE

```
#define TXDMAEN_MODE DISABLE
```

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

7.98.1.18 MULTI_MSTR_MODE

```
#define MULTI_MSTR_MODE DISABLE
```

Definition at line 162 of file [SPI_config.h](#).

7.98.1.19 FRF_MODE

```
#define FRF_MODE
```

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

7.98.1.20 ERRIE_MODE

```
#define ERRIE_MODE DISABLE
```

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

7.98.1.21 RXNEIE_MODE [1/2]

```
#define RXNEIE_MODE DISABLE
```

Definition at line 253 of file [SPI_config.h](#).

7.98.1.22 TXNEIE_MODE [1/2]

```
#define TXNEIE_MODE DISABLE
```

Definition at line 261 of file [SPI_config.h](#).

7.98.1.23 FRE_MODE

```
#define FRE_MODE DISABLE
```

Definition at line 205 of file [SPI_config.h](#).

7.98.1.24 BSY_MODE

```
#define BSY_MODE DISABLE
```

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

7.98.1.25 OVR_MODE

```
#define OVR_MODE DISABLE
```

Definition at line 221 of file [SPI_config.h](#).

7.98.1.26 MODF_MODE

```
#define MODF_MODE DISABLE
```

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

7.98.1.27 UDR_MODE

```
#define UDR_MODE
```

Definition at line 237 of file [SPI_config.h](#).

7.98.1.28 CRCERR_MODE

```
#define CRCERR_MODE DISABLE
```

Definition at line 245 of file [SPI_config.h](#).

7.98.1.29 RXNEIE_MODE [2/2]

```
#define RXNEIE_MODE DISABLE
```

Definition at line 253 of file [SPI_config.h](#).

7.98.1.30 TXNEIE_MODE [2/2]

```
#define TXNEIE_MODE DISABLE
```

Definition at line 261 of file [SPI_config.h](#).

7.99 SPI_config.h

[Go to the documentation of this file.](#)

```
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 /***** // SPI_CRL configurations *****/
00011 // SPI_CRL configurations //
00012 /***** *****/
00013
00014 /*options:
00015 *FIRST_CLK_Captured
00016 *SECOND_CLK_Captured
00017 */
00018 #define CPHA_MODE FIRST_CLK_Captured
00019
00020 /***** *****/
00021
00022 /*options:
00023 *CLK_IdleAt0
00024 *CLK_IdleAt1
00025 */
00026 #define CPOL_MODE CLK_IdleAt0
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_By8
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 /***** */
00139 // SPI_CR2 configurations //
00140 /***** */
00141
00142 /*options:
00143 *ENABLE
00144 *DISABLE
00145 */
00146 #define RXDMAEN_MODE DISABLE
00147
00148 /***** */
00149
00150 /*options:
00151 *ENABLE
00152 *DISABLE
00153 */
00154 #define TXDMAEN_MODE DISABLE
00155
00156 /***** */
00157
00158 /*options:
00159 *ENABLE
00160 *DISABLE
00161 */
00162 #define MULTI_MSTR_MODE DISABLE
00163
00164 /***** */
00165
00166 /*options:
00167 *MOTOROLA
00168 *TI
00169 */
```

```
00170 #define FRF_MODE
00171
00172 /******
00173
00174 /*options:
00175 *ENABLE
00176 *DISABLE
00177 */
00178 #define ERRIE_MODE DISABLE
00179
00180 //*****
00181
00182 /*options:
00183 *ENABLE
00184 *DISABLE
00185 */
00186 #define RXNEIE_MODE DISABLE
00187
00188 //*****
00189
00190 /*options:
00191 *ENABLE
00192 *DISABLE
00193 */
00194 #define TXNEIE_MODE DISABLE
00195
00196
00197 //***** // SPI_SR configurations //
00198
00199 //*****
00200
00201 /*options:
00202 *ERROR
00203 *NO_ERROR
00204 */
00205 #define FRE_MODE DISABLE
00206
00207 //*****
00208
00209 /*options:
00210 *BUSY
00211 *NOT_BUSY
00212 */
00213 #define BSY_MODE DISABLE
00214
00215 //*****
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

```

7.100 COTS/MCAL/SPI/SPI_interface.h File Reference

Data Structures

- struct [SPI_MemoryMapType](#)

Macros

- `#define SPI1_BASE_ADDRESS 0x40013000`
- `#define SPI2_BASE_ADDRESS 0x40003800`
- `#define SPI3_BASE_ADDRESS 0x40003C00`
- `#define SPI4_BASE_ADDRESS 0x40013400`
- `#define SPI1 ((SPI_MemoryMapType*) SPI1_BASE_ADDRESS)`
- `#define SPI2 ((SPI_MemoryMapType*) SPI2_BASE_ADDRESS)`
- `#define SPI3 ((SPI_MemoryMapType*) SPI3_BASE_ADDRESS)`
- `#define SPIx_MSTR 1`
- `#define SPIx_SLAVE 2`
- `#define SPIx_SIMPLEX 1`
- `#define SPIx_HALF_DUPLEX 2`
- `#define SPIx_FULL_DUPLEX 3`

Functions

- `void MSPI_vInit (SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection)`
- `u16_t MSPI_u16Transcieve (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)`
- `void MSPI_vMasterTransmit (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)`
- `u16_t MSPI_u16MasterRecieve (SPI_MemoryMapType *A_SPIx)`
- `void MSPI_vSlaveTransmit (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)`
- `u16_t MSPI_u16SlaveRecieve (SPI_MemoryMapType *A_SPIx)`
- `void MSPI_vDISABLE (SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection)`
- `void MSPI1_vSetCallBack (void(*Fptr)(void))`
- `void MSPI2_vSetCallBack (void(*Fptr)(void))`
- `void MSPI3_vSetCallBack (void(*Fptr)(void))`

7.100.1 Macro Definition Documentation

7.100.1.1 SPI1_BASE_ADDRESS

```
#define SPI1_BASE_ADDRESS 0x40013000
```

Definition at line 13 of file [SPI_interface.h](#).

7.100.1.2 SPI2_BASE_ADDRESS

```
#define SPI2_BASE_ADDRESS 0x40003800
```

Definition at line 14 of file [SPI_interface.h](#).

7.100.1.3 SPI3_BASE_ADDRESS

```
#define SPI3_BASE_ADDRESS 0x40003C00
```

Definition at line 15 of file [SPI_interface.h](#).

7.100.1.4 SPI4_BASE_ADDRESS

```
#define SPI4_BASE_ADDRESS 0x40013400
```

Definition at line 16 of file [SPI_interface.h](#).

7.100.1.5 SPI1

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

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

7.100.1.6 SPI2

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

Definition at line 43 of file [SPI_interface.h](#).

7.100.1.7 SPI3

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

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

7.100.1.8 SPIx_MSTR

```
#define SPIx_MSTR 1
```

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

7.100.1.9 SPIx_SLAVE

```
#define SPIx_SLAVE 2
```

Definition at line 75 of file [SPI_interface.h](#).

7.100.1.10 SPIx_SIMPLEX

```
#define SPIx_SIMPLEX 1
```

Definition at line 77 of file [SPI_interface.h](#).

7.100.1.11 SPIx_HALF_DUPLEX

```
#define SPIx_HALF_DUPLEX 2
```

Definition at line 78 of file [SPI_interface.h](#).

7.100.1.12 SPIx_FULL_DUPLEX

```
#define SPIx_FULL_DUPLEX 3
```

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

7.100.2 Function Documentation

7.100.2.1 MSPI_vInit()

```
void MSPI_vInit (
    SPI_MemoryMapType * A_SPIx,
    u8_t A_u8Relationship,
    u8_t A_u8DataDirection )
```

Definition at line 33 of file [SPI_program.c](#).

```
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_VeryHighSpeed , .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_VeryHighSpeed , .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_VeryHighSpeed , .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_VeryHighSpeed , .InputType    = GPIOx_NoPull
00082             .AF_Type       = GPIOx_AF5
00083         } ;
00084
00085         MGPIox_ConfigType MISO2 =
00086         {
00087             .Port          = GPIO_PORTB           , .Pin          = GPIOx_PIN14
00088         }
```

```

00088     .Mode      = GPIOx_MODE_AF      ,   .OutputType = GPIOx_PUSH_PULL ,
00089     .OutputSpeed = GPIOx_VeryHighSpeed ,   .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_VeryHighSpeed ,   .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_VeryHighSpeed ,   .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_VeryHighSpeed ,   .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_VeryHighSpeed ,   .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     SET_BIT( A_SPIx->CR1, CPHA ) ;
00150     SET_BIT( A_SPIx->CR1, CPOL ) ;
00151
00152 // Data format.
00153 #if DFF_MODE == EIGHT_BITS
00154
00155     CLR_BIT( A_SPIx->CR1, DFF ) ;
00156
00157 #elif DFF_MODE == SIXTEEN_BITS
00158
00159     SET_BIT( A_SPIx->CR1, DFF ) ;
00160
00161 #endif
00162
00163 // MSB/LSB first.
00164     CLR_BIT( A_SPIx->CR1, LSBFIRST ) ;
00165
00166 // Select SW slave management.
00167     switch( A_u8RelationShip )
00168 {
00169     case SPIx_MSTR:
00170
00171         SET_BIT( A_SPIx->CR1, SSM ) ;
00172         SET_BIT( A_SPIx->CR1, SSI ) ;
00173
00174     break ;

```

```

00175
00176     case SPIx_SLAVE:
00177
00178         SET_BIT( A_SPIx->CR1, SSM ) ;
00179         CLR_BIT( A_SPIx->CR1, SSI ) ;
00180
00181     break ;
00182
00183 }
00184
00185 // Select Motorola or TI Mode:
00186 CLR_BIT( A_SPIx->CR2, FRF ) ; // Motorola
00187
00188 // MASTER/SLAVE Mode.
00189 switch( A_u8Relationship )
00190 {
00191     case SPIx_MSTR:    SET_BIT( A_SPIx->CR1, MSTR ) ; break ;
00192
00193     case SPIx_SLAVE:   CLR_BIT( A_SPIx->CR1, MSTR ) ; break ;
00194 }
00195
00196 // Data direction:
00197 switch( A_u8Relationship )
00198 {
00199
00200     case SPIx_MSTR:
00201
00202         switch( A_u8DataDirection )
00203     {
00204
00205             case SPIx_FULL_DUPLEX:
00206
00207                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00208                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00209
00210             break ;
00211
00212             case SPIx_SIMPLEX      :
00213
00214                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00215
00216 #if SIMPLEX_MODE == SIMPLEX_TX
00217
00218                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00219
00220 #elif SIMPLEX_MODE == SIMPLEX_RX
00221
00222                 SET_BIT( A_SPIx->CR1, RXONLY ) ;
00223
00224 #endif
00225
00226     break ;
00227
00228     case SPIx_HALF_DUPLEX:
00229
00230 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00231
00232                 SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00233                 SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00234
00235 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00236
00237                 SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00238                 CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00239
00240 #endif
00241
00242     break ;
00243
00244 }
00245
00246     break;
00247
00248     case SPIx_SLAVE:
00249
00250         switch( A_u8DataDirection )
00251     {
00252
00253             case SPIx_FULL_DUPLEX:
00254
00255                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00256                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00257
00258             break ;
00259
00260             case SPIx_SIMPLEX      :
00261

```

```

00262             CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00263
00264 #if SIMPLEX_MODE == SIMPLEX_TX
00265             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00266
00268 #elif SIMPLEX_MODE == SIMPLEX_RX
00269             SET_BIT( A_SPIx->CR1, RXONLY ) ;
00270
00271 #endif
00272         break ;
00273
00274     case SPIx_HALF_DUPLEX:
00275
00276 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00277             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00278             SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00279
00280 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00281             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00282             CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00283
00284 #endif
00285
00286         break ;
00287
00288 #endif
00289
00290         break ;
00291
00292     }
00293
00294     break;
00295
00296 }
00297
00298 // Multi-master mode:
00299 #if MULTI_MSTR_MODE == ENABLE
00300     CLR_BIT( A_SPIx->CR2, SSOE ) ;
00301
00302 #elif MULTI_MSTR_MODE == DISABLE
00303     SET_BIT( A_SPIx->CR2, SSOE ) ;
00304
00305 #endif
00306
00307
00308 // EN/DIS IRQ:
00309 #if TXNEIE_MODE == ENABLE
00310     SET_BIT( A_SPIx->CR2, TXEIE ) ;
00311
00312 #elif TXNEIE_MODE == DISABLE
00313     CLR_BIT( A_SPIx->CR2, TXEIE ) ;
00314
00315 #endif
00316
00317 #endif
00318
00319 #if RXNEIE_MODE == ENABLE
00320     SET_BIT( A_SPIx->CR2, RXNEIE ) ;
00321
00322 #elif RXNEIE_MODE == DISABLE
00323     CLR_BIT( A_SPIx->CR2, RXNEIE ) ;
00324
00325 #endif
00326
00327 #endif
00328
00329 // EN/DIS SPI.
00330     SET_BIT( A_SPIx->CR1, SPE ) ;
00331
00332 }
00333 }
```

References APB1ENR_SPI2EN, APB1ENR_SPI3EN, APB2ENR_SPI1EN, BIDIMODE, BIDIOE, BR_MODE, CLR_BIT, CPHA, CPOL, SPI_MemoryMapType::CR1, SPI_MemoryMapType::CR2, DFF, FRF, GPIO_PORTA, GPIO_PORTB, GPIOx_AF5, GPIOx_MODE_AF, GPIOx_NoPull, GPIOx_PIN13, GPIOx_PIN14, GPIOx_PIN15, GPIOx_PIN3, GPIOx_PIN4, GPIOx_PIN5, GPIOx_PIN6, GPIOx_PIN7, GPIOx_PUSH_PULL, GPIOx_VeryHighSpeed, LSBFIRST, GPIOx_vInit(), MRCC_vEnablePeriphraICLK(), MSTR, GPIOx_ConfigType::Port, RCC_APB1, RCC_APB2, RXNEIE, RXONLY, SET_BIT, SPE, SPI1, SPI2, SPI3, SPIx_FULL_DUPLEX, SPIx_HALF_DUPLEX, SPIx_MSTR, SPIx_SIMPLEX, SPIx_SLAVE, SSI, SSM, SSOE, and TXEIE.

7.100.2.2 MSPI_u16Transcieve()

```
u16_t MSPI_u16Transcieve (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 338 of file [SPI_program.c](#).

```
00339 {
00340
00341     SPI1->DR = A_u16Data ;
00342
00343     // Wait for transmission complete.
00344     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00345     {
00346         asm("NOP") ;
00347     }
00348
00349     // Wait for the last transmission bit.
00350     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00351     {
00352         asm("NOP") ;
00353     }
00354
00355     // Wait for reception complete.
00356     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00357     {
00358         asm("NOP") ;
00359     }
00360
00361     return A_SPIx->DR ;
00362
00363
00364 }
```

References [BSY](#), [SPI_MemoryMapType::DR](#), [FLAG_CLEARED](#), [FLAG_SET](#), [GET_BIT](#), [RXNE](#), [SPI1](#), [SPI_MemoryMapType::SR](#), and [TXE](#).

7.100.2.3 MSPI_vMasterTransmit()

```
void MSPI_vMasterTransmit (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 369 of file [SPI_program.c](#).

```
00370 {
00371
00372     // Start transmission.
00373     SPI1->DR = A_u16Data ;
00374
00375     // Wait for transmission complete.
00376     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00377     {
00378         asm("NOP") ;
00379     }
00380
00381     // Wait for the last transmission bit.
00382     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00383     {
00384         asm("NOP") ;
00385     }
00386
00387 }
```

References [BSY](#), [FLAG_CLEARED](#), [FLAG_SET](#), [GET_BIT](#), [SPI1](#), [SPI_MemoryMapType::SR](#), and [TXE](#).

7.100.2.4 MSPI_u16MasterRecieve()

```
u16_t MSPI_u16MasterRecieve (
    SPI_MemoryMapType * A_SPIx )
```

Definition at line 392 of file [SPI_program.c](#).

```
00393 {
00394
00395     // Wait for reception complete.
00396     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00397     {
00398         asm("NOP") ;
00399     }
00400
00401     // Return data register.
00402     return A_SPIx->DR ;
00403
00404 }
```

References [SPI_MemoryMapType::DR](#), [FLAG_CLEARED](#), [GET_BIT](#), [RXNE](#), and [SPI_MemoryMapType::SR](#).

7.100.2.5 MSPI_vSlaveTransmit()

```
void MSPI_vSlaveTransmit (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 409 of file [SPI_program.c](#).

```
00410 {
00411
00412     // Start transmission.
00413     SPI1->DR = A_u16Data ;
00414
00415     // Wait for transmission complete.
00416     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00417     {
00418         asm("NOP") ;
00419     }
00420
00421     // Wait for the last transmission bit.
00422     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00423     {
00424         asm("NOP") ;
00425     }
00426
00427 }
```

References [BSY](#), [FLAG_CLEARED](#), [FLAG_SET](#), [GET_BIT](#), [SPI1](#), [SPI_MemoryMapType::SR](#), and [TXE](#).

7.100.2.6 MSPI_u16SlaveRecieve()

```
u16_t MSPI_u16SlaveRecieve (
    SPI_MemoryMapType * A_SPIx )
```

Definition at line 432 of file [SPI_program.c](#).

```
00433 {
00434
00435     // Wait for reception complete.
00436     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00437     {
00438         asm("NOP") ;
00439     }
00440
00441     // Return data register.
00442     return A_SPIx->DR ;
00443
00444 }
```

References [SPI_MemoryMapType::DR](#), [FLAG_CLEARED](#), [GET_BIT](#), [RXNE](#), and [SPI_MemoryMapType::SR](#).

7.100.2.7 MSPI_vDISABLE()

```
void MSPI_vDISABLE (
    SPI_MemoryMapType * A_SPIx,
    u8_t A_u8Relationship,
    u8_t A_u8DataDirection )
```

Definition at line 449 of file [SPI_program.c](#).

```
00450 {
00451
00452     switch( A_u8DataDirection )
00453     {
00454
00455         case SPIx_FULL_DUPLEX:
00456
00457             // Wait for reception complete.
00458             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00459
00460             // Wait for transmission complete.
00461             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00462
00463             // Wait for the last transmission bit.
00464             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00465
00466             // Disabling the SPIx.
00467             CLR_BIT( A_SPIx->CR1, SPE ) ;
00468
00469         break ;
00470
00471         case SPIx_SIMPLEX      :
00472
00473 #if SIMPLEX_MODE == SIMPLEX_TX
00474
00475             // Wait for transmission complete.
00476             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00477
00478             // Wait for the last transmission bit.
00479             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00480
00481             // Disabling the SPIx.
00482             CLR_BIT( A_SPIx->CR1, SPE ) ;
00483
00484 #elif SIMPLEX_MODE == SIMPLEX_RX
00485
00486             switch( A_u8Relationship )
00487             {
00488
00489                 case SPIx_MSTR:
00490
00491                     // Wait for reception complete.
00492                     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00493
00494                     // Disabling the SPIx.
00495                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00496
00497                 break ;
00498
00499                 case SPIx_SLAVE:
00500
00501                     // Disabling the SPIx.
00502                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00503
00504                     // Wait for the last transmission bit.
00505                     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00506
00507                 break ;
00508
00509             }
00510
00511 #endif
00512
00513         break ;
00514
00515
00516
00517         case SPIx_HALF_DUPLEX:
00518
00519 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00520
00521             // Wait for transmission complete.
00522             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00523
00524             // Wait for the last transmission bit.
```

```

00525     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00526
00527     // Disabling the SPIx.
00528     CLR_BIT( A_SPIx->CR1, SPE ) ;
00529
00530 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00531
00532     switch( A_u8Relationship )
00533     {
00534
00535         case SPIx_MSTR:
00536
00537             // Wait for reception complete.
00538             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00539
00540             // Disabling the SPIx.
00541             CLR_BIT( A_SPIx->CR1, SPE ) ;
00542
00543             break ;
00544
00545         case SPIx_SLAVE:
00546
00547             // Disabling the SPIx.
00548             CLR_BIT( A_SPIx->CR1, SPE ) ;
00549
00550             // Wait for the last transmission bit.
00551             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00552
00553             break ;
00554
00555     }
00556
00557 #endif
00558
00559     break ;
00560
00561 }
00562
00563 if( A_SPIx == SPI1 )
00564 {
00565     MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_SPI1EN ) ;
00566 }
00567 else if( A_SPIx == SPI2 )
00568 {
00569     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI2EN ) ;
00570 }
00571 else if( A_SPIx == SPI3 )
00572 {
00573     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI3EN ) ;
00574 }
00575
00576 }
```

References APB1ENR_SPI2EN, APB1ENR_SPI3EN, APB2ENR_SPI1EN, BSY, CLR_BIT, SPI_MemoryMapType::CR1, FLAG_CLEARED, FLAG_SET, GET_BIT, MRCC_vDisablePeriphralCLK(), RCC_APB1, RCC_APB2, RXNE, SPE, SPI1, SPI2, SPI3, SPIx_FULL_DUPLEX, SPIx_HALF_DUPLEX, SPIx_MSTR, SPIx_SIMPLEX, SPIx_SLAVE, SPI_MemoryMapType::SR, and TXE.

7.100.2.8 MSPI1_vSetCallBack()

```
void MSPI1_vSetCallBack (
    void(*) (void) Fptr )
```

Definition at line 581 of file [SPI_program.c](#).

```
00582 {
00583     MSPI1_CallBack = Fptr ;
00584 }
```

References [MSPI1_CallBack](#).

7.100.2.9 MSPI2_vSetCallBack()

```
void MSPI2_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 589 of file [SPI_program.c](#).

```
00590 {
00591     MSPI2_CallBack = Fptr ;
00592 }
```

References [MSPI2_CallBack](#).

7.100.2.10 MSPI3_vSetCallBack()

```
void MSPI3_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 597 of file [SPI_program.c](#).

```
00598 {
00599     MSPI3_CallBack = Fptr ;
00600 }
```

References [MSPI3_CallBack](#).

7.101 SPI_interface.h

[Go to the documentation of this file.](#)

```
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 /***** Peripherals declaration *****/
00010 /*
00011 *****/
00012
00013 #define SPI1_BASE_ADDRESS 0x40013000
00014 #define SPI2_BASE_ADDRESS 0x40003800
00015 #define SPI3_BASE_ADDRESS 0x40003C00
00016 #define SPI4_BASE_ADDRESS 0x40013400
00017
00018 typedef struct
00019 {
00020
00021     volatile u32_t CR1 ; 
00022
00023     volatile u32_t CR2 ; 
00024
00025     volatile u32_t SR ; 
00026
00027     volatile u32_t DR ; 
00028
00029     volatile u32_t CRCPR ; 
00030
00031     volatile u32_t RXCRCR ; 
00032
00033     volatile u32_t TXCRCR ; 
00034
00035     volatile u32_t I2SCFGR ; 
00036
00037     volatile u32_t I2SPR ; 
00038
00039 } SPI_MemoryMapType ;
00040
```

```

00041
00042 #define SPI1 ( ( SPI_MemoryMapType*) SPI1_BASE_ADDRESS)
00043 #define SPI2 ( ( SPI_MemoryMapType*) SPI2_BASE_ADDRESS)
00044 #define SPI3 ( ( SPI_MemoryMapType*) SPI3_BASE_ADDRESS)
00045
00046 /***** Functions prototypes *****/
00047 /* Functions prototypes */
00048 /***** */
00049
00050 void MSPI_vInit( SPI_MemoryMapType * A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection ) ;
00051
00052 u16_t MSPI_u16Transcieve( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data ) ;
00053
00054 void MSPI_vMasterTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data ) ;
00055
00056 u16_t MSPI_u16MasterRecieve( SPI_MemoryMapType * A_SPIx ) ;
00057
00058 void MSPI_vSlaveTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data ) ;
00059
00060 u16_t MSPI_u16SlaveRecieve( SPI_MemoryMapType * A_SPIx ) ;
00061
00062 void MSPI_vDISABLE( SPI_MemoryMapType * A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection ) ;
00063
00064 void MSPI1_vSetCallBack( void (*Fptr) (void) ) ;
00065
00066 void MSPI2_vSetCallBack( void (*Fptr) (void) ) ;
00067
00068 void MSPI3_vSetCallBack( void (*Fptr) (void) ) ;
00069
00070 /***** Interfacing macros *****/
00071 /* Interfacing macros */
00072 /***** */
00073
00074 #define SPIx_MSTR    1
00075 #define SPIx_SLAVE   2
00076
00077 #define SPIx_SIMPLEX     1
00078 #define SPIx_HALF_DUPLEX 2
00079 #define SPIx_FULL_DUPLEX 3
00080
00081
00082
00083
00084
00085
00086
00087
00088 #endif // _SPI_interface_H

```

7.102 COTS/MCAL/SPI/SPI_private.h File Reference

Macros

- #define CPHA 0
- #define CPOL 1
- #define MSTR 2
- #define SPE 6
- #define LSBFIRST 7
- #define SSI 8
- #define SSM 9
- #define RXONLY 10
- #define DFF 11
- #define CRCNEXT 12
- #define CRCEN 13
- #define BIDIOE 14
- #define BIDIMODE 15
- #define RXDMAEN 0
- #define TXDMAEN 1
- #define SSOE 2
- #define FRF 4

- #define **ERRIE** 5
- #define **RXNEIE** 6
- #define **TXEIE** 7
- #define **RXNE** 0
- #define **TXE** 1
- #define **CHSIDE** 2
- #define **UDR** 3
- #define **CRCERR** 4
- #define **MODF** 5
- #define **OVR** 6
- #define **BSY** 7
- #define **FRE** 8
- #define **FIRST_CLK_Captured** 1
- #define **SECOND_CLK_Captured** 0
- #define **CLK_IdleAt0** 0
- #define **CLK_IdleAt1** 1
- #define **SLAVE** 0
- #define **MASTER** 1
- #define **CLK_By2** 0b000U
- #define **CLK_By4** 0b001U
- #define **CLK_By8** 0b010U
- #define **CLK_By16** 0b011U
- #define **CLK_By32** 0b100U
- #define **CLK_By64** 0b101U
- #define **CLK_By128** 0b110U
- #define **CLK_By256** 0b111U
- #define **ENABLE** 1
- #define **DISABLE** 0
- #define **MSB_First** 0
- #define **LSB_First** 1
- #define **FULL_DUPLEX** 0
- #define **RX_ONLY** 1
- #define **EIGHT_BITS** 0
- #define **SIXTEEN_BITS** 1
- #define **CRC_PHASE** 1
- #define **NO_CRC_PHASE** 0
- #define **OneLine** 1
- #define **TwoLines** 0
- #define **MOTOROLA** 0
- #define **TI** 1
- #define **EMPTY** 0
- #define **NOTEMPTY** 1
- #define **LEFT** 0
- #define **RIGHT** 1
- #define **OCCURRED** 1
- #define **NOT_OCCURRED** 0
- #define **MATCHED** 0
- #define **NOT_MATCHED** 1
- #define **FAULT_OCCURRED** 1
- #define **NO_FAULT_OCCURRED** 0
- #define **BUSY** 1
- #define **NOT_BUSY** 0
- #define **ERROR** 1
- #define **NO_ERROR** 0
- #define **HALF_DUPLEX_TX** 1

- #define HALF_DUPLEX_RX 2
- #define SIMPLEX_TX 1
- #define SIMPLEX_RX 2
- #define SR_RESET 0x0002

7.102.1 Macro Definition Documentation

7.102.1.1 CPHA

```
#define CPHA 0
```

Definition at line 11 of file [SPI_private.h](#).

7.102.1.2 CPOL

```
#define CPOL 1
```

Definition at line 12 of file [SPI_private.h](#).

7.102.1.3 MSTR

```
#define MSTR 2
```

Definition at line 13 of file [SPI_private.h](#).

7.102.1.4 SPE

```
#define SPE 6
```

Definition at line 14 of file [SPI_private.h](#).

7.102.1.5 LSBFIRST

```
#define LSBFIRST 7
```

Definition at line 15 of file [SPI_private.h](#).

7.102.1.6 SSI

```
#define SSI 8
```

Definition at line 16 of file [SPI_private.h](#).

7.102.1.7 SSM

```
#define SSM 9
```

Definition at line 17 of file [SPI_private.h](#).

7.102.1.8 RXONLY

```
#define RXONLY 10
```

Definition at line 18 of file [SPI_private.h](#).

7.102.1.9 DFF

```
#define DFF 11
```

Definition at line 19 of file [SPI_private.h](#).

7.102.1.10 CRCNEXT

```
#define CRCNEXT 12
```

Definition at line 20 of file [SPI_private.h](#).

7.102.1.11 CRCEN

```
#define CRCEN 13
```

Definition at line 21 of file [SPI_private.h](#).

7.102.1.12 BIDIOE

```
#define BIDIOE 14
```

Definition at line 22 of file [SPI_private.h](#).

7.102.1.13 BIDIMODE

```
#define BIDIMODE 15
```

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

7.102.1.14 RXDMAEN

```
#define RXDMAEN 0
```

Definition at line 26 of file [SPI_private.h](#).

7.102.1.15 TXDMAEN

```
#define TXDMAEN 1
```

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

7.102.1.16 SSOE

```
#define SSOE 2
```

Definition at line 28 of file [SPI_private.h](#).

7.102.1.17 FRF

```
#define FRF 4
```

Definition at line 29 of file [SPI_private.h](#).

7.102.1.18 ERRIE

```
#define ERRIE 5
```

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

7.102.1.19 RXNEIE

```
#define RXNEIE 6
```

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

7.102.1.20 TXEIE

```
#define TXEIE 7
```

Definition at line 32 of file [SPI_private.h](#).

7.102.1.21 RXNE

```
#define RXNE 0
```

Definition at line 35 of file [SPI_private.h](#).

7.102.1.22 TXE

```
#define TXE 1
```

Definition at line 36 of file [SPI_private.h](#).

7.102.1.23 CHSIDE

```
#define CHSIDE 2
```

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

7.102.1.24 UDR

```
#define UDR 3
```

Definition at line 38 of file [SPI_private.h](#).

7.102.1.25 CRCERR

```
#define CRCERR 4
```

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

7.102.1.26 MODF

```
#define MODF 5
```

Definition at line 40 of file [SPI_private.h](#).

7.102.1.27 OVR

```
#define OVR 6
```

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

7.102.1.28 BSY

```
#define BSY 7
```

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

7.102.1.29 FRE

```
#define FRE 8
```

Definition at line 43 of file [SPI_private.h](#).

7.102.1.30 FIRST_CLK_Captured

```
#define FIRST_CLK_Captured 1
```

Definition at line 49 of file [SPI_private.h](#).

7.102.1.31 SECOND_CLK_Captured

```
#define SECOND_CLK_Captured 0
```

Definition at line 50 of file [SPI_private.h](#).

7.102.1.32 CLK_IdleAt0

```
#define CLK_IdleAt0 0
```

Definition at line 52 of file [SPI_private.h](#).

7.102.1.33 CLK_IdleAt1

```
#define CLK_IdleAt1 1
```

Definition at line 53 of file [SPI_private.h](#).

7.102.1.34 SLAVE

```
#define SLAVE 0
```

Definition at line 55 of file [SPI_private.h](#).

7.102.1.35 MASTER

```
#define MASTER 1
```

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

7.102.1.36 CLK_By2

```
#define CLK_By2 0b000U
```

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

7.102.1.37 CLK_By4

```
#define CLK_By4 0b001U
```

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

7.102.1.38 CLK_By8

```
#define CLK_By8 0b010U
```

Definition at line 60 of file [SPI_private.h](#).

7.102.1.39 CLK_By16

```
#define CLK_By16 0b011U
```

Definition at line 61 of file [SPI_private.h](#).

7.102.1.40 CLK_By32

```
#define CLK_By32 0b100U
```

Definition at line 62 of file [SPI_private.h](#).

7.102.1.41 CLK_By64

```
#define CLK_By64 0b101U
```

Definition at line 63 of file [SPI_private.h](#).

7.102.1.42 CLK_By128

```
#define CLK_By128 0b110U
```

Definition at line 64 of file [SPI_private.h](#).

7.102.1.43 CLK_By256

```
#define CLK_By256 0b111U
```

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

7.102.1.44 ENABLE

```
#define ENABLE 1
```

Definition at line 67 of file [SPI_private.h](#).

7.102.1.45 DISABLE

```
#define DISABLE 0
```

Definition at line 68 of file [SPI_private.h](#).

7.102.1.46 MSB_First

```
#define MSB_First 0
```

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

7.102.1.47 LSB_First

```
#define LSB_First 1
```

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

7.102.1.48 FULL_DUPLEX

```
#define FULL_DUPLEX 0
```

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

7.102.1.49 RX_ONLY

```
#define RX_ONLY 1
```

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

7.102.1.50 EIGHT_BITS

```
#define EIGHT_BITS 0
```

Definition at line [76](#) of file [SPI_private.h](#).

7.102.1.51 SIXTEEN_BITS

```
#define SIXTEEN_BITS 1
```

Definition at line [77](#) of file [SPI_private.h](#).

7.102.1.52 CRC_PHASE

```
#define CRC_PHASE 1
```

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

7.102.1.53 NO_CRC_PHASE

```
#define NO_CRC_PHASE 0
```

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

7.102.1.54 OneLine

```
#define OneLine 1
```

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

7.102.1.55 TwoLines

```
#define TwoLines 0
```

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

7.102.1.56 MOTOROLA

```
#define MOTOROLA 0
```

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

7.102.1.57 TI

```
#define TI 1
```

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

7.102.1.58 EMPTY

```
#define EMPTY 0
```

Definition at line 88 of file [SPI_private.h](#).

7.102.1.59 NOTEMPTY

```
#define NOTEMPTY 1
```

Definition at line 89 of file [SPI_private.h](#).

7.102.1.60 LEFT

```
#define LEFT 0
```

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

7.102.1.61 RIGHT

```
#define RIGHT 1
```

Definition at line 92 of file [SPI_private.h](#).

7.102.1.62 OCCURRED

```
#define OCCURRED 1
```

Definition at line 94 of file [SPI_private.h](#).

7.102.1.63 NOT_OCCURRED

```
#define NOT_OCCURRED 0
```

Definition at line 95 of file [SPI_private.h](#).

7.102.1.64 MATCHED

```
#define MATCHED 0
```

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

7.102.1.65 NOT_MATCHED

```
#define NOT_MATCHED 1
```

Definition at line 98 of file [SPI_private.h](#).

7.102.1.66 FAULT_OCCURRED

```
#define FAULT_OCCURRED 1
```

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

7.102.1.67 NO_FAULT_OCCURRED

```
#define NO_FAULT_OCCURRED 0
```

Definition at line 101 of file [SPI_private.h](#).

7.102.1.68 BUSY

```
#define BUSY 1
```

Definition at line 103 of file [SPI_private.h](#).

7.102.1.69 NOT_BUSY

```
#define NOT_BUSY 0
```

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

7.102.1.70 ERROR

```
#define ERROR 1
```

Definition at line 106 of file [SPI_private.h](#).

7.102.1.71 NO_ERROR

```
#define NO_ERROR 0
```

Definition at line 107 of file [SPI_private.h](#).

7.102.1.72 HALF_DUPLEX_TX

```
#define HALF_DUPLEX_TX 1
```

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

7.102.1.73 HALF_DUPLEX_RX

```
#define HALF_DUPLEX_RX 2
```

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

7.102.1.74 SIMPLEX_TX

```
#define SIMPLEX_TX 1
```

Definition at line 114 of file [SPI_private.h](#).

7.102.1.75 SIMPLEX_RX

```
#define SIMPLEX_RX 2
```

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

7.102.1.76 SR_RESET

```
#define SR_RESET 0x0002
```

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

7.103 SPI_private.h

[Go to the documentation of this file.](#)

```
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

```

7.104 COTS/MCAL/SPI/SPI_program.c File Reference

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../RCC/MRCC_interface.h"
#include "../GPIO/GPIO_interface.h"
#include "SPI_interface.h"
#include "SPI_private.h"
#include "SPI_config.h"

```

Functions

- void **MSPI_vInit** (SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection)
- u16_t **MSPI_u16Transcieve** (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)
- void **MSPI_vMasterTransmit** (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)
- u16_t **MSPI_u16MasterRecieve** (SPI_MemoryMapType *A_SPIx)
- void **MSPI_vSlaveTransmit** (SPI_MemoryMapType *A_SPIx, u16_t A_u16Data)
- u16_t **MSPI_u16SlaveRecieve** (SPI_MemoryMapType *A_SPIx)
- void **MSPI_vDISABLE** (SPI_MemoryMapType *A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection)

- void **MSPI1_vSetCallBack** (void(*Fptr)(void))
- void **MSPI2_vSetCallBack** (void(*Fptr)(void))
- void **MSPI3_vSetCallBack** (void(*Fptr)(void))
- void **SPI1_IRQHandler** (void)
- void **SPI2_IRQHandler** (void)
- void **SPI3_IRQHandler** (void)

Variables

- void(* **MSPI1_CallBack**)(void)
- void(* **MSPI2_CallBack**)(void)
- void(* **MSPI3_CallBack**)(void)

7.104.1 Function Documentation

7.104.1.1 **MSPI_vInit()**

```
void MSPI_vInit (
    SPI_MemoryMapType * A_SPIx,
    u8_t A_u8Relationship,
    u8_t A_u8DataDirection )
```

Definition at line 33 of file [SPI_program.c](#).

```
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_VeryHighSpeed , .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_VeryHighSpeed , .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_VeryHighSpeed , .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_PIN5
00080         .Mode       = GPIOx_MODE_AF, .OutputType = GPIOx_PUSH_PULL
00081         .OutputSpeed = GPIOx_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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 SET_BIT( A_SPIx->CR1, CPHA ) ;
00150 SET_BIT( A_SPIx->CR1, CPOL ) ;
00151
00152 // Data format.
00153 #if DFF_MODE == EIGHT_BITS
00154
00155     CLR_BIT( A_SPIx->CR1, DFF ) ;
00156
00157 #elif DFF_MODE == SIXTEEN_BITS
00158
00159     SET_BIT( A_SPIx->CR1, DFF ) ;
00160
00161 #endif
00162

```

```

00163     // MSB/LSB first.
00164     CLR_BIT( A_SPIx->CR1, LSBFIRST ) ;
00165
00166     // Select SW slave management.
00167     switch( A_u8Relationship )
00168     {
00169         case SPIx_MSTR:
00170             SET_BIT( A_SPIx->CR1, SSM ) ;
00171             SET_BIT( A_SPIx->CR1, SSI ) ;
00172
00173             break ;
00174
00175         case SPIx_SLAVE:
00176
00177             SET_BIT( A_SPIx->CR1, SSM ) ;
00178             CLR_BIT( A_SPIx->CR1, SSI ) ;
00179
00180             break ;
00181
00182     }
00183
00184
00185     // Select Motorola or TI Mode:
00186     CLR_BIT( A_SPIx->CR2, FRF ) ; // Motorola
00187
00188     // MASTER/SLAVE Mode.
00189     switch( A_u8Relationship )
00190     {
00191         case SPIx_MSTR:    SET_BIT( A_SPIx->CR1, MSTR ) ; break ;
00192
00193         case SPIx_SLAVE:   CLR_BIT( A_SPIx->CR1, MSTR ) ; break ;
00194     }
00195
00196     // Data direction:
00197     switch( A_u8Relationship )
00198     {
00199
00200         case SPIx_MSTR:
00201
00202             switch( A_u8DataDirection )
00203             {
00204
00205                 case SPIx_FULL_DUPLEX:
00206
00207                     CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00208                     CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00209
00210                     break ;
00211
00212                 case SPIx_SIMPLEX      :
00213
00214                     CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00215
00216 #if SIMPLEX_MODE == SIMPLEX_TX
00217
00218                     CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00219
00220 #elif SIMPLEX_MODE == SIMPLEX_RX
00221
00222                     SET_BIT( A_SPIx->CR1, RXONLY ) ;
00223
00224 #endif
00225
00226                     break ;
00227
00228                 case SPIx_HALF_DUPLEX:
00229
00230 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00231
00232                     SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00233                     SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00234
00235 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00236
00237                     SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00238                     CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00239
00240 #endif
00241
00242                     break ;
00243
00244             }
00245
00246             break;
00247
00248         case SPIx_SLAVE:
00249

```

```

00250     switch( A_u8DataDirection )
00251     {
00252         case SPIx_FULL_DUPLEX:
00253             CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00254             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00255             break ;
00256         case SPIx_SIMPLEX      :
00257             CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00258 #if SIMPLEX_MODE == SIMPLEX_TX
00259             CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00260 #elif SIMPLEX_MODE == SIMPLEX_RX
00261             SET_BIT( A_SPIx->CR1, RXONLY ) ;
00262 #endif
00263         case SPIx_HALF_DUPLEX:
00264 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00265             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00266             SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00267 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00268             SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00269             CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00270 #endif
00271         break ;
00272     }
00273     break ;
00274     case SPIx_MULTI_MASTER:
00275         // Multi-master mode:
00276 #if MULTI_MSTR_MODE == ENABLE
00277             CLR_BIT( A_SPIx->CR2, SSOE ) ;
00278 #elif MULTI_MSTR_MODE == DISABLE
00279             SET_BIT( A_SPIx->CR2, SSOE ) ;
00280 #endif
00281         break;
00282     }
00283     break ;
00284     case SPIx_EN_DIS_IRQ:
00285         // EN/DIS IRQ:
00286 #if TXNEIE_MODE == ENABLE
00287             SET_BIT( A_SPIx->CR2, TXEIE ) ;
00288 #elif TXNEIE_MODE == DISABLE
00289             CLR_BIT( A_SPIx->CR2, TXEIE ) ;
00290 #endif
00291         break;
00292     }
00293     break;
00294     case SPIx_EN_DIS_SPI:
00295         // EN/DIS SPI.
00296 #if RXNEIE_MODE == ENABLE
00297             SET_BIT( A_SPIx->CR2, RXNEIE ) ;
00298 #elif RXNEIE_MODE == DISABLE
00299             CLR_BIT( A_SPIx->CR2, RXNEIE ) ;
00300 #endif
00301         break;
00302     }
00303     break;
00304     case SPIx_EN_DIS_FRF:
00305         // EN/DIS FRF.
00306 #if DFF_MODE == ENABLE
00307             SET_BIT( A_SPIx->CR2, DFF ) ;
00308 #elif DFF_MODE == DISABLE
00309             CLR_BIT( A_SPIx->CR2, DFF ) ;
00310 #endif
00311         break;
00312     }
00313     break;
00314     case SPIx_EN_DIS_GPIO:
00315         // EN/DIS GPIO.
00316 #if BR_MODE == ENABLE
00317             SET_BIT( A_SPIx->CR2, BR_MODE ) ;
00318 #elif BR_MODE == DISABLE
00319             CLR_BIT( A_SPIx->CR2, BR_MODE ) ;
00320 #endif
00321         break;
00322     }
00323     break;
00324     case SPIx_EN_DIS_CPHA:
00325         // EN/DIS CPHA.
00326 #if CPOL_MODE == ENABLE
00327             SET_BIT( A_SPIx->CR2, CPOL ) ;
00328 #elif CPOL_MODE == DISABLE
00329             CLR_BIT( A_SPIx->CR2, CPOL ) ;
00330 #endif
00331         break;
00332     }
00333 }

```

References [APB1ENR_SPI2EN](#), [APB1ENR_SPI3EN](#), [APB2ENR_SPI1EN](#), [BIDIMODE](#), [BIDIOE](#), [BR_MODE](#), [CLR_BIT](#), [CPOL](#), [SPI_MemoryMapType::CR1](#), [SPI_MemoryMapType::CR2](#), [DFF](#), [FRF](#), [GPIO_PORTA](#),

`GPIO_PORTB, GPIOx_AF5, GPIOx_MODE_AF, GPIOx_NoPull, GPIOx_PIN13, GPIOx_PIN14, GPIOx_PIN15, GPIOx_PIN3, GPIOx_PIN4, GPIOx_PIN5, GPIOx_PIN6, GPIOx_PIN7, GPIOx_PUSH_PULL, GPIOx_VeryHighSpeed, LSBFIRST, MGPIOx_vInit(), MRCC_vEnablePeripheralCLK(), MSTR, MGPIOx_ConfigType::Port, RCC_APB1, RCC_APB2, RXNEIE, RXONLY, SET_BIT, SPE, SPI1, SPI2, SPI3, SPIx_FULL_DUPLEX, SPIx_HALF_DUPLEX, SPIx_MSTR, SPIx_SIMPLEX, SPIx_SLAVE, SSI, SSM, SSOE, and TXEIE.`

7.104.1.2 MSPI_u16Transcieve()

```
u16_t MSPI_u16Transcieve (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 338 of file [SPI_program.c](#).

```
00339 {
00340     SPI1->DR = A_u16Data ;
00342
00343     // Wait for transmission complete.
00344     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00345     {
00346         asm("NOP") ;
00347     }
00348
00349     // Wait for the last transmission bit.
00350     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00351     {
00352         asm("NOP") ;
00353     }
00354
00355     // Wait for reception complete.
00356     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00357     {
00358         asm("NOP") ;
00359     }
00360
00361     return A_SPIx->DR ;
00363
00364 }
```

References `BSY`, `SPI_MemoryMapType::DR`, `FLAG_CLEARED`, `FLAG_SET`, `GET_BIT`, `RXNE`, `SPI1`, `SPI_MemoryMapType::SR`, and `TXE`.

7.104.1.3 MSPI_vMasterTransmit()

```
void MSPI_vMasterTransmit (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 369 of file [SPI_program.c](#).

```
00370 {
00371     // Start transmission.
00372     SPI1->DR = A_u16Data ;
00374
00375     // Wait for transmission complete.
00376     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00377     {
00378         asm("NOP") ;
00379     }
00380
00381     // Wait for the last transmission bit.
00382     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00383     {
00384         asm("NOP") ;
00385     }
00386
00387 }
```

References `BSY`, `FLAG_CLEARED`, `FLAG_SET`, `GET_BIT`, `SPI1`, `SPI_MemoryMapType::SR`, and `TXE`.

7.104.1.4 MSPI_u16MasterRecieve()

```
u16_t MSPI_u16MasterRecieve (
    SPI_MemoryMapType * A_SPIx )
```

Definition at line 392 of file [SPI_program.c](#).

```
00393 {
00394
00395     // Wait for reception complete.
00396     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00397     {
00398         asm("NOP") ;
00399     }
00400
00401     // Return data register.
00402     return A_SPIx->DR ;
00403
00404 }
```

References [SPI_MemoryMapType::DR](#), [FLAG_CLEARED](#), [GET_BIT](#), [RXNE](#), and [SPI_MemoryMapType::SR](#).

7.104.1.5 MSPI_vSlaveTransmit()

```
void MSPI_vSlaveTransmit (
    SPI_MemoryMapType * A_SPIx,
    u16_t A_u16Data )
```

Definition at line 409 of file [SPI_program.c](#).

```
00410 {
00411
00412     // Start transmission.
00413     SPI1->DR = A_u16Data ;
00414
00415     // Wait for transmission complete.
00416     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00417     {
00418         asm("NOP") ;
00419     }
00420
00421     // Wait for the last transmission bit.
00422     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00423     {
00424         asm("NOP") ;
00425     }
00426
00427 }
```

References [BSY](#), [FLAG_CLEARED](#), [FLAG_SET](#), [GET_BIT](#), [SPI1](#), [SPI_MemoryMapType::SR](#), and [TXE](#).

7.104.1.6 MSPI_u16SlaveRecieve()

```
u16_t MSPI_u16SlaveRecieve (
    SPI_MemoryMapType * A_SPIx )
```

Definition at line 432 of file [SPI_program.c](#).

```
00433 {
00434
00435     // Wait for reception complete.
00436     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00437     {
00438         asm("NOP") ;
00439     }
00440
00441     // Return data register.
00442     return A_SPIx->DR ;
00443
00444 }
```

References [SPI_MemoryMapType::DR](#), [FLAG_CLEARED](#), [GET_BIT](#), [RXNE](#), and [SPI_MemoryMapType::SR](#).

7.104.1.7 MSPI_vDISABLE()

```
void MSPI_vDISABLE (
    SPI_MemoryMapType * A_SPIx,
    u8_t A_u8Relationship,
    u8_t A_u8DataDirection )
```

Definition at line 449 of file [SPI_program.c](#).

```
00450 {
00451
00452     switch( A_u8DataDirection )
00453     {
00454
00455         case SPIx_FULL_DUPLEX:
00456
00457             // Wait for reception complete.
00458             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00459
00460             // Wait for transmission complete.
00461             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00462
00463             // Wait for the last transmission bit.
00464             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00465
00466             // Disabling the SPIx.
00467             CLR_BIT( A_SPIx->CR1, SPE ) ;
00468
00469         break ;
00470
00471         case SPIx_SIMPLEX      :
00472
00473 #if SIMPLEX_MODE == SIMPLEX_TX
00474
00475             // Wait for transmission complete.
00476             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00477
00478             // Wait for the last transmission bit.
00479             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00480
00481             // Disabling the SPIx.
00482             CLR_BIT( A_SPIx->CR1, SPE ) ;
00483
00484 #elif SIMPLEX_MODE == SIMPLEX_RX
00485
00486             switch( A_u8Relationship )
00487             {
00488
00489                 case SPIx_MSTR:
00490
00491                     // Wait for reception complete.
00492                     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00493
00494                     // Disabling the SPIx.
00495                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00496
00497                 break ;
00498
00499                 case SPIx_SLAVE:
00500
00501                     // Disabling the SPIx.
00502                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00503
00504                     // Wait for the last transmission bit.
00505                     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00506
00507                 break ;
00508
00509             }
00510
00511 #endif
00512
00513         break ;
00514
00515
00516
00517         case SPIx_HALF_DUPLEX:
00518
00519 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00520
00521             // Wait for transmission complete.
00522             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00523
00524             // Wait for the last transmission bit.
```

```

00525     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00526
00527     // Disabling the SPIx.
00528     CLR_BIT( A_SPIx->CR1, SPE ) ;
00529
00530 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00531
00532     switch( A_u8Relationship )
00533     {
00534
00535         case SPIx_MSTR:
00536
00537             // Wait for reception complete.
00538             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00539
00540             // Disabling the SPIx.
00541             CLR_BIT( A_SPIx->CR1, SPE ) ;
00542
00543             break ;
00544
00545         case SPIx_SLAVE:
00546
00547             // Disabling the SPIx.
00548             CLR_BIT( A_SPIx->CR1, SPE ) ;
00549
00550             // Wait for the last transmission bit.
00551             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00552
00553             break ;
00554
00555     }
00556
00557 #endif
00558
00559     break ;
00560
00561 }
00562
00563 if( A_SPIx == SPI1 )
00564 {
00565     MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_SPI1EN ) ;
00566 }
00567 else if( A_SPIx == SPI2 )
00568 {
00569     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI2EN ) ;
00570 }
00571 else if( A_SPIx == SPI3 )
00572 {
00573     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI3EN ) ;
00574 }
00575
00576 }
```

References APB1ENR_SPI2EN, APB1ENR_SPI3EN, APB2ENR_SPI1EN, BSY, CLR_BIT, SPI_MemoryMapType::CR1, FLAG_CLEARED, FLAG_SET, GET_BIT, MRCC_vDisablePeriphralCLK(), RCC_APB1, RCC_APB2, RXNE, SPE, SPI1, SPI2, SPI3, SPIx_FULL_DUPLEX, SPIx_HALF_DUPLEX, SPIx_MSTR, SPIx_SIMPLEX, SPIx_SLAVE, SPI_MemoryMapType::SR, and TXE.

7.104.1.8 MSPI1_vSetCallBack()

```
void MSPI1_vSetCallBack (
    void(*) (void) Fptr )
```

Definition at line 581 of file [SPI_program.c](#).

```
00582 {
00583     MSPI1_CallBack = Fptr ;
00584 }
```

References [MSPI1_CallBack](#).

7.104.1.9 MSPI2_vSetCallBack()

```
void MSPI2_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 589 of file SPI_program.c.

```
00590 {
00591     MSPI2_CallBack = Fptr ;
00592 }
```

References [MSPI2_CallBack](#).

7.104.1.10 MSPI3_vSetCallBack()

```
void MSPI3_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 597 of file SPI_program.c.

```
00598 {
00599     MSPI3_CallBack = Fptr ;
00600 }
```

References [MSPI3_CallBack](#).

7.104.1.11 SPI1_IRQHandler()

```
void SPI1_IRQHandler (
    void )
```

Definition at line 605 of file SPI_program.c.

```
00606 {
00607
00608     // CLEAR FLAGS:
00609     SPI1->SR = SR_RESET ;
00610
00611     MSPI1_CallBack() ;
00612
00613 }
```

References [MSPI1_CallBack](#), [SPI1](#), and [SR_RESET](#).

7.104.1.12 SPI2_IRQHandler()

```
void SPI2_IRQHandler (
    void )
```

Definition at line 618 of file SPI_program.c.

```
00619 {
00620
00621     // CLEAR FLAGS:
00622     SPI2->SR = SR_RESET ;
00623
00624     MSPI2_CallBack() ;
00625
00626 }
```

References [MSPI2_CallBack](#), [SPI2](#), and [SR_RESET](#).

7.104.1.13 SPI3_IRQHandler()

```
void SPI3_IRQHandler (
    void )
```

Definition at line 631 of file [SPI_program.c](#).

```
00632 {
00633
00634     // CLEAR FLAGS:
00635     SPI3->SR = SR_RESET ;
00636
00637     MSPI3_CallBack() ;
00638
00639 }
```

References [MSPI3_CallBack](#), [SPI3](#), and [SR_RESET](#).

7.104.2 Variable Documentation

7.104.2.1 MSPI1_Callback

```
void(* MSPI1_Callback) (void) (
    void )
```

Definition at line 25 of file [SPI_program.c](#).

Referenced by [MSPI1_vSetCallBack\(\)](#), and [SPI1_IRQHandler\(\)](#).

7.104.2.2 MSPI2_Callback

```
void(* MSPI2_Callback) (void) (
    void )
```

Definition at line 26 of file [SPI_program.c](#).

Referenced by [MSPI2_vSetCallBack\(\)](#), and [SPI2_IRQHandler\(\)](#).

7.104.2.3 MSPI3_Callback

```
void(* MSPI3_Callback) (void) (
    void )
```

Definition at line 27 of file [SPI_program.c](#).

Referenced by [MSPI3_vSetCallBack\(\)](#), and [SPI3_IRQHandler\(\)](#).

7.105 SPI_program.c

[Go to the documentation of this file.](#)

```

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 /* 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 #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_VeryHighSpeed , .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_VeryHighSpeed , .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_VeryHighSpeed , .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_VeryHighSpeed , .InputType = GPIOx_NoPull
00082             .AF_Type   = GPIOx_AF5
00083
00084     }
00085 }
```

```

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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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_VeryHighSpeed, .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 SET_BIT( A_SPIx->CR1, CPHA ) ;
00150 SET_BIT( A_SPIx->CR1, CPOL ) ;
00151
00152 // Data format.
00153 #if DFF_MODE == EIGHT_BITS
00154
00155     CLR_BIT( A_SPIx->CR1, DFF ) ;
00156
00157 #elif DFF_MODE == SIXTEEN_BITS
00158
00159     SET_BIT( A_SPIx->CR1, DFF ) ;
00160
00161 #endif
00162
00163 // MSB/LSB first.
00164 CLR_BIT( A_SPIx->CR1, LSBFIRST ) ;
00165
00166 // Select SW slave management.
00167 switch( A_u8RelationShip )
00168 {
00169     case SPIx_MSTR:

```

```
00170
00171     SET_BIT( A_SPIx->CR1, SSM ) ;
00172     SET_BIT( A_SPIx->CR1, SSI ) ;
00173
00174     break ;
00175
00176     case SPIx_SLAVE:
00177
00178         SET_BIT( A_SPIx->CR1, SSM ) ;
00179         CLR_BIT( A_SPIx->CR1, SSI ) ;
00180
00181     break ;
00182
00183 }
00184
00185 // Select Motorola or TI Mode:
00186 CLR_BIT( A_SPIx->CR2, FRF ) ; // Motorola
00187
00188 // MASTER/SLAVE Mode.
00189 switch( A_u8Relationship )
00190 {
00191     case SPIx_MSTR:    SET_BIT( A_SPIx->CR1, MSTR ) ; break ;
00192
00193     case SPIx_SLAVE:   CLR_BIT( A_SPIx->CR1, MSTR ) ; break ;
00194 }
00195
00196 // Data direction:
00197 switch( A_u8Relationship )
00198 {
00199
00200     case SPIx_MSTR:
00201
00202         switch( A_u8DataDirection )
00203     {
00204
00205             case SPIx_FULL_DUPLEX:
00206
00207                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00208                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00209
00210             break ;
00211
00212             case SPIx_SIMPLEX      :
00213
00214                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00215
00216 #if SIMPLEX_MODE == SIMPLEX_TX
00217
00218                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00219
00220 #elif SIMPLEX_MODE == SIMPLEX_RX
00221
00222                 SET_BIT( A_SPIx->CR1, RXONLY ) ;
00223
00224 #endif
00225
00226     break ;
00227
00228     case SPIx_HALF_DUPLEX:
00229
00230 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00231
00232                 SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00233                 SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00234
00235 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00236
00237                 SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00238                 CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00239
00240 #endif
00241
00242     break ;
00243
00244 }
00245
00246     break;
00247
00248     case SPIx_SLAVE:
00249
00250         switch( A_u8DataDirection )
00251     {
00252
00253             case SPIx_FULL_DUPLEX:
00254
00255                 CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00256                 CLR_BIT( A_SPIx->CR1, RXONLY ) ;
```

```

00257
00258         break ;
00259
00260     case SPIx_SIMPLEX      :
00261         CLR_BIT( A_SPIx->CR1, BIDIMODE ) ;
00262
00263 #if SIMPLEX_MODE == SIMPLEX_TX
00264
00265         CLR_BIT( A_SPIx->CR1, RXONLY ) ;
00266
00267 #elif SIMPLEX_MODE == SIMPLEX_RX
00268
00269         SET_BIT( A_SPIx->CR1, RXONLY ) ;
00270
00271 #endif
00272
00273         break ;
00274
00275     case SPIx_HALF_DUPLEX:
00276
00277 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00278
00279         SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00280         SET_BIT( A_SPIx->CR1, BIDIOE ) ;
00281
00282 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00283
00284         SET_BIT( A_SPIx->CR1, BIDIMODE ) ;
00285         CLR_BIT( A_SPIx->CR1, BIDIOE ) ;
00286
00287 #endif
00288
00289         break ;
00290
00291     }
00292
00293     break;
00294
00295     }
00296
00297 // Multi-master mode:
00298 #if MULTI_MSTR_MODE == ENABLE
00299
00300         CLR_BIT( A_SPIx->CR2, SSOE ) ;
00301
00302 #elif MULTI_MSTR_MODE == DISABLE
00303
00304         SET_BIT( A_SPIx->CR2, SSOE ) ;
00305
00306 #endif
00307
00308 // EN/DIS IRQ:
00309 #if TXNEIE_MODE == ENABLE
00310
00311         SET_BIT( A_SPIx->CR2, TXEIE ) ;
00312
00313 #elif TXNEIE_MODE == DISABLE
00314
00315         CLR_BIT( A_SPIx->CR2, TXEIE ) ;
00316
00317 #endif
00318
00319 #if RXNEIE_MODE == ENABLE
00320
00321         SET_BIT( A_SPIx->CR2, RXNEIE ) ;
00322
00323 #elif RXNEIE_MODE == DISABLE
00324
00325         CLR_BIT( A_SPIx->CR2, RXNEIE ) ;
00326
00327 #endif
00328
00329 // EN/DIS SPI.
00330         SET_BIT( A_SPIx->CR1, SPE ) ;
00331
00332
00333 }
00334
00335 /***** */
00336 /***** */
00337
00338 u16_t MSPI_u16Transcieve( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00339 {
00340
00341     SPI1->DR = A_u16Data ;
00342
00343 // Wait for transmission complete.

```

```

00344     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00345     {
00346         asm("NOP") ;
00347     }
00348
00349     // Wait for the last transmission bit.
00350     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00351     {
00352         asm("NOP") ;
00353     }
00354
00355     // Wait for reception complete.
00356     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00357     {
00358         asm("NOP") ;
00359     }
00360
00361     return A_SPIx->DR ;
00362
00363 }
00364
00365
00366 /*****
00367 *****/
00368
00369 void MSPI_vMasterTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00370 {
00371
00372     // Start transmission.
00373     SPI1->DR = A_u16Data ;
00374
00375     // Wait for transmission complete.
00376     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00377     {
00378         asm("NOP") ;
00379     }
00380
00381     // Wait for the last transmission bit.
00382     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00383     {
00384         asm("NOP") ;
00385     }
00386
00387 }
00388
00389 /*****
00390 *****/
00391
00392 u16_t MSPI_u16MasterRecieve( SPI_MemoryMapType * A_SPIx )
00393 {
00394
00395     // Wait for reception complete.
00396     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00397     {
00398         asm("NOP") ;
00399     }
00400
00401     // Return data register.
00402     return A_SPIx->DR ;
00403
00404 }
00405
00406 /*****
00407 *****/
00408
00409 void MSPI_vSlaveTransmit( SPI_MemoryMapType * A_SPIx, u16_t A_u16Data )
00410 {
00411
00412     // Start transmission.
00413     SPI1->DR = A_u16Data ;
00414
00415     // Wait for transmission complete.
00416     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED )
00417     {
00418         asm("NOP") ;
00419     }
00420
00421     // Wait for the last transmission bit.
00422     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET )
00423     {
00424         asm("NOP") ;
00425     }
00426
00427 }
00428
00429 /*****
00430 *****/

```

```

00431
00432 u16_t MSPI_u16SlaveRecieve( SPI_MemoryMapType * A_SPIx )
00433 {
00434
00435     // Wait for reception complete.
00436     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED )
00437     {
00438         asm("NOP") ;
00439     }
00440
00441     // Return data register.
00442     return A_SPIx->DR ;
00443
00444 }
00445
00446 /*****
00447 *****/
00448
00449 void MSPI_vDISABLE( SPI_MemoryMapType * A_SPIx, u8_t A_u8RelationShip, u8_t A_u8DataDirection )
00450 {
00451
00452     switch( A_u8DataDirection )
00453     {
00454
00455         case SPIx_FULL_DUPLEX:
00456
00457             // Wait for reception complete.
00458             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00459
00460             // Wait for transmission complete.
00461             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00462
00463             // Wait for the last transmission bit.
00464             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00465
00466             // Disabling the SPIx.
00467             CLR_BIT( A_SPIx->CR1, SPE ) ;
00468
00469         break ;
00470
00471
00472         case SPIx_SIMPLEX      :
00473
00474 #if SIMPLEX_MODE == SIMPLEX_TX
00475
00476             // Wait for transmission complete.
00477             while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00478
00479             // Wait for the last transmission bit.
00480             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00481
00482             // Disabling the SPIx.
00483             CLR_BIT( A_SPIx->CR1, SPE ) ;
00484
00485 #elif SIMPLEX_MODE == SIMPLEX_RX
00486
00487             switch( A_u8RelationShip )
00488             {
00489
00490                 case SPIx_MSTR:
00491
00492                     // Wait for reception complete.
00493                     while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00494
00495                     // Disabling the SPIx.
00496                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00497
00498                 break ;
00499
00500                 case SPIx_SLAVE:
00501
00502                     // Disabling the SPIx.
00503                     CLR_BIT( A_SPIx->CR1, SPE ) ;
00504
00505                     // Wait for the last transmission bit.
00506                     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00507
00508                 break ;
00509
00510             }
00511
00512 #endif
00513
00514         break ;
00515
00516         case SPIx_HALF_DUPLEX:
00517

```

```

00518
00519 #if HALF_DUPLEX_MODE == HALF_DUPLEX_TX
00520
00521     // Wait for transmission complete.
00522     while( GET_BIT( A_SPIx->SR, TXE ) == FLAG_CLEARED ) ;
00523
00524     // Wait for the last transmission bit.
00525     while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00526
00527     // Disabling the SPIx.
00528     CLR_BIT( A_SPIx->CR1, SPE ) ;
00529
00530 #elif HALF_DUPLEX_MODE == HALF_DUPLEX_RX
00531
00532     switch( A_u8Relationship )
00533     {
00534
00535         case SPIx_MSTR:
00536
00537             // Wait for reception complete.
00538             while( GET_BIT( A_SPIx->SR, RXNE ) == FLAG_CLEARED ) ;
00539
00540             // Disabling the SPIx.
00541             CLR_BIT( A_SPIx->CR1, SPE ) ;
00542
00543             break ;
00544
00545         case SPIx_SLAVE:
00546
00547             // Disabling the SPIx.
00548             CLR_BIT( A_SPIx->CR1, SPE ) ;
00549
00550             // Wait for the last transmission bit.
00551             while( GET_BIT( A_SPIx->SR, BSY ) == FLAG_SET ) ;
00552
00553             break ;
00554
00555     }
00556
00557 #endif
00558
00559     break ;
00560
00561 }
00562
00563 if( A_SPIx == SPI1 )
00564 {
00565     MRCC_vDisablePeriphralCLK( RCC_APB2, APB2ENR_SPI1EN ) ;
00566 }
00567 else if( A_SPIx == SPI2 )
00568 {
00569     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI2EN ) ;
00570 }
00571 else if( A_SPIx == SPI3 )
00572 {
00573     MRCC_vDisablePeriphralCLK( RCC_APB1, APB1ENR_SPI3EN ) ;
00574 }
00575
00576 }
00577
00578 /*****
00579 *****/
00580
00581 void MSPI1_vSetCallBack( void (*Fptr) (void) )
00582 {
00583     MSPI1_CallBack = Fptr ;
00584 }
00585
00586 /*****
00587 *****/
00588
00589 void MSPI2_vSetCallBack( void (*Fptr) (void) )
00590 {
00591     MSPI2_CallBack = Fptr ;
00592 }
00593
00594 /*****
00595 *****/
00596
00597 void MSPI3_vSetCallBack( void (*Fptr) (void) )
00598 {
00599     MSPI3_CallBack = Fptr ;
00600 }
00601
00602 /*****
00603 *****/
00604

```

```

00605 void SPI1_IRQHandler(void)
00606 {
00607     // CLEAR FLAGS:
00608     SPI1->SR = SR_RESET ;
00610
00611     MSP1_Callback() ;
00612
00613 }
00614
00615 /*****
00616 ****/
00617
00618 void SPI2_IRQHandler(void)
00619 {
00620
00621     // CLEAR FLAGS:
00622     SPI2->SR = SR_RESET ;
00623
00624     MSP2_Callback() ;
00625
00626 }
00627
00628 /*****
00629 ****/
00630
00631 void SPI3_IRQHandler(void)
00632 {
00633
00634     // CLEAR FLAGS:
00635     SPI3->SR = SR_RESET ;
00636
00637     MSP3_Callback() ;
00638
00639 }
00640
00641 /*****
00642 ****/
00643
00644
00645
00646
00647
00648
00649
00650

```

7.106 COTS/MCAL/SysTick/SysTick_config.h File Reference

Macros

- #define CLK_SOURCE AHB_DividedBy8
- #define Exception_Request Dont AssertRequest
- #define SINGLE_INTERVAL_MODE (1)
Single interval mode.
- #define PERIODIC_INTERVAL_MODE (2)
Periodic interval mode.
- #define MAX_TICKS 16777216

7.106.1 Macro Definition Documentation

7.106.1.1 Exception_Request

#define Exception_Request Dont AssertRequest

Options: Dont AssertRequest AssertRequest

Definition at line 38 of file [SysTick_config.h](#).

7.106.1.2 MAX_TICKS

```
#define MAX_TICKS 16777216
```

Definition at line 61 of file [SysTick_config.h](#).

7.107 SysTick_config.h

[Go to the documentation of this file.](#)

```
00001 /* FILENAME: SysTick_config
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Sun 09/04/2022
00005 */
00006 #ifndef _SysTick_config_H
00007 #define _SysTick_config_H
00008
00009 /*****
00010 // SysTick Configurations //
00011 *****/
00012
00024 /*options:
00025 *AHB_DividedBy8
00026 *AHB
00027 */
00028 #define CLK_SOURCE AHB_DividedBy8
00031 *****/
00032
00038 #define Exception_Request Dont_AassertRequest
00039
00040 *****/
00041
00052 #define SINGLE_INTERVAL_MODE (1)
00057 #define PERIODIC_INTERVAL_MODE (2)
00061 #define MAX_TICKS 16777216
00062
00063
00064 #endif //_SysTick_config_H
```

7.108 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)`
- `#define SEC (3)`

Delay in seconds.

Functions

- void [MSysTick_vInit](#) (void)
Initialize the Systick module.
- void [MSysTick_vSetBusyWait](#) (VAR(u32_t) A_u32Ticks)
Synchronous delay function.
- void [MSysTick_vDelay](#) (VAR(u32_t) A_u32Ticks, VAR(u32_t) A_u32TickType)
Synchronous delay function.
- void [MSysTick_vDelayMicroSec](#) (VAR(u32_t) A_u32Ticks)
Synchronous delay function.
- void [MSysTick_vDelayMilliSec](#) (VAR(u32_t) A_u32Ticks)
Synchronous delay function.
- void [MSysTick_vDelaySec](#) (VAR(u32_t) A_u32Ticks)
Synchronous delay function.
- void [MSysTick_vSetSingleInterval](#) (VAR(u32_t) A_u32Ticks, void(*A_Fptr)(void))
Asynchronous 1-cycle delay function.
- void [MSysTick_vSetPeriodicInterval](#) (VAR(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.

7.108.1 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](#).

7.108.2 Macro Definition Documentation

7.108.2.1 BUSY_TICK_TIME

```
#define BUSY_TICK_TIME (1000)
```

Set the time you want to count when using the busy wait function.

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

7.108.2.2 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 119 of file [SysTick_interface.h](#).

7.108.2.3 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 125 of file [SysTick_interface.h](#).

7.108.3 Function Documentation

7.108.3.1 MSysTick_vInit()

```
void MSysTick_vInit (
    void )
```

Initialize the Systick module.

Definition at line 35 of file [SysTick_program.c](#).

```
00036 {
00037
00038     // Reset timer value.
00039     SysTick->VAL = INITIAL_ZERO;
00040
00041     // Choose the input CLK source.
00042 #if CLK_SOURCE == AHB_DividedBy8
00043     CLR_BIT(SysTick->CTRL, CLKSOURCE);
00044
00045 #elif CLK_SOURCE == AHB
00046     SET_BIT(SysTick->CTRL, CLKSOURCE);
00047 #endif
00048
00049     // SysTick exception request enable.
00050 #if Exception_Request == Dont AssertRequest
00051     CLR_BIT( SysTick->CTRL, TICKINT ) ;
00052
00053 #elif Exception_Request == AssertRequest
00054     SET_BIT( SysTick->CTRL, TICKINT ) ;
00055 #endif
00056
00057 }
```

References [CLKSOURCE](#), [CLR_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

Referenced by [LCD_vInit\(\)](#).

7.108.3.2 MSysTick_vSetBusyWait()

```
void MSysTick_vSetBusyWait (
    VAR(u32_t) A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay for
----	-------------------	------------------------------

7.108.3.3 MSysTick_vDelay()

```
void MSysTick_vDelay (
    VAR(u32_t) A_u32Ticks,
    VAR(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

7.108.3.4 MSysTick_vDelayMicroSec()

```
void MSysTick_vDelayMicroSec (
    VAR(u32_t) A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay in micro seconds for
----	-------------------	---

7.108.3.5 MSysTick_vDelayMilliSec()

```
void MSysTick_vDelayMilliSec (
    VAR(u32_t) A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay in milli seconds for
----	-------------------	---

Referenced by [HLCD_vClear\(\)](#), [HLCD_vDispCursorWithBlinking\(\)](#), [HLCD_vDispShiftLeftString\(\)](#), [HLCD_vInit\(\)](#), [HLCD_vSendCommand\(\)](#), [HLCD_vSendData\(\)](#), [HLCD_vSetCursorBlinkingOFF\(\)](#), [HLCD_vSetDispOFF\(\)](#), and [HLCD_vSetShiftLeftOn\(\)](#).

7.108.3.6 MSysTick_vDelaySec()

```
void MSysTick_vDelaySec (
    VAR(u32_t) A_u32Ticks )
```

Synchronous delay function.

Parameters

in	<i>A_u32Ticks</i>	Number of ticks to delay in seconds for
----	-------------------	---

7.108.3.7 MSysTick_vSetSingleInterval()

```
void MSysTick_vSetSingleInterval (
    VAR(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	A_u32Ticks	Number of ticks to delay for
in	A_Fptr	Callback function to call when the countdown has finished

7.108.3.8 MSysTick_vSetPeriodicInterval()

```
void MSysTick_vSetPeriodicInterval (
    VAR(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	A_u32Ticks	Number of ticks to delay for
in	A_Fptr	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

7.108.3.9 MSysTick_vStopInterval()

```
void MSysTick_vStopInterval (
    void )
```

Stop the interval delay function.

See also

[MSysTick_vSetPeriodicInterval](#) for a interval delay function

Definition at line 302 of file [SysTick_program.c](#).

```
00303 {
00304
00305     // Disable the IRQ.
00306     SET_BIT( SysTick->CTRL, TICKINT ) ;
00307
00308     // Stop the timer.
00309     SET_BIT( SysTick->CTRL, TICKINT ) ;
00310
00311     // Clear the LOAD and VAL registers
00312     SysTick->LOAD = INITIAL_ZERO;
00313     SysTick->VAL = INITIAL_ZERO;
00314
00315 }
```

References [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.108.3.10 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 320 of file [SysTick_program.c](#).

```
00321 {
00322     VAR(u32_t) L_u32ElapsedTime = INITIAL_ZERO;
00323
00324     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);
00325
00326     return L_u32ElapsedTime;
00327 }
```

References [INITIAL_ZERO](#), [SysTick](#), and [VAR](#).

7.108.3.11 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 332 of file [SysTick_program.c](#).

```
00333 {
00334     VAR(u32_t) L_u32RemainingTime = INITIAL_ZERO;
00335
00336     L_u32RemainingTime = SysTick->VAL;
00337
00338     return L_u32RemainingTime;
00339 }
```

References [INITIAL_ZERO](#), [SysTick](#), and [VAR](#).

7.108.3.12 MSysTick_vEnable()

```
void MSysTick_vEnable (
    void )
```

Enable the Systick timer.

Definition at line 344 of file [SysTick_program.c](#).

```
00345 {
00346
00347     // Start Timer.
00348     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00349
00350 }
```

References [COUNTER_ENABLE](#), [SET_BIT](#), and [SysTick](#).

7.108.3.13 MSysTick_vDisable()

```
void MSysTick_vDisable (
    void )
```

Disable the Systick timer.

Definition at line 355 of file [SysTick_program.c](#).

```
00356 {
00357
00358     // Disable the peripheral interrupt.
00359     SET_BIT( SysTick->CTRL, TICKINT );
00360
00361     // Stop the timer.
00362     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00363
00364 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.108.3.14 MSysTick_vEnableException()

```
void MSysTick_vEnableException (
    void )
```

Enable the Systick Exception.

Definition at line 27 of file [SysTick_program.c](#).

```
00028 {
00029     SET_BIT(SysTick->CTRL, TICKINT);
00030 }
```

References [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.109 SysTick_interface.h

[Go to the documentation of this file.](#)

```

00001
00009 /* Header file guard */
0010 #ifndef _SysTick_interface_H
0011 #define _SysTick_interface_H
0012
0013 /*****
0014 *          Functions prototypes
0015 *****/
0016
0020 void MSysTick_vInit(void);
0021
0026 void MSysTick_vSetBusyWait(VAR(u32_t) A_u32Ticks);
0027
0033 void MSysTick_vDelay(VAR(u32_t) A_u32Ticks, VAR(u32_t) A_u32TickType);
0034
0039 void MSysTick_vDelayMicroSec(VAR(u32_t) A_u32Ticks);
0040
0045 void MSysTick_vDelayMilliSec(VAR(u32_t) A_u32Ticks);
0046
0051 void MSysTick_vDelaySec(VAR(u32_t) A_u32Ticks);
0052
0059 void MSysTick_vSetSingleInterval(VAR(u32_t) A_u32Ticks, void (*A_Fptr)(void));
0060
0070 void MSysTick_vSetPeriodicInterval(VAR(u32_t) A_u32Ticks, void (*A_Fptr)(void));
0071
0076 void MSysTick_vStopInterval(void);
0077
0082 u32_t MSysTick_u32GetElapsedTime(void);
0083
0088 u32_t MSysTick_u32GetRemainingTime(void);
0089
0093 void MSysTick_vEnable(void);
0094
0098 void MSysTick_vDisable(void);
0099
0103 void MSysTick_vEnableException(void);
0104
0105 /*****
0106 *          Interfacing macros
0107 *****/
0108
0113 #define BUSY_TICK_TIME (1000)
0114
0119 #define SINGLE_INTERVAL_TICK_TIME (1000)
0120
0125 #define PERIODIC_INTERVAL_TICK_TIME (1000)
0126
0137 #define MILLI_SEC (1)
0138
0143 #define MICRO_SEC (2)
0144
0149 #define SEC (3)
0152#endif // _SysTick_interface_H

```

7.110 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)

7.110.1 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](#).

7.111 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

```

7.112 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_vEnableException](#) (void)
Enable the Systick Exception.
- void [MSysTick_vInit](#) (void)
Initialize the Systick module.
- void [MSysTick_vSetBusyWait](#) (u32_t A_u32Ticks)
- void [MSysTick_vDelay](#) (u32_t A_u32Ticks, u32_t A_u32TickType)
- void [MSysTick_vDelayMicroSec](#) (u32_t A_u32Ticks)
- void [MSysTick_vDelayMilliSec](#) (u32_t A_u32Ticks)
- void [MSysTick_vDelaySec](#) (u32_t A_u32Ticks)
- void [MSysTick_vSetSingleInterval](#) (u32_t A_u32Ticks, void(*A_Fptr)(void))
- void [MSysTick_vSetPeriodicInterval](#) (u32_t A_u32Ticks, void(*A_Fptr)(void))
- 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 [SysTick_Handler](#) (void)

7.112.1 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](#).

7.112.2 Function Documentation

7.112.2.1 MSysTick_vEnableException()

```
void MSysTick_vEnableException (
    void )
```

Enable the Systick Exception.

Definition at line 27 of file [SysTick_program.c](#).

```
00028 {
00029     SET_BIT(SysTick->CTRL, TICKINT);
00030 }
```

References [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.112.2.2 MSysTick_vInit()

```
void MSysTick_vInit (
    void )
```

Initialize the Systick module.

Definition at line 35 of file [SysTick_program.c](#).

```
00036 {
00037
00038     // Reset timer value.
00039     SysTick->VAL = INITIAL_ZERO;
00040
00041     // Choose the input CLK source.
00042 #if CLK_SOURCE == AHB_DividedBy8
00043     CLR_BIT(SysTick->CTRL, CLKSOURCE);
00044
00045 #elif CLK_SOURCE == AHB
00046     SET_BIT(SysTick->CTRL, CLKSOURCE);
00047 #endif
00048
00049     // SysTick exception request enable.
00050 #if Exception_Request == Dont AssertRequest
00051     CLR_BIT( SysTick->CTRL, TICKINT ) ;
00052
00053 #elif Exception_Request == AssertRequest
00054     SET_BIT( SysTick->CTRL, TICKINT ) ;
00055 #endif
00056
00057 }
```

References [CLKSOURCE](#), [CLR_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

Referenced by [HLCD_vInit\(\)](#).

7.112.2.3 MSysTick_vSetBusyWait()

```
void MSysTick_vSetBusyWait (
    u32_t A_u32Ticks )
```

Definition at line 62 of file [SysTick_program.c](#).

```
00063 {
00064
00065     // Load Ticks to the load register.
00066     SysTick->LOAD = A_u32Ticks;
00067
00068     // Reset timer value.
00069     SysTick->VAL = INITIAL_ZERO;
00070
00071     // Start Timer.
00072     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00073
00074     // Wait till the flag is raised (= 1).
00075     while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00076
00077     // Stop the timer.
00078     CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00079
00080     // Clear the LOAD and VAL registers
00081     SysTick->LOAD = INITIAL_ZERO;
00082     SysTick->VAL = INITIAL_ZERO;
00083
00084 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [SET_BIT](#), and [SysTick](#).

7.112.2.4 MSysTick_vDelay()

```
void MSysTick_vDelay (
    u32_t A_u32Ticks,
    u32_t A_u32TickType )
```

Definition at line 88 of file [SysTick_program.c](#).

```
00089 {
00090
00091     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00092
00093     if( A_u32TickType == MILLI_SEC )
00094     {
00095         l_u32TickNum = (A_u32Ticks * 1000) - 1 ;
00096     }
00097
00098     else if( A_u32TickType == MICRO_SEC )
00099     {
00100         l_u32TickNum = A_u32Ticks - 1 ;
00101     }
00102
00103     else if( A_u32TickType == SEC )
00104     {
00105         l_u32TickNum = (A_u32Ticks * 1000000) - 1 ;
00106     }
00107
00108     else
00109     {
00110         // error
00111     }
00112
00113     if (l_u32TickNum < MAX_TICKS)
00114     {
00115
00116         // Load Ticks to the load register.
00117         SysTick->LOAD = l_u32TickNum;
00118
00119         // Reset timer value.
00120         SysTick->VAL = INITIAL_ZERO;
00121
00122         // Start Timer.
```

```

00123     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00124
00125     // Wait till the flag is raised (= 1).
00126     while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00127
00128     // Stop the timer.
00129     CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00130
00131     // Clear the LOAD and VAL registers
00132     SysTick->LOAD = INITIAL_ZERO;
00133     SysTick->VAL = INITIAL_ZERO;
00134
00135 }
00136
00137 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [MICRO_SEC](#), [MILLI_SEC](#), [SEC](#), [SET_BIT](#), [SysTick](#), and [VAR](#).

7.112.2.5 MSysTick_vDelayMicroSec()

```

void MSysTick_vDelayMicroSec (
    u32_t A_u32Ticks )
```

Definition at line 142 of file [SysTick_program.c](#).

```

00143 {
00144
00145     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00146
00147     l_u32TickNum = (A_u32Ticks) - 1 ;
00148
00149     if (l_u32TickNum < MAX_TICKS)
00150     {
00151
00152         // Load Ticks to the load register.
00153         SysTick->LOAD = l_u32TickNum;
00154
00155         // Reset timer value.
00156         SysTick->VAL = INITIAL_ZERO;
00157
00158         // Start Timer.
00159         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00160
00161         // Wait till the flag is raised (= 1).
00162         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00163
00164         // Stop the timer.
00165         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00166
00167         // Clear the LOAD and VAL registers
00168         SysTick->LOAD = INITIAL_ZERO;
00169         SysTick->VAL = INITIAL_ZERO;
00170
00171     }
00172
00173 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), [SysTick](#), and [VAR](#).

7.112.2.6 MSysTick_vDelayMilliSec()

```

void MSysTick_vDelayMilliSec (
    u32_t A_u32Ticks )
```

Definition at line 178 of file [SysTick_program.c](#).

```

00179 {
00180     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00182
00183     l_u32TickNum = (A_u32Ticks * 1000) - 1 ;
00184
00185     if (l_u32TickNum < MAX_TICKS)
00186     {
00187
00188         // Load Ticks to the load register.
00189         SysTick->LOAD = l_u32TickNum;
00190
00191         // Reset timer value.
00192         SysTick->VAL = INITIAL_ZERO;
00193
00194         // Start Timer.
00195         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00196
00197         // Wait till the flag is raised (= 1).
00198         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00199
00200         // Stop the timer.
00201         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00202
00203         // Clear the LOAD and VAL registers
00204         SysTick->LOAD = INITIAL_ZERO;
00205         SysTick->VAL = INITIAL_ZERO;
00206
00207     }
00208
00209 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), [SysTick](#), and [VAR](#).

7.112.2.7 MSysTick_vDelaySec()

```
void MSysTick_vDelaySec (
    u32_t A_u32Ticks )
```

Definition at line 214 of file [SysTick_program.c](#).

```

00215 {
00216
00217     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00218
00219     l_u32TickNum = (A_u32Ticks * 1000000) - 1 ;
00220
00221     if (l_u32TickNum < MAX_TICKS)
00222     {
00223
00224         // Load Ticks to the load register.
00225         SysTick->LOAD = l_u32TickNum;
00226
00227         // Reset timer value.
00228         SysTick->VAL = INITIAL_ZERO;
00229
00230         // Start Timer.
00231         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00232
00233         // Wait till the flag is raised (= 1).
00234         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00235
00236         // Stop the timer.
00237         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00238
00239         // Clear the LOAD and VAL registers
00240         SysTick->LOAD = INITIAL_ZERO;
00241         SysTick->VAL = INITIAL_ZERO;
00242
00243     }
00244
00245 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [FLAG_CLEARED](#), [GET_BIT](#), [INITIAL_ZERO](#), [MAX_TICKS](#), [SET_BIT](#), [SysTick](#), and [VAR](#).

7.112.2.8 MSysTick_vSetSingleInterval()

```
void MSysTick_vSetSingleInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Definition at line 250 of file [SysTick_program.c](#).

```
00251 {
00252
00253     // Save the callback.
00254     GS_vCallbackFunc = A_Fptr ;
00255
00256     // Reset timer value.
00257     SysTick->VAL = INITIAL_ZERO ;
00258
00259     // Load Ticks to the load register.
00260     SysTick->LOAD = A_u32Ticks ;
00261
00262     // Start Timer.
00263     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00264
00265     // Set interval mode to single.
00266     GS_u8MyIntervalMode = SINGLE_INTERVAL_MODE ;
00267
00268     // Enable the IRQ.
00269     SET_BIT( SysTick->CTRL, TICKINT ) ;
00270
00271 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [SET_BIT](#), [SINGLE_INTERVAL_MODE](#), [SysTick](#), and [TICKINT](#).

7.112.2.9 MSysTick_vSetPeriodicInterval()

```
void MSysTick_vSetPeriodicInterval (
    u32_t A_u32Ticks,
    void(*)(void) A_Fptr )
```

Definition at line 276 of file [SysTick_program.c](#).

```
00277 {
00278
00279     // Save the callback.
00280     GS_vCallbackFunc = A_Fptr;
00281
00282     // Reset timer value.
00283     SysTick->VAL = INITIAL_ZERO ;
00284
00285     // Load Ticks to the load register.
00286     SysTick->LOAD = A_u32Ticks;
00287
00288     // Start Timer.
00289     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00290
00291     // Set interval mode to single.
00292     GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00293
00294     // Enable the IRQ.
00295     SET_BIT( SysTick->CTRL, TICKINT ) ;
00296
00297 }
```

References [COUNTER_ENABLE](#), [INITIAL_ZERO](#), [PERIODIC_INTERVAL_MODE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.112.2.10 MSysTick_vStopInterval()

```
void MSysTick_vStopInterval (
    void )
```

Stop the interval delay function.

See also

[MSysTick_vSetPeriodicInterval](#) for a interval delay function

Definition at line 302 of file [SysTick_program.c](#).

```
00303 {
00304
00305     // Disable the IRQ.
00306     SET_BIT( SysTick->CTRL, TICKINT ) ;
00307
00308     // Stop the timer.
00309     SET_BIT( SysTick->CTRL, TICKINT ) ;
00310
00311     // Clear the LOAD and VAL registers
00312     SysTick->LOAD = INITIAL_ZERO;
00313     SysTick->VAL = INITIAL_ZERO;
00314
00315 }
```

References [INITIAL_ZERO](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.112.2.11 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 320 of file [SysTick_program.c](#).

```
00321 {
00322     VAR(u32_t) L_u32ElapsedTime = INITIAL_ZERO;
00323
00324     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);
00325
00326     return L_u32ElapsedTime;
00327 }
```

References [INITIAL_ZERO](#), [SysTick](#), and [VAR](#).

7.112.2.12 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 332 of file [SysTick_program.c](#).

```
00333 {
00334     VAR(u32_t) L_u32RemainingTime = INITIAL_ZERO;
00335
00336     L_u32RemainingTime = SysTick->VAL;
00337
00338     return L_u32RemainingTime;
00339 }
```

References [INITIAL_ZERO](#), [SysTick](#), and [VAR](#).

7.112.2.13 MSysTick_vEnable()

```
void MSysTick_vEnable (
    void )
```

Enable the Systick timer.

Definition at line 344 of file [SysTick_program.c](#).

```
00345 {
00346
00347     // Start Timer.
00348     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);
00349
00350 }
```

References [COUNTER_ENABLE](#), [SET_BIT](#), and [SysTick](#).

7.112.2.14 MSysTick_vDisable()

```
void MSysTick_vDisable (
    void )
```

Disable the Systick timer.

Definition at line 355 of file [SysTick_program.c](#).

```
00356 {
00357
00358     // Disable the peripheral interrupt.
00359     SET_BIT( SysTick->CTRL, TICKINT );
00360
00361     // Stop the timer.
00362     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00363
00364 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [SET_BIT](#), [SysTick](#), and [TICKINT](#).

7.112.2.15 SysTick_Handler()

```
void SysTick_Handler (
    void )
```

Definition at line 369 of file [SysTick_program.c](#).

```
00370 {
00371     volatile VAR(u8_t) L_u8ReadFlag = INITIAL_ZERO;
00373
00374     if (GS_vCallbackFunc != NULL)
00375     {
00376         // Callback notification.
00377         GS_vCallbackFunc();
00378     }
00379
00380     if (GS_u8MyIntervalMode == SINGLE_INTERVAL_MODE)
00381     {
00382         // Disable the IRQ.
00383         CLR_BIT( SysTick->CTRL, TICKINT ) ;
00384
00385         // Stop the timer.
00386         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00387
00388         // Clear the LOAD and VAL registers
00389         SysTick->LOAD = INITIAL_ZERO;
00390         SysTick->VAL = INITIAL_ZERO;
00391
00392         GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00393     }
00394
00395     // Clear IRQ flag.
00396     L_u8ReadFlag = GET_BIT(SysTick->CTRL, COUNTFLAG);
00397
00398 }
```

References [CLR_BIT](#), [COUNTER_ENABLE](#), [COUNTFLAG](#), [GET_BIT](#), [INITIAL_ZERO](#), [NULL](#), [PERIODIC_INTERVAL_MODE](#), [SINGLE_INTERVAL_MODE](#), [SysTick](#), [TICKINT](#), and [VAR](#).

7.113 SysTick_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 #include "SysTick_interface.h"
0017 #include "SysTick_private.h"
0018 #include "SysTick_config.h"
0019
0020 STATIC P2FUNC(VAR(void), GS_vCallbackFunc) (void) = NULL ;
0021
0022 STATIC VAR(u8_t) GS_u8MyIntervalMode = INITIAL_ZERO ;
0023
0024
0025
0026
0027 FUNC(void) MSysTick_vEnableException(void)
0028 {
0029     SET_BIT(SysTick->CTRL, TICKINT);
0030 }
0031
0032
0033
0034
0035 FUNC(void) MSysTick_vInit(void)
0036 {
0037 }
```

```

00038     // Reset timer value.
00039     SysTick->VAL = INITIAL_ZERO;
00040
00041     // Choose the input CLK source.
00042 #if CLK_SOURCE == AHB_DividedBy8
00043     CLR_BIT(SysTick->CTRL, CLKSOURCE);
00044
00045 #elif CLK_SOURCE == AHB
00046     SET_BIT(SysTick->CTRL, CLKSOURCE);
00047 #endif
00048
00049     // SysTick exception request enable.
00050 #if Exception_Request == Dont AssertRequest
00051     CLR_BIT( SysTick->CTRL, TICKINT ) ;
00052
00053 #elif Exception_Request == AssertRequest
00054     SET_BIT( SysTick->CTRL, TICKINT ) ;
00055 #endif
00056
00057 }
00058
00059
00060 /******
00061 *****/
00062 FUNC(void) MSysTick_vSetBusyWait(VAR(u32_t) A_u32Ticks)
00063 {
00064
00065     // Load Ticks to the load register.
00066     SysTick->LOAD = A_u32Ticks;
00067
00068     // Reset timer value.
00069     SysTick->VAL = INITIAL_ZERO;
00070
00071     // Start Timer.
00072     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00073
00074     // Wait till the flag is raised (= 1).
00075     while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00076
00077     // Stop the timer.
00078     CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00079
00080     // Clear the LOAD and VAL registers
00081     SysTick->LOAD = INITIAL_ZERO;
00082     SysTick->VAL = INITIAL_ZERO;
00083
00084 }
00085
00086
00087 /******
00088 *****/
00088 FUNC(void) MSysTick_vDelay(VAR(u32_t) A_u32Ticks, VAR(u32_t) A_u32TickType )
00089 {
00090
00091     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00092
00093     if( A_u32TickType == MILLI_SEC )
00094     {
00095         l_u32TickNum = (A_u32Ticks * 1000) - 1 ;
00096     }
00097
00098     else if( A_u32TickType == MICRO_SEC )
00099     {
00100         l_u32TickNum = A_u32Ticks - 1 ;
00101     }
00102
00103     else if( A_u32TickType == SEC )
00104     {
00105         l_u32TickNum = (A_u32Ticks * 1000000) - 1 ;
00106     }
00107
00108     else
00109     {
00110         // error
00111     }
00112
00113     if (l_u32TickNum < MAX_TICKS)
00114     {
00115
00116         // Load Ticks to the load register.
00117         SysTick->LOAD = l_u32TickNum;
00118
00119         // Reset timer value.
00120         SysTick->VAL = INITIAL_ZERO;

```

```

00121     // Start Timer.
00122     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00123
00124     // Wait till the flag is raised (= 1).
00125     while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00126
00127     // Stop the timer.
00128     CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00129
00130     // Clear the LOAD and VAL registers
00131     SysTick->LOAD = INITIAL_ZERO;
00132     SysTick->VAL = INITIAL_ZERO;
00133
00134 }
00135
00136 }
00137 }
00138
00139
00140
00141
00142 FUNC(void) MSysTick_vDelayMicroSec( VAR(u32_t) A_u32Ticks )
00143 {
00144
00145     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00146
00147     l_u32TickNum = (A_u32Ticks) - 1 ;
00148
00149     if (l_u32TickNum < MAX_TICKS)
00150     {
00151
00152         // Load Ticks to the load register.
00153         SysTick->LOAD = l_u32TickNum;
00154
00155         // Reset timer value.
00156         SysTick->VAL = INITIAL_ZERO;
00157
00158         // Start Timer.
00159         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00160
00161         // Wait till the flag is raised (= 1).
00162         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00163
00164         // Stop the timer.
00165         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00166
00167         // Clear the LOAD and VAL registers
00168         SysTick->LOAD = INITIAL_ZERO;
00169         SysTick->VAL = INITIAL_ZERO;
00170
00171     }
00172
00173 }
00174
00175
00176
00177
00178 FUNC(void) MSysTick_vDelayMilliSec( VAR(u32_t) A_u32Ticks )
00179 {
00180
00181     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00182
00183     l_u32TickNum = (A_u32Ticks * 1000) - 1 ;
00184
00185     if (l_u32TickNum < MAX_TICKS)
00186     {
00187
00188         // Load Ticks to the load register.
00189         SysTick->LOAD = l_u32TickNum;
00190
00191         // Reset timer value.
00192         SysTick->VAL = INITIAL_ZERO;
00193
00194         // Start Timer.
00195         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00196
00197         // Wait till the flag is raised (= 1).
00198         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00199
00200         // Stop the timer.
00201         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00202
00203         // Clear the LOAD and VAL registers

```

```

00204     SysTick->LOAD = INITIAL_ZERO;
00205     SysTick->VAL = INITIAL_ZERO;
00206
00207 }
00208
00209 }
00210
00211 //*****
00212 //*****
00213
00214 FUNC(void) MSysTick_vDelaySec( VAR(u32_t) A_u32Ticks )
00215 {
00216
00217     VAR(u32_t) l_u32TickNum = INITIAL_ZERO ;
00218
00219     l_u32TickNum = (A_u32Ticks * 1000000) - 1 ;
00220
00221     if (l_u32TickNum < MAX_TICKS)
00222     {
00223
00224         // Load Ticks to the load register.
00225         SysTick->LOAD = l_u32TickNum;
00226
00227         // Reset timer value.
00228         SysTick->VAL = INITIAL_ZERO;
00229
00230         // Start Timer.
00231         SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00232
00233         // Wait till the flag is raised (= 1).
00234         while( GET_BIT(SysTick->CTRL, COUNTFLAG) == FLAG_CLEARED ) ;
00235
00236         // Stop the timer.
00237         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00238
00239         // Clear the LOAD and VAL registers
00240         SysTick->LOAD = INITIAL_ZERO;
00241         SysTick->VAL = INITIAL_ZERO;
00242
00243     }
00244
00245 }
00246
00247 //*****
00248 //*****
00249
00250 FUNC(void) MSysTick_vSetSingleInterval(VAR(u32_t) A_u32Ticks, P2FUNC(VAR(void), A_Fptr)(void))
00251 {
00252
00253     // Save the callback.
00254     GS_vCallbackFunc = A_Fptr ;
00255
00256     // Reset timer value.
00257     SysTick->VAL = INITIAL_ZERO ;
00258
00259     // Load Ticks to the load register.
00260     SysTick->LOAD = A_u32Ticks ;
00261
00262     // Start Timer.
00263     SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00264
00265     // Set interval mode to single.
00266     GS_u8MyIntervalMode = SINGLE_INTERVAL_MODE ;
00267
00268     // Enable the IRQ.
00269     SET_BIT( SysTick->CTRL, TICKINT ) ;
00270
00271 }
00272
00273 //*****
00274 //*****
00275
00276 FUNC(void) MSysTick_vSetPeriodicInterval(VAR(u32_t) A_u32Ticks, P2FUNC(VAR(void), A_Fptr)(void))
00277 {
00278
00279     // Save the callback.
00280     GS_vCallbackFunc = A_Fptr;
00281
00282     // Reset timer value.
00283     SysTick->VAL = INITIAL_ZERO ;
00284

```

```
00285 // Load Ticks to the load register.  
00286 SysTick->LOAD = A_u32Ticks;  
00287  
00288 // Start Timer.  
00289 SET_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;  
00290  
00291 // Set interval mode to single.  
00292 GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;  
00293  
00294 // Enable the IRQ.  
00295 SET_BIT( SysTick->CTRL, TICKINT ) ;  
00296  
00297 }  
00298  
00299  
00300 //*****  
00301 //*****  
00302 FUNC(void) MSysTick_vStopInterval(void)  
00303 {  
00304  
00305 // Disable the IRQ.  
00306 SET_BIT( SysTick->CTRL, TICKINT ) ;  
00307  
00308 // Stop the timer.  
00309 SET_BIT( SysTick->CTRL, TICKINT ) ;  
00310  
00311 // Clear the LOAD and VAL registers  
00312 SysTick->LOAD = INITIAL_ZERO;  
00313 SysTick->VAL = INITIAL_ZERO;  
00314  
00315 }  
00316  
00317  
00318 //*****  
00319  
00320 VAR(u32_t) MSysTick_u32GetElapsedTime(void)  
00321 {  
00322     VAR(u32_t) L_u32ElapsedTime = INITIAL_ZERO;  
00323  
00324     L_u32ElapsedTime = (SysTick->LOAD - SysTick->VAL);  
00325  
00326     return L_u32ElapsedTime;  
00327 }  
00328  
00329  
00330 //*****  
00331  
00332 VAR(u32_t) MSysTick_u32GetRemainingTime(void)  
00333 {  
00334     VAR(u32_t) L_u32RemainingTime = INITIAL_ZERO;  
00335  
00336     L_u32RemainingTime = SysTick->VAL;  
00337  
00338     return L_u32RemainingTime;  
00339 }  
00340  
00341  
00342 //*****  
00343  
00344 FUNC(void) MSysTick_vEnable(void)  
00345 {  
00346  
00347     // Start Timer.  
00348     SET_BIT(SysTick->CTRL, COUNTER_ENABLE);  
00349  
00350 }  
00351  
00352  
00353  
00354  
00355 FUNC(void) MSysTick_vDisable(void)  
00356 {  
00357  
00358     // Disable the peripheral interrupt.  
00359     SET_BIT( SysTick->CTRL, TICKINT ) ;  
00360  
00361     // Stop the timer.
```

```

00362     CLR_BIT(SysTick->CTRL, COUNTER_ENABLE);
00363
00364 }
00365
00366 //*****
00367 //*****
00368
00369 FUNC(void) SysTick_Handler(void)
00370 {
00371
00372     volatile VAR(u8_t) L_u8ReadFlag = INITIAL_ZERO;
00373
00374     if (GS_vCallbackFunc != NULL)
00375     {
00376         // Callback notification.
00377         GS_vCallbackFunc();
00378     }
00379
00380     if (GS_u8MyIntervalMode == SINGLE_INTERVAL_MODE)
00381     {
00382         // Disable the IRQ.
00383         CLR_BIT( SysTick->CTRL, TICKINT ) ;
00384
00385         // Stop the timer.
00386         CLR_BIT( SysTick->CTRL, COUNTER_ENABLE ) ;
00387
00388         // Clear the LOAD and VAL registers
00389         SysTick->LOAD = INITIAL_ZERO;
00390         SysTick->VAL = INITIAL_ZERO;
00391
00392         GS_u8MyIntervalMode = PERIODIC_INTERVAL_MODE;
00393     }
00394
00395     // Clear IRQ flag.
00396     L_u8ReadFlag = GET_BIT(SysTick->CTRL, COUNTFLAG);
00397
00398 }
00399
00400 //*****
00401 //*****

```

7.114 COTS/MCAL/UART/UART_config.h File Reference

Macros

- #define __PCLK__ 8000000
- #define THRESHOLD_VALUE 5000000
- #define USART1_PORT GPIO_PORTB
- #define TX1_PIN GPIOx_PIN6
- #define RX1_PIN GPIOx_PIN7
- #define USART2_PORT GPIO_PORTA
- #define TX2_PIN GPIOx_PIN2
- #define RX2_PIN GPIOx_PIN3
- #define USART6_PORT GPIO_PORTA
- #define TX6_PIN GPIOx_PIN11
- #define RX6_PIN GPIOx_PIN12

7.114.1 Macro Definition Documentation

7.114.1.1 PCLK

```
#define __PCLK__ 8000000
```

Definition at line 11 of file [UART_config.h](#).

7.114.1.2 THRESHOLD_VALUE

```
#define THRESHOLD_VALUE 5000000
```

Definition at line 13 of file [UART_config.h](#).

7.114.1.3 USART1_PORT

```
#define USART1_PORT GPIO_PORTB
```

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

7.114.1.4 TX1_PIN

```
#define TX1_PIN GPIOx_PIN6
```

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

7.114.1.5 RX1_PIN

```
#define RX1_PIN GPIOx_PIN7
```

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

7.114.1.6 USART2_PORT

```
#define USART2_PORT GPIO_PORTA
```

Definition at line 48 of file [UART_config.h](#).

7.114.1.7 TX2_PIN

```
#define TX2_PIN GPIOx_PIN2
```

Definition at line 55 of file [UART_config.h](#).

7.114.1.8 RX2_PIN

```
#define RX2_PIN GPIOx_PIN3
```

Definition at line 62 of file [UART_config.h](#).

7.114.1.9 USART6_PORT

```
#define USART6_PORT GPIO_PORTA
```

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

7.114.1.10 TX6_PIN

```
#define TX6_PIN GPIOx_PIN11
```

Definition at line 78 of file [UART_config.h](#).

7.114.1.11 RX6_PIN

```
#define RX6_PIN GPIOx_PIN12
```

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

7.115 UART_config.h

[Go to the documentation of this file.](#)

```

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 /***** USART2 configurations *****/
00026
00027 /*options:
00028 *GPIOx_PIN9
00029 *GPIOx_PIN6
00030 */
00031 #define TX1_PIN GPIOx_PIN6
00032
00033 /***** USART3 configurations *****/
00034
00035 /*options:
00036 *GPIOx_PIN10
00037 *GPIOx_PIN7
00038 */
00039 #define RX1_PIN GPIOx_PIN7
00040
00041 /***** USART4 configurations *****/
00042 /*          USART4 configurations */
00043 /***** USART4 configurations *****/
00044
00045 /*options:
00046 *GPIO_PORTA
00047 */
00048 #define USART2_PORT GPIO_PORTA
00049
00050 /***** USART5 configurations *****/
00051
00052 /*options:
00053 *GPIOx_PIN2
00054 */
00055 #define TX2_PIN GPIOx_PIN2
00056
00057 /***** USART6 configurations *****/
00058
00059 /*options:
00060 *GPIOx_PIN3
00061 */
00062 #define RX2_PIN GPIOx_PIN3
00063
00064 /***** USART7 configurations *****/
00065 /*          USART7 configurations */
00066 /***** USART7 configurations *****/
00067
00068 /*options:
00069 *GPIO_PORTA
00070 */
00071 #define USART6_PORT GPIO_PORTA
00072
00073 /***** USART8 configurations *****/
00074
00075 /*options:
00076 *GPIOx_PIN2
00077 */
00078 #define TX6_PIN GPIOx_PIN11
00079
00080 /***** USART9 configurations *****/
00081
00082 /*options:

```

```

00083 *GPIOx_PIN3
00084 */
00085 #define RX6_PIN GPIOx_PIN12
00086
00087 /***** */
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

```

7.116 COTS/MCAL/UART/UART_interface.h File Reference

Data Structures

- struct `USART_MemoryMapType`
- struct `USART_InitType`
- struct `USART_ClockInitTypeDef`

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_USARTTx)
- void **MUSART_vEnable** (USART_MemoryMapType *A_USARTTx)
- void **MUSART_vDisable** (USART_MemoryMapType *A_USARTTx)
- void **MUSART_vTransmitByte** (USART_MemoryMapType *A_USARTTx, u8_t A_u8Byte)
- void **MUSART_vTransmitString** (USART_MemoryMapType *A_USARTTx, c8_t *A_ptrc8String)
- u8_t **MUSART_u8ReceiveByteSynchNonBlocking** (USART_MemoryMapType *A_USARTTx)
- u8_t **MUSART_u8ReceiveByteSynchBlocking** (USART_MemoryMapType *A_USARTTx)
- u8_t * **MUSART_ptrReceiveStringSynchNonBlocking** (USART_MemoryMapType *A_USARTTx)
- void **MUSART_vRecieveString** (USART_MemoryMapType *A_USARTTx, c8_t A_c8YourString[])
- u8_t **MUSART_u8CompareString** (c8_t *String1, c8_t *String2)
- u8_t **MUSART_u8ReadDataRegister** (USART_MemoryMapType *A_USARTTx)
- void **MUSART_vClearFlags** (USART_MemoryMapType *A_USARTTx)
- void **MUSART_vRxIntSetStatus** (USART_MemoryMapType *A_USARTTx, u8_t A_u8Status)
- void **MUSART1_vSetCallBack** (void(*Fptr)(void))
- void **MUSART2_vSetCallBack** (void(*Fptr)(void))
- void **MUSART6_vSetCallBack** (void(*Fptr)(void))

7.116.1 Macro Definition Documentation

7.116.1.1 USART1_BASE_ADDRESS

```
#define USART1_BASE_ADDRESS 0x40011000
```

Definition at line 15 of file [UART_interface.h](#).

7.116.1.2 USART2_BASE_ADDRESS

```
#define USART2_BASE_ADDRESS 0x40004400
```

Definition at line 16 of file [UART_interface.h](#).

7.116.1.3 USART6_BASE_ADDRESS

```
#define USART6_BASE_ADDRESS 0x40011400
```

Definition at line 17 of file [UART_interface.h](#).

7.116.1.4 USART1_REG

```
#define USART1_REG ( ( USART_MemoryMapType*) USART1_BASE_ADDRESS)
```

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

7.116.1.5 USART2_REG

```
#define USART2_REG ( ( USART_MemoryMapType*) USART2_BASE_ADDRESS)
```

Definition at line 40 of file [UART_interface.h](#).

7.116.1.6 USART6_REG

```
#define USART6_REG ( ( USART_MemoryMapType*) USART6_BASE_ADDRESS)
```

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

7.116.1.7 OVER_SAMPLING_16

```
#define OVER_SAMPLING_16 0
```

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

7.116.1.8 OVER_SAMPLING_8

```
#define OVER_SAMPLING_8 1
```

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

7.116.1.9 TX_ONLY

```
#define TX_ONLY 0
```

Definition at line 123 of file [UART_interface.h](#).

7.116.1.10 RX_ONLY

```
#define RX_ONLY 1
```

Definition at line 124 of file [UART_interface.h](#).

7.116.1.11 TX_RX

```
#define TX_RX 2
```

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

7.116.1.12 EVEN_PARITY

```
#define EVEN_PARITY 0
```

Definition at line 127 of file [UART_interface.h](#).

7.116.1.13 ODD_PARITY

```
#define ODD_PARITY 1
```

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

7.116.1.14 MODE_8BIT

```
#define MODE_8BIT 0
```

Definition at line 130 of file [UART_interface.h](#).

7.116.1.15 MODE_9BIT

```
#define MODE_9BIT 1
```

Definition at line 131 of file [UART_interface.h](#).

7.116.1.16 STOP_BIT_1

```
#define STOP_BIT_1 0
```

Definition at line 133 of file [UART_interface.h](#).

7.116.1.17 STOP_BIT_0_5

```
#define STOP_BIT_0_5 1
```

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

7.116.1.18 STOP_BIT_2

```
#define STOP_BIT_2 2
```

Definition at line 135 of file [UART_interface.h](#).

7.116.1.19 STOP_BIT_1_5

```
#define STOP_BIT_1_5 3
```

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

7.116.1.20 ENABLE

```
#define ENABLE 1
```

Definition at line 138 of file [UART_interface.h](#).

7.116.1.21 DISABLE

```
#define DISABLE 0
```

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

7.116.1.22 __BAUDRATE__

```
#define __BAUDRATE__ 9600
```

Definition at line 141 of file [UART_interface.h](#).

7.116.2 Function Documentation

7.116.2.1 MUSART_vInit()

```
void MUSART_vInit (
    USART_InitType * A_InitStruct,
    USART_ClockInitTypeDef * A_ClockInitStruct,
    USART_MemoryMapType * A_USARTTx )
```

Definition at line 37 of file [UART_program.c](#).

```
00039 {
00040
00041     if( A_USARTTx == USART1_REG )
00042     {
00043
00044         MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART1EN ) ;
00045
00046         /*configer Tx1 as alt fun*/
00047         MGPIox_ConfigType TX1 =
00048         {
00049             .Port          = USART1_PORT      ,   .Pin          = TX1_PIN
00050             .Mode          = GPIOx_MODE_AF   ,   .OutputType = GPIOx_PUSH_PULL,
00051             .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull ,
00052             .AF_Type       = GPIOx_AF7
00053         } ;
00054
00055         MGPIox_vInit( &TX1 ) ;
00056
00057         /*configer Rx1 as alt fun*/
00058         MGPIox_ConfigType RX1 =
00059         {
00060             .Port          = USART1_PORT      ,   .Pin          = RX1_PIN
00061             .Mode          = GPIOx_MODE_AF   ,   .OutputType = GPIOx_PUSH_PULL,
00062             .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull ,
00063             .AF_Type       = GPIOx_AF7
00064         } ;
00065
00066         MGPIox_vInit( &RX1 ) ;
00067
00068     }
00069     else if( A_USARTTx == USART2_REG )
00070     {
00071
00072         MRCC_vEnablePeriphralCLK( RCC_APB2, APB1ENR_USART2EN ) ;
00073
00074         /*configer Tx2 as alt fun*/
00075         MGPIox_ConfigType TX2 =
00076         {
00077             .Port          = USART2_PORT      ,   .Pin          = TX2_PIN
00078             .Mode          = GPIOx_MODE_AF   ,   .OutputType = GPIOx_PUSH_PULL,
00079             .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull ,
00080             .AF_Type       = GPIOx_AF7
00081         } ;
00082
00083         MGPIox_vInit( &TX2 ) ;
00084
00085         /*configer Rx2 as alt fun*/
00086         MGPIox_ConfigType RX2 =
00087         {
00088             .Port          = USART2_PORT      ,   .Pin          = RX2_PIN
00089             .Mode          = GPIOx_MODE_AF   ,   .OutputType = GPIOx_PUSH_PULL,
00090             .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull ,
00091             .AF_Type       = GPIOx_AF7
00092         } ;
```

```

00093
00094     MGPIox_vInit( &RX2 ) ;
00095
00096 }
00097 else if( A_USARTx == USART6_REG )
00098 {
00099
00100     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART6EN ) ;
00101
00102     /*configer Tx6 as alt fun*/
00103     MGPIox_ConfigType TX6 =
00104     {
00105         .Port      = USART6_PORT      ,   .Pin      = TX6_PIN      ,
00106         .Mode      = GPIOx_MODE_AF    ,   .OutputType = GPIOx_PUSH_PULL,
00107         .OutputSpeed = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull  ,
00108         .AF_Type   = GPIOx_AF7       ,
00109     } ;
00110
00111     MGPIox_vInit( &TX6 ) ;
00112
00113     /*configer Rx6 as alt fun*/
00114     MGPIox_ConfigType RX6 =
00115     {
00116         .Port      = USART6_PORT      ,   .Pin      = RX6_PIN      ,
00117         .Mode      = GPIOx_MODE_AF    ,   .OutputType = GPIOx_PUSH_PULL,
00118         .OutputSpeed = GPIOx_MediumSpeed ,   .InputType  = GPIOx_NoPull  ,
00119         .AF_Type   = GPIOx_AF7       ,
00120     } ;
00121
00122     MGPIox_vInit( &RX6 ) ;
00123
00124 }
00125
00126 // BuadRate/Over-sampling selections:
00127 switch( A_InitStruct->Oversampling )
00128 {
00129
00130     case OVER_SAMPLING_16:
00131
00132         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00133
00134     break;
00135
00136     case OVER_SAMPLING_8:
00137
00138         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00139
00140     break;
00141
00142 }
00143
00144 A_USARTx->CR1_REG = ( A_InitStruct->Oversampling << MUSART_CR1_OVER8_BIT ) | 
00145             ( A_InitStruct->DataWidth << MUSART_CR1_M_BIT ) | 
00146             ( A_InitStruct->Parity_Enable << MUSART_CR1_PCE_BIT ) | 
00147             ( A_InitStruct->Parity_Selection << MUSART_CR1_PS_BIT ) ;
00148
00149 switch (A_InitStruct->TransferDirection)
00150 {
00151     case TX_ONLY:
00152
00153         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00154
00155     break ;
00156
00157     case RX_ONLY:
00158
00159         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;
00160
00161     break ;
00162
00163     case TX_RX :
00164
00165         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00166         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;
00167
00168     break ;
00169
00170 }
00171
00172 A_USARTx->CR2_REG = ( A_InitStruct->StopBits << MUSART_CR2_STOP_BIT ) | 
00173             ( A_ClockInitStruct->ClockOutput << MUSART_CR2_CLKEN_BIT ) | 
00174             ( A_ClockInitStruct->ClockPhase << MUSART_CR2_CPHA_BIT ) | 
00175             ( A_ClockInitStruct->ClockPolarity << MUSART_CR2_CPOL_BIT ) | 
00176             ( A_ClockInitStruct->LastBitClockPulse << MUSART_CR2_LBCL_BIT ) ;
00177
00178 A_USARTx->SR_REG = 0 ; // Clear all the flags.
00179

```

```
00180     SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ; // EN the peripheral.
00181
00182 }
```

References [__PCLK__](#), [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](#), [MGPIOx_vInit\(\)](#), [MRCC_vEnablePeripheralCLK\(\)](#), [MUSART_CR1_M_BIT](#), [MUSART_CR1_OVER8_BIT](#), [MUSART_CR1_PCE_BIT](#), [MUSART_CR1_PS_BIT](#), [MUSART_CR1_RE_BIT](#), [MUSART_CR1_TE_BIT](#), [MUSART_CR1_UE_BIT](#), [MUSART_CR2_CLKEN_BIT](#), [MUSART_CR2_CPHA_BIT](#), [MUSART_CR2_CPOL_BIT](#), [MUSART_CR2_LBCL_BIT](#), [MUSART_CR2_STOP_BIT](#), [OVER_SAMPLING_16](#), [OVER_SAMPLING_8](#), [USART_InitType::Oversampling](#), [USART_InitType::Parity_Enable](#), [USART_InitType::Parity_Selection](#), [MGPIOx_ConfigType::Port](#), [RCC_APB2](#), [RX1_PIN](#), [RX2_PIN](#), [RX6_PIN](#), [RX_ONLY](#), [SET_BIT](#), [USART_MemoryMapType::SR_REG](#), [USART_InitType::StopBits](#), [USART_InitType::TransferDirection](#), [TX1_PIN](#), [TX2_PIN](#), [TX6_PIN](#), [TX_ONLY](#), [TX_RX](#), [UART_BRR_SAMPLING16](#), [USART1_PORT](#), [USART1_REG](#), [USART2_PORT](#), [USART2_REG](#), [USART6_PORT](#), and [USART6_REG](#).

Referenced by [HBluetooth_vInit\(\)](#).

7.116.2.2 MUSART_vEnable()

```
void MUSART_vEnable (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 187 of file [UART_program.c](#).

```
00188 {
00189     SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00190 }
```

References [USART_MemoryMapType::CR1_REG](#), [MUSART_CR1_UE_BIT](#), and [SET_BIT](#).

Referenced by [HBluetooth_vEnable\(\)](#).

7.116.2.3 MUSART_vDisable()

```
void MUSART_vDisable (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 195 of file [UART_program.c](#).

```
00196 {
00197     CLR_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00198 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), and [MUSART_CR1_UE_BIT](#).

Referenced by [HBluetooth_vDisable\(\)](#).

7.116.2.4 MUSART_vTransmitByte()

```
void MUSART_vTransmitByte (
    USART_MemoryMapType * A_USARTTx,
    u8_t A_u8Byte )
```

Definition at line 203 of file [UART_program.c](#).

```
00204 {
00205
00206     while( GET_BIT( A_USARTTx->SR_REG, MUSART_SR_TXE_BIT ) == 0 ) ;
00207
00208     A_USARTTx->DR_REG = A_u8Byte ;
00209
00210     while( GET_BIT( A_USARTTx->SR_REG, MUSART_SR_TC_BIT ) == 0 ) ;
00211
00212     // Clear the TC flag:
00213     CLR_BIT( A_USARTTx->SR_REG, MUSART_SR_TC_BIT ) ;
00214
00215 }
```

References [CLR_BIT](#), [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_SR_TC_BIT](#), [MUSART_SR_TXE_BIT](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [HBluetooth_vSendByte\(\)](#), and [MUSART_vTransmitString\(\)](#).

7.116.2.5 MUSART_vTransmitString()

```
void MUSART_vTransmitString (
    USART_MemoryMapType * A_USARTTx,
    c8_t * A_ptrc8String )
```

Definition at line 220 of file [UART_program.c](#).

```
00221 {
00222
00223     u8_t loc_u8Iterator = INITIAL_ZERO ;
00224
00225     while( A_ptrc8String[loc_u8Iterator] != '\0' )
00226     {
00227
00228         MUSART_vTransmitByte( A_USARTTx, A_ptrc8String[loc_u8Iterator] ) ;
00229
00230         loc_u8Iterator++ ;
00231
00232     }
00233
00234 }
```

References [INITIAL_ZERO](#), and [MUSART_vTransmitByte\(\)](#).

Referenced by [HBluetooth_vSendString\(\)](#).

7.116.2.6 MUSART_u8ReceiveByteSynchNonBlocking()

```
u8_t MUSART_u8ReceiveByteSynchNonBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 239 of file [UART_program.c](#).

```
00240 {
00241     u8_t loc_u8Data = INITIAL_ZERO ;
00242     u32_t loc_u8TimeOut = INITIAL_ZERO ;
00243
00244     while( (GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0) && (loc_u8TimeOut < THRESHOLD_VALUE) )
00245     {
00246         loc_u8TimeOut++;
00247     }
00248
00249     if ( loc_u8TimeOut == THRESHOLD_VALUE )
00250     {
00251         loc_u8Data = 255 ; // To detect the error.
00252     }
00253     else
00254     {
00255         loc_u8Data = A_USARTx->DR_REG ;
00256     }
00257
00258     return loc_u8Data ;
00259
00260 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [INITIAL_ZERO](#), [MUSART_SR_RXNE_BIT](#), [USART_MemoryMapType::SR_REG](#), and [THRESHOLD_VALUE](#).

Referenced by [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#).

7.116.2.7 MUSART_u8ReceiveByteSynchBlocking()

```
u8_t MUSART_u8ReceiveByteSynchBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 267 of file [UART_program.c](#).

```
00268 {
00269     while( GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0 ) ;
00270
00271     return A_USARTx->DR_REG ;
00272
00273 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_SR_RXNE_BIT](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [HBluetooth_u8ReceiveByte\(\)](#), and [MUSART_vRecieveString\(\)](#).

7.116.2.8 MUSART_ptrReceiveStringSynchNonBlocking()

```
u8_t * MUSART_ptrReceiveStringSynchNonBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 279 of file [UART_program.c](#).

```
00280 {
00281     u8_t loc_u8Iterator = INITIAL_ZERO ;
00282
00283     u8_t loc_u8DataCome ;
00284
00285     while( ( loc_u8DataCome = MUSART_u8ReceiveByteSynchNonBlocking(A_USARTx) ) != 13 )
00286     {
00287         G_u8String[ loc_u8Iterator ] = loc_u8DataCome ;
00288
00289         loc_u8Iterator++ ;
00290     }
00291
00292     G_u8String[loc_u8Iterator] = '\0' ;
00293
00294     return( G_u8String ) ;
00295
00296
00297 }
```

References [G_u8String](#), [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#).

7.116.2.9 MUSART_vRecieveString()

```
void MUSART_vRecieveString (
    USART_MemoryMapType * A_USARTx,
    c8_t A_c8YourString[] )
```

Definition at line 302 of file [UART_program.c](#).

```
00303 {
00304
00305     u32_t L_u32Counter = INITIAL_ZERO ;
00306
00307     A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00308
00309     while( A_c8YourString[L_u32Counter] != '\0' )
00310     {
00311
00312         L_u32Counter++ ;
00313
00314         A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00315
00316         if( (A_c8YourString[L_u32Counter] == '\n') || (A_c8YourString[L_u32Counter] == '\r') )
00317         {
00318             A_c8YourString[L_u32Counter] = '\0' ;
00319         }
00320
00321     }
00322
00323 }
```

References [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSynchBlocking\(\)](#).

Referenced by [HBluetooth_vReceiveString\(\)](#).

7.116.2.10 MUSART_u8CompareString()

```
u8_t MUSART_u8CompareString (
    c8_t * String1,
    c8_t * String2 )
```

Definition at line 328 of file [UART_program.c](#).

```
00329 {
00330
00331     u8_t L_u8TheComparingResult = INITIAL_ZERO ;
00332
00333     if( strcmp( String1, String2 ) == SAME_STRING )
00334     {
00335         L_u8TheComparingResult = SAME_STRING ;
00336     }
00337     else
00338     {
00339         L_u8TheComparingResult = DIFFERENT_STRING ;
00340     }
00341
00342     return L_u8TheComparingResult ;
00343
00344 }
```

References [DIFFERENT_STRING](#), [INITIAL_ZERO](#), and [SAME_STRING](#).

Referenced by [HBluetooth_u8CompStrings\(\)](#).

7.116.2.11 MUSART_u8ReadDataRegister()

```
u8_t MUSART_u8ReadDataRegister (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 349 of file [UART_program.c](#).

```
00350 {
00351     return A_USARTx->DR_REG ;
00352 }
```

References [USART_MemoryMapType::DR_REG](#).

7.116.2.12 MUSART_vClearFlags()

```
void MUSART_vClearFlags (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 357 of file [UART_program.c](#).

```
00358 {
00359     A_USARTx->SR_REG = 0 ;
00360 }
```

References [USART_MemoryMapType::SR_REG](#).

7.116.2.13 MUSART_vRxIntSetStatus()

```
void MUSART_vRxIntSetStatus (
    USART_MemoryMapType * A_USARTTx,
    u8_t A_u8Status )
```

Definition at line 365 of file [UART_program.c](#).

```
00366 {
00367
00368     switch(A_u8Status)
00369     {
00370
00371         case ENABLE : SET_BIT( A_USARTTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00372
00373         case DISABLE: CLR_BIT( A_USARTTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00374
00375     }
00376
00377 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), [DISABLE](#), [ENABLE](#), [MUSART_CR1_RXNEIE_BIT](#), and [SET_BIT](#).

7.116.2.14 MUSART1_vSetCallBack()

```
void MUSART1_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 382 of file [UART_program.c](#).

```
00383 {
00384     MUSART1_CallBack = Fptr ;
00385 }
```

References [MUSART1_CallBack](#).

7.116.2.15 MUSART2_vSetCallBack()

```
void MUSART2_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 390 of file [UART_program.c](#).

```
00391 {
00392     MUSART2_CallBack = Fptr ;
00393 }
```

References [MUSART2_CallBack](#).

7.116.2.16 MUSART6_vSetCallBack()

```
void MUSART6_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 398 of file [UART_program.c](#).

```
00399 {
00400     MUSART6_CallBack = Fptr ;
00401 }
```

References [MUSART6_CallBack](#).

7.117 UART_interface.h

[Go to the documentation of this file.](#)

```

00001 /* FILENAME: UART_interface
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 10/14/2022
00005 */
00006 #ifndef _UART_interface_H
00007 #define _UART_interface_H
00008
00009
00010
00011 /***** Peripherals declaration *****/
00012 /* Peripherals declaration */
00013 /***** Functions prototypes *****/
00014
00015 #define USART1_BASE_ADDRESS 0x40011000
00016 #define USART2_BASE_ADDRESS 0x40004400
00017 #define USART6_BASE_ADDRESS 0x40011400
00018
00019 typedef struct
00020 {
00021     volatile u32_t SR_REG      ;
00022     volatile u32_t DR_REG      ;
00023     volatile u32_t BRR_REG     ;
00024     volatile u32_t CR1_REG    ;
00025     volatile u32_t CR2_REG    ;
00026     volatile u32_t CR3_REG    ;
00027     volatile u32_t GTPR_REG   ;
00028
00029 } USART_MemoryMapType ;
00030
00031
00032
00033
00034
00035
00036 } USART_MemoryMapType ;
00037
00038
00039 #define USART1_REG ( ( USART_MemoryMapType*) USART1_BASE_ADDRESS)
00040 #define USART2_REG ( ( USART_MemoryMapType*) USART2_BASE_ADDRESS)
00041 #define USART6_REG ( ( USART_MemoryMapType*) USART6_BASE_ADDRESS)
00042
00043
00044 typedef struct
00045 {
00046     u32_t BaudRate      ;
00047     u8_t DataWidth      ;
00048     u8_t StopBits       ;
00049     u8_t Parity_Enable  ;
00050     u8_t Parity_Selection ;
00051     u8_t TransferDirection ; // TX_ONLY RX_ONLY TX_RX
00052     u8_t HardwareFlowControl;
00053     u8_t Oversampling    ;
00054
00055
00056 }USART_InitType ;
00057
00058
00059
00060
00061
00062 }USART_InitType ;
00063
00064
00065 typedef struct
00066 {
00067     u8_t ClockOutput      ;
00068     u8_t ClockPolarity     ;
00069     u8_t ClockPhase       ;
00070     u8_t LastBitClockPulse ;
00071
00072
00073
00074 }USART_ClockInitTypeDef ;
00075
00076
00077
00078 /***** Functions prototypes *****/
00079 /* Functions prototypes */
00080 /***** Functions prototypes *****/
00081
00082 void MUSART_vInit( USART_InitType *A_InitStruct,

```

```

00083     USART_ClockInitTypeDef *A_ClockInitStruct, USART_MemoryMapType *A_USARTx ) ;
00084
00085 void MUSART_vEnable( USART_MemoryMapType *A_USARTx ) ;
00086
00087 void MUSART_vDisable( USART_MemoryMapType *A_USARTx ) ;
00088
00089 void MUSART_vTransmitByte( USART_MemoryMapType *A_USARTx, u8_t A_u8Byte ) ;
00090
00091 void MUSART_vTransmitString( USART_MemoryMapType *A_USARTx, c8_t *A_ptrc8String ) ;
00092
00093 u8_t MUSART_u8ReceiveByteSyncNonBlocking ( USART_MemoryMapType *A_USARTx ) ;
00094
00095 u8_t MUSART_u8ReceiveByteSyncBlocking ( USART_MemoryMapType *A_USARTx ) ;
00096
00097 u8_t * MUSART_ptrReceiveStringSyncNonBlocking( USART_MemoryMapType *A_USARTx ) ;
00098
00099 void MUSART_vRecieveString( USART_MemoryMapType *A_USARTx, c8_t A_c8YourString[] ) ;
00100
00101 u8_t MUSART_u8CompareString( c8_t *String1, c8_t *String2 ) ;
00102
00103 u8_t MUSART_u8ReadDataRegister( USART_MemoryMapType *A_USARTx ) ;
00104
00105 void MUSART_vClearFlags( USART_MemoryMapType *A_USARTx ) ;
00106
00107 void MUSART_vRxIntSetStatus(USART_MemoryMapType *A_USARTx, u8_t A_u8Status) ;
00108
00109 void MUSART1_vSetCallBack( void (*Fptr) (void) ) ;
00110
00111 void MUSART2_vSetCallBack( void (*Fptr) (void) ) ;
00112
00113 void MUSART6_vSetCallBack( void (*Fptr) (void) ) ;
00114
00115
00116 /***** Interfacing macros ****/
00117 /*                                         */
00118 /*****                                         *****/
00119
00120 #define OVER_SAMPLING_16      0
00121 #define OVER_SAMPLING_8       1
00122
00123 #define TX_ONLY                0
00124 #define RX_ONLY                1
00125 #define TX_RX                 2
00126
00127 #define EVEN_PARITY           0
00128 #define ODD_PARITY            1
00129
00130 #define MODE_8BIT             0
00131 #define MODE_9BIT             1
00132
00133 #define STOP_BIT_1             0
00134 #define STOP_BIT_0_5           1
00135 #define STOP_BIT_2             2
00136 #define STOP_BIT_1_5           3
00137
00138 #define ENABLE                1
00139 #define DISABLE              0
00140
00141 #define __BAUDRATE__ 9600
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162 #endif // _UART_interface_H

```

7.118 COTS/MCAL/UART/UART_private.h File Reference

Macros

- #define `UART_DIV_SAMPLING16(_PCLK_, _BAUD_)` $((\text{u32_t})(((\text{u64_t})(\text{_PCLK_})*25U)/(4U*((\text{u64_t})(\text{_BAUD_}))))$
- #define `UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_)` $(\text{UART_DIV_AMPLING16}(\text{_PCLK_}, (\text{_BAUD_}))/100U)$
- #define `UART_DIVFRAQ_SAMPLING16(_PCLK_, _BAUD_)` $((((\text{UART_DIV_AMPLING16}(\text{_PCLK_}, (\text{_BAUD_}))) * 100U) * 16U) / 100U)$
- #define `UART_BRR_SAMPLING16(_PCLK_, _BAUD_)`
- #define `UART_DIV_SAMPLING8(_PCLK_, _BAUD_)` $((\text{u32_t})(((\text{u64_t})(\text{_PCLK_})*25U)/(2U*((\text{u64_t})(\text{_BAUD_}))))$
- #define `UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_)` $(\text{UART_DIV_AMPLING8}(\text{_PCLK_}, (\text{_BAUD_}))/100U)$
- #define `UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_)` $((((\text{UART_DIV_AMPLING8}(\text{_PCLK_}, (\text{_BAUD_}))) * 100U) * 8U) / 100U)$
- #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

7.118.1 Macro Definition Documentation

7.118.1.1 UART_DIV_SAMPLING16

```
#define UART_DIV_SAMPLING16(
    _PCLK_,
    _BAUD_ ) (((u32_t) (((u64_t) (_PCLK_) ) * 25U) / (4U * ((u64_t) (_BAUD_)))))
```

Definition at line 10 of file [UART_private.h](#).

7.118.1.2 UART_DIVMANT_SAMPLING16

```
#define UART_DIVMANT_SAMPLING16(
    _PCLK_,
    _BAUD_ ) (UART_DIV_SAMPLING16( (_PCLK_), (_BAUD_) ) / 100U)
```

Definition at line 11 of file [UART_private.h](#).

7.118.1.3 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 12 of file [UART_private.h](#).

7.118.1.4 UART_BRR_SAMPLING16

```
#define UART_BRR_SAMPLING16(
    _PCLK_,
    _BAUD_ )
```

Value:

```
+ \
0xFOU) + \
0xOFU))
```

$$\left(\left(\text{UART_DIVMANT_SAMPLING16}(\text{_PCLK_}, \text{_BAUD_}) \right) \ll 4U \right. \\ \left. + \left(\text{UART_DIVFRAQ_SAMPLING16}(\text{_PCLK_}, \text{_BAUD_}) \right) \& \right. \\ \left. \left(\text{UART_DIVFRAQ_SAMPLING16}(\text{_PCLK_}, \text{_BAUD_}) \right) \& \right. \\ \left. 0xOFU \right)$$

Definition at line 15 of file [UART_private.h](#).

7.118.1.5 UART_DIV_SAMPLING8

```
#define UART_DIV_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (((u32_t) (((u64_t) (_PCLK_) * 25U) / (2U * ((u64_t) (_BAUD_)))))
```

Definition at line 19 of file [UART_private.h](#).

7.118.1.6 UART_DIVMANT_SAMPLING8

```
#define UART_DIVMANT_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (UART_DIV_SAMPLING8(_PCLK_, _BAUD_) / 100U)
```

Definition at line 20 of file [UART_private.h](#).

7.118.1.7 UART_DIVFRAQ_SAMPLING8

```
#define UART_DIVFRAQ_SAMPLING8(
    _PCLK_,
    _BAUD_ ) (((UART_DIV_SAMPLING8(_PCLK_, _BAUD_) - (UART_DIVMANT_SAMPLING8((\_\_PCLK_), (\_\_BAUD_)) * 100U)) * 8U) + 50U) / 100U)
```

Definition at line 21 of file [UART_private.h](#).

7.118.1.8 **UART_BRR_SAMPLING8**

```
#define UART_BRR_SAMPLING8 (  
    __PCLK__,  
    __BAUD__ )
```

Value:

```
\  
    ((UART_DIVMANT_SAMPLING8((__PCLK__), (__BAUD__)) << 4U) +  
     ((UART_DIVFRAQ_SAMPLING8((__PCLK__), (__BAUD__)) &  
      (UART_DIVFRAQ_SAMPLING8((__PCLK__), (__BAUD__)) &  
      0x07U)))
```

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

7.118.1.9 **MUSART_SR_PE_BIT**

```
#define MUSART_SR_PE_BIT 0
```

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

7.118.1.10 **MUSART_SR_FE_BIT**

```
#define MUSART_SR_FE_BIT 1
```

Definition at line 35 of file [UART_private.h](#).

7.118.1.11 **MUSART_SR_NE_BIT**

```
#define MUSART_SR_NE_BIT 2
```

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

7.118.1.12 **MUSART_SR_ORE_BIT**

```
#define MUSART_SR_ORE_BIT 3
```

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

7.118.1.13 MUSART_SR_IDLE_BIT

```
#define MUSART_SR_IDLE_BIT 4
```

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

7.118.1.14 MUSART_SR_RXNE_BIT

```
#define MUSART_SR_RXNE_BIT 5
```

Definition at line 43 of file [UART_private.h](#).

7.118.1.15 MUSART_SR_TC_BIT

```
#define MUSART_SR_TC_BIT 6
```

Definition at line 45 of file [UART_private.h](#).

7.118.1.16 MUSART_SR_TXE_BIT

```
#define MUSART_SR_TXE_BIT 7
```

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

7.118.1.17 MUSART_SR_LBD_BIT

```
#define MUSART_SR_LBD_BIT 8
```

Definition at line 49 of file [UART_private.h](#).

7.118.1.18 MUSART_SR_CTS_BIT

```
#define MUSART_SR_CTS_BIT 9
```

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

7.118.1.19 MUSART_CR1_SBK_BIT

```
#define MUSART_CR1_SBK_BIT 0
```

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

7.118.1.20 MUSART_CR1_RWU_BIT

```
#define MUSART_CR1_RWU_BIT 1
```

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

7.118.1.21 MUSART_CR1_RE_BIT

```
#define MUSART_CR1_RE_BIT 2
```

Definition at line 61 of file [UART_private.h](#).

7.118.1.22 MUSART_CR1_TE_BIT

```
#define MUSART_CR1_TE_BIT 3
```

Definition at line 63 of file [UART_private.h](#).

7.118.1.23 MUSART_CR1_IDLEIE_BIT

```
#define MUSART_CR1_IDLEIE_BIT 4
```

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

7.118.1.24 MUSART_CR1_RXNEIE_BIT

```
#define MUSART_CR1_RXNEIE_BIT 5
```

Definition at line 67 of file [UART_private.h](#).

7.118.1.25 MUSART_CR1_TCIE_BIT

```
#define MUSART_CR1_TCIE_BIT 6
```

Definition at line [69](#) of file [UART_private.h](#).

7.118.1.26 MUSART_CR1_TXEIE_BIT

```
#define MUSART_CR1_TXEIE_BIT 7
```

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

7.118.1.27 MUSART_CR1_PEIE_BIT

```
#define MUSART_CR1_PEIE_BIT 8
```

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

7.118.1.28 MUSART_CR1_PS_BIT

```
#define MUSART_CR1_PS_BIT 9
```

Definition at line [75](#) of file [UART_private.h](#).

7.118.1.29 MUSART_CR1_PCE_BIT

```
#define MUSART_CR1_PCE_BIT 10
```

Definition at line [77](#) of file [UART_private.h](#).

7.118.1.30 MUSART_CR1_WAKE_BIT

```
#define MUSART_CR1_WAKE_BIT 11
```

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

7.118.1.31 MUSART_CR1_M_BIT

```
#define MUSART_CR1_M_BIT 12
```

Definition at line 81 of file [UART_private.h](#).

7.118.1.32 MUSART_CR1_UE_BIT

```
#define MUSART_CR1_UE_BIT 13
```

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

7.118.1.33 MUSART_CR1_OVER8_BIT

```
#define MUSART_CR1_OVER8_BIT 15
```

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

7.118.1.34 MUSART_CR2_ADD0_BIT

```
#define MUSART_CR2_ADD0_BIT 0
```

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

7.118.1.35 MUSART_CR2_ADD1_BIT

```
#define MUSART_CR2_ADD1_BIT 1
```

Definition at line 92 of file [UART_private.h](#).

7.118.1.36 MUSART_CR2_ADD2_BIT

```
#define MUSART_CR2_ADD2_BIT 2
```

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

7.118.1.37 MUSART_CR2_ADD3_BIT

```
#define MUSART_CR2_ADD3_BIT 3
```

Definition at line 94 of file [UART_private.h](#).

7.118.1.38 MUSART_CR2_LBDL_BIT

```
#define MUSART_CR2_LBDL_BIT 5
```

Definition at line 96 of file [UART_private.h](#).

7.118.1.39 MUSART_CR2_LBDIE_BIT

```
#define MUSART_CR2_LBDIE_BIT 6
```

Definition at line 98 of file [UART_private.h](#).

7.118.1.40 MUSART_CR2_LBCL_BIT

```
#define MUSART_CR2_LBCL_BIT 8
```

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

7.118.1.41 MUSART_CR2_CPHA_BIT

```
#define MUSART_CR2_CPHA_BIT 9
```

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

7.118.1.42 MUSART_CR2_CPOL_BIT

```
#define MUSART_CR2_CPOL_BIT 10
```

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

7.118.1.43 MUSART_CR2_CLKEN_BIT

```
#define MUSART_CR2_CLKEN_BIT 11
```

Definition at line 106 of file [UART_private.h](#).

7.118.1.44 MUSART_CR2_STOP_BIT

```
#define MUSART_CR2_STOP_BIT 12
```

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

7.118.1.45 MUSART_CR2_STOP0_BIT

```
#define MUSART_CR2_STOP0_BIT 12
```

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

7.118.1.46 MUSART_CR2_STOP1_BIT

```
#define MUSART_CR2_STOP1_BIT 13
```

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

7.118.1.47 MUSART_CR2_LINEN_BIT

```
#define MUSART_CR2_LINEN_BIT 14
```

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

7.118.1.48 MUSART_CR3_CTSIE_BIT

```
#define MUSART_CR3_CTSIE_BIT 10
```

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

7.118.1.49 MUSART_CR3_CTSE_BIT

```
#define MUSART_CR3_CTSE_BIT 9
```

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

7.118.1.50 MUSART_CR3_RTSE_BIT

```
#define MUSART_CR3_RTSE_BIT 8
```

Definition at line 123 of file [UART_private.h](#).

7.118.1.51 MUSART_CR3_DMAT_BIT

```
#define MUSART_CR3_DMAT_BIT 7
```

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

7.118.1.52 MUSART_CR3_DMAR_BIT

```
#define MUSART_CR3_DMAR_BIT 6
```

Definition at line 127 of file [UART_private.h](#).

7.118.1.53 MUSART_CR3_SCEN_BIT

```
#define MUSART_CR3_SCEN_BIT 5
```

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

7.118.1.54 MUSART_CR3_NACK_BIT

```
#define MUSART_CR3_NACK_BIT 4
```

Definition at line 131 of file [UART_private.h](#).

7.118.1.55 MUSART_CR3_HDSEL_BIT

```
#define MUSART_CR3_HDSEL_BIT 3
```

Definition at line 133 of file [UART_private.h](#).

7.118.1.56 MUSART_CR3_IRLP_BIT

```
#define MUSART_CR3_IRLP_BIT 2
```

Definition at line 135 of file [UART_private.h](#).

7.118.1.57 MUSART_CR3_IREN_BIT

```
#define MUSART_CR3_IREN_BIT 1
```

Definition at line 137 of file [UART_private.h](#).

7.118.1.58 MUSART_CR3_EIE_BIT

```
#define MUSART_CR3_EIE_BIT 0
```

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

7.119 UART_private.h

[Go to the documentation of this file.](#)

```
00001 /* FILENAME: UART_private
00002 * Author: Ali El Bana
00003 * Version: V1.0
00004 * DATE: Fri 10/14/2022
00005 */
00006 #ifndef _UART_private_H
00007 #define _UART_private_H
00008
00009
00010 #define UART_DIV_SAMPLING16(_PCLK_, _BAUD_)
    ((u32_t) (((u64_t) (_PCLK_)) *25U) / (4U*((u64_t) (_BAUD_))))
00011 #define UART_DIVMANT_SAMPLING16 (_PCLK_, _BAUD_)          (UART_DIV_SAMPLING16(_PCLK_, (_BAUD_))/100U)
00012 #define UART_DIVFRAQ_SAMPLING16 (_PCLK_, _BAUD_)         (((UART_DIV_SAMPLING16(_PCLK_, (_BAUD_)) -
    (UART_DIVMANT_SAMPLING16(_PCLK_, (_BAUD_)) * 100U)) * 16U) + 50U) / 100U)
00013 /* UART BRR = mantissa + overflow + fraction
00014 = (UART DIVMANT << 4) + (UART DIVFRAQ & 0xF0) + (UART DIVFRAQ & 0x0FU) */
00015 #define UART_BRR_SAMPLING16 (_PCLK_, _BAUD_)           ((UART_DIVMANT_SAMPLING16(_PCLK_, (_BAUD_)) <<
    4U) + \
00016 (UART_DIVFRAQ_SAMPLING16(_PCLK_, (_BAUD_)) & 0x0FU) + \
00017 (UART_DIVFRAQ_SAMPLING16(_PCLK_, (_BAUD_)) & 0x0FU))
00018
00019 #define UART_DIV_SAMPLING8(_PCLK_, _BAUD_)
    ((u32_t) (((u64_t) (_PCLK_)) *25U) / (2U*((u64_t) (_BAUD_))))
00020 #define UART_DIVMANT_SAMPLING8 (_PCLK_, _BAUD_)          (UART_DIV_SAMPLING8(_PCLK_, (_BAUD_))/100U)
00021 #define UART_DIVFRAQ_SAMPLING8 (_PCLK_, _BAUD_)         (((UART_DIV_SAMPLING8(_PCLK_, (_BAUD_)) -
    (UART_DIVMANT_SAMPLING8(_PCLK_, (_BAUD_)) * 100U)) * 8U) + 50U) / 100U)
```

```

00022 /* UART_BRR = mantissa + overflow + fraction
00023 = (UART_DIVMANT « 4) + ((UART_DIVFRAQ & 0xF8) « 1) + (UART_DIVFRAQ & 0x07U) */
00024 #define UART_BRR_SAMPLING8(_PCLK_, _BAUD_) ((UART_DIVMANT_SAMPLING8(_PCLK_), (_BAUD_)) «
4U) + \
00025 ((UART_DIVFRAQ_SAMPLING8(_PCLK_), (_BAUD_)) & 0x07U) « 1U) + \
00026 ((UART_DIVFRAQ_SAMPLING8(_PCLK_), (_BAUD_)) & 0x07U))
00027
00028
00029 /***** SR BITS Mapping ****/
00030 /*          SR_BITS Mapping           */
00031 /***** SR BITS Mapping ****/
00032 /*  Parity error           */
00033 #define MUSART_SR_PE_BIT    0
00034 /*  Framing error          */
00035 #define MUSART_SR_FE_BIT    1
00036 /*  Noise error flag        */
00037 #define MUSART_SR_NE_BIT    2
00038 /*  Overrun error           */
00039 #define MUSART_SR_ORE_BIT   3
00040 /*  IDLE line detected      */
00041 #define MUSART_SR_IDLE_BIT  4
00042 /*  Read data register not empty */
00043 #define MUSART_SR_RXNE_BIT  5
00044 /*  Transmission complete   */
00045 #define MUSART_SR_TC_BIT    6
00046 /*  Transmit data register empty */
00047 #define MUSART_SR_TXE_BIT   7
00048 /*  LIN break detection flag */
00049 #define MUSART_SR_LBD_BIT   8
00050 /*  CTS flag                */
00051 #define MUSART_SR_CTS_BIT   9
00052
00053 /***** CRL BITS Mapping ****/
00054 /*          CRL_BITS Mapping           */
00055 /***** CRL BITS Mapping ****/
00056 /*  Send break bit */
00057 #define MUSART_CRL_SBK_BIT  0
00058 /*  Recevier Wakeup bit */
00059 #define MUSART_CRL_RWU_BIT  1
00060 /*  Recevier Enable bit */
00061 #define MUSART_CRL_RE_BIT   2
00062 /*  Transmitter Enable bit */
00063 #define MUSART_CRL_TE_BIT   3
00064 /*  IDLE interrupt enable bit */
00065 #define MUSART_CRL_IDLEIE_BIT 4
00066 /*  RXNEIE interrupt enable bit */
00067 #define MUSART_CRL_RXNEIE_BIT 5
00068 /*  Transmission complete interrupt enable bit */
00069 #define MUSART_CRL_TCIE_BIT  6
00070 /*  TXE interrupt enable bit */
00071 #define MUSART_CRL_TXEIE_BIT 7
00072 /*  PE interrupt enable bit */
00073 #define MUSART_CRL_PEIE_BIT  8
00074 /*  Parity selection bit */
00075 #define MUSART_CRL_PS_BIT   9
00076 /*  Parity control enable bit */
00077 #define MUSART_CRL_PCE_BIT  10
00078 /*  Wakeup method bit */
00079 #define MUSART_CRL_WAKE_BIT 11
00080 /*  Word length bit */
00081 #define MUSART_CRL_M_BIT   12
00082 /*  USART enable bit */
00083 #define MUSART_CRL_UE_BIT  13
00084 /*  USART Oversampling bit */
00085 #define MUSART_CRL_OVER8_BIT 15
00086
00087 /***** CR2 BITS Mapping ****/
00088 /*          CR2_BITS Mapping           */
00089 /***** CR2 BITS Mapping ****/
00090 /*  Address of the USART node bits */
00091 #define MUSART_CR2_ADD0_BIT  0
00092 #define MUSART_CR2_ADD1_BIT  1
00093 #define MUSART_CR2_ADD2_BIT  2
00094 #define MUSART_CR2_ADD3_BIT  3
00095 /*  lin break detection length bit */
00096 #define MUSART_CR2_LBDL_BIT 5
00097 /*  LIN break detection interrupt enable bit */
00098 #define MUSART_CR2_LBDIE_BIT 6
00099 /*  Last bit clock pulse bit */
00100 #define MUSART_CR2_LBCL_BIT 8
00101 /*  Clock phase bit */
00102 #define MUSART_CR2_CPHA_BIT 9
00103 /*  Clock polarity bit */
00104 #define MUSART_CR2_CPOL_BIT 10
00105 /*  Clock enable bit */
00106 #define MUSART_CR2_CLKEN_BIT 11
00107 /*  STOP bit start */

```

```

00108 #define MUSART_CR2_STOP_BIT      12
00109 /* STOP bits */
00110 #define MUSART_CR2_STOP0_BIT     12
00111 #define MUSART_CR2_STOP1_BIT     13
00112 /* LIN mode enable bit */
00113 #define MUSART_CR2_LINEN_BIT    14
00114
00115 /***** CR3 BITS Mapping *****/
00116 /*          CR3 BITS Mapping          */
00117 /***** CR3 BITS Mapping *****/
00118 /* CTS interrupt enable bit */
00119 #define MUSART_CR3_CTSIE_BIT    10
00120 /* CTS enable bit */
00121 #define MUSART_CR3_CTSE_BIT      9
00122 /* RTS enable bit */
00123 #define MUSART_CR3_RTSE_BIT      8
00124 /* DMA enable transmitter bit */
00125 #define MUSART_CR3_DMAT_BIT      7
00126 /* DMA enable receiver bit */
00127 #define MUSART_CR3_DMAR_BIT      6
00128 /* Smartcard mode enable bit */
00129 #define MUSART_CR3_SCEN_BIT      5
00130 /* Smartcard NACK enable bit */
00131 #define MUSART_CR3_NACK_BIT      4
00132 /* Half-duplex selection bit */
00133 #define MUSART_CR3_HDSEL_BIT      3
00134 /* IrDA low-power bit */
00135 #define MUSART_CR3_IRLP_BIT      2
00136 /* IrDA mode enable bit */
00137 #define MUSART_CR3_IREN_BIT      1
00138 /* Error interrupt enable bit */
00139 #define MUSART_CR3_EIE_BIT       0
00140
00141
00142
00143
00144
00145
00146
00147 #endif // _UART_private_H

```

7.120 COTS/MCAL/UART/UART_program.c File Reference

```

#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "string.h"
#include "../RCC/MRCC_interface.h"
#include "../GPIO/GPIO_interface.h"
#include "UART_interface.h"
#include "UART_private.h"
#include "UART_config.h"

```

Functions

- void **MUSART_vInit** (USART_InitType *A_InitStruct, USART_ClockInitTypeDef *A_ClockInitStruct, USART_MemoryMapType *A_USARTx)
- void **MUSART_vEnable** (USART_MemoryMapType *A_USARTx)
- void **MUSART_vDisable** (USART_MemoryMapType *A_USARTx)
- void **MUSART_vTransmitByte** (USART_MemoryMapType *A_USARTx, u8_t A_u8Byte)
- void **MUSART_vTransmitString** (USART_MemoryMapType *A_USARTx, c8_t *A_ptrc8String)
- u8_t **MUSART_u8ReceiveByteNonBlocking** (USART_MemoryMapType *A_USARTx)
- u8_t **MUSART_u8ReceiveByteSyncBlocking** (USART_MemoryMapType *A_USARTx)
- u8_t * **MUSART_ptrReceiveStringSyncNonBlocking** (USART_MemoryMapType *A_USARTx)
- void **MUSART_vRecieveString** (USART_MemoryMapType *A_USARTx, c8_t A_c8YourString[])

- `u8_t MUSART_u8CompareString (c8_t *String1, c8_t *String2)`
- `u8_t MUSART_u8ReadDataRegister (USART_MemoryMapType *A_USARTTx)`
- `void MUSART_vClearFlags (USART_MemoryMapType *A_USARTTx)`
- `void MUSART_vRxIntSetStatus (USART_MemoryMapType *A_USARTTx, u8_t A_u8Status)`
- `MUSART1_vSetCallBack (void(*Fptr)(void))`
- `MUSART2_vSetCallBack (void(*Fptr)(void))`
- `MUSART6_vSetCallBack (void(*Fptr)(void))`
- `void USART1_IRQHandler (void)`
- `void USART2_IRQHandler (void)`
- `void USART6_IRQHandler (void)`

Variables

- `u8_t G_u8String [20]`
- `void(* MUSART1_CallBack)(void)`
- `void(* MUSART2_CallBack)(void)`
- `void(* MUSART6_CallBack)(void)`

7.120.1 Function Documentation

7.120.1.1 MUSART_vInit()

```
void MUSART_vInit (
    USART_InitType * A_InitStruct,
    USART_ClockInitTypeDef * A_ClockInitStruct,
    USART_MemoryMapType * A_USARTx )
```

Definition at line 37 of file [UART_program.c](#).

```
00039 {
00040
00041     if( A_USARTx == USART1_REG )
00042     {
00043
00044         MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART1EN ) ;
00045
00046         /*configer Tx1 as alt fun*/
00047         MGPIox_ConfigType TX1 =
00048         {
00049             .Port          = USART1_PORT      ,   .Pin           = TX1_PIN
00050             .Mode          = GPIOx_MODE_AF    ,   .OutputType     = GPIOx_PUSH_PULL,
00051             .OutputSpeed   = GPIOx_MediumSpeed,   .InputType      = GPIOx_NoPull ,
00052             .AF_Type       = GPIOx_AF7
00053         } ;
00054
00055         MGPIox_vInit( &TX1 ) ;
00056
00057         /*configer Rx1 as alt fun*/
00058         MGPIox_ConfigType RX1 =
00059         {
00060             .Port          = USART1_PORT      ,   .Pin           = RX1_PIN
00061             .Mode          = GPIOx_MODE_AF    ,   .OutputType     = GPIOx_PUSH_PULL,
00062             .OutputSpeed   = GPIOx_MediumSpeed,   .InputType      = GPIOx_NoPull ,
00063             .AF_Type       = GPIOx_AF7
00064         } ;
00065
00066         MGPIox_vInit( &RX1 ) ;
00067
00068     }
00069     else if( A_USARTx == USART2_REG )
00070     {
00071         MRCC_vEnablePeriphralCLK( RCC_APB2, APB1ENR_USART2EN ) ;
```

```

00073
00074     /*configer Tx2 as alt fun*/
00075     MGPIox_ConfigType TX2 =
00076     {
00077         .Port          = USART2_PORT      ,   .Pin          = TX2_PIN      ,
00078         .Mode          = GPIOx_MODE_AF    ,   .OutputType   = GPIOx_PUSH_PULL,
00079         .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType    = GPIOx_NoPull  ,
00080         .AF_Type       = GPIOx_AF7
00081     } ;
00082
00083     MGPIox_vInit( &TX2 ) ;
00084
00085     /*configer Rx2 as alt fun*/
00086     MGPIox_ConfigType RX2 =
00087     {
00088         .Port          = USART2_PORT      ,   .Pin          = RX2_PIN      ,
00089         .Mode          = GPIOx_MODE_AF    ,   .OutputType   = GPIOx_PUSH_PULL,
00090         .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType    = GPIOx_NoPull  ,
00091         .AF_Type       = GPIOx_AF7
00092     } ;
00093
00094     MGPIox_vInit( &RX2 ) ;
00095
00096 }
00097 else if( A_USARTx == USART6_REG )
00098 {
00099
00100     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART6EN ) ;
00101
00102     /*configer Tx6 as alt fun*/
00103     MGPIox_ConfigType TX6 =
00104     {
00105         .Port          = USART6_PORT      ,   .Pin          = TX6_PIN      ,
00106         .Mode          = GPIOx_MODE_AF    ,   .OutputType   = GPIOx_PUSH_PULL,
00107         .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType    = GPIOx_NoPull  ,
00108         .AF_Type       = GPIOx_AF7
00109     } ;
00110
00111     MGPIox_vInit( &TX6 ) ;
00112
00113     /*configer Rx6 as alt fun*/
00114     MGPIox_ConfigType RX6 =
00115     {
00116         .Port          = USART6_PORT      ,   .Pin          = RX6_PIN      ,
00117         .Mode          = GPIOx_MODE_AF    ,   .OutputType   = GPIOx_PUSH_PULL,
00118         .OutputSpeed   = GPIOx_MediumSpeed ,   .InputType    = GPIOx_NoPull  ,
00119         .AF_Type       = GPIOx_AF7
00120     } ;
00121
00122     MGPIox_vInit( &RX6 ) ;
00123
00124 }
00125
00126 // BuadRate/Over-sampling selections:
00127 switch( A_InitStruct->Oversampling )
00128 {
00129
00130     case OVER_SAMPLING_16:
00131
00132         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00133
00134     break;
00135
00136     case OVER_SAMPLING_8:
00137
00138         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00139
00140     break;
00141
00142 }
00143
00144 A_USARTx->CR1_REG = ( A_InitStruct->Oversampling << MUSART_CR1_OVER8_BIT )   |
00145             ( A_InitStruct->DataWidth << MUSART_CR1_M_BIT )           |
00146             ( A_InitStruct->Parity_Enable << MUSART_CR1_PCE_BIT )        |
00147             ( A_InitStruct->Parity_Selection << MUSART_CR1_PS_BIT ) ;
00148
00149 switch (A_InitStruct->TransferDirection)
00150 {
00151     case TX_ONLY:
00152
00153         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00154
00155     break ;
00156
00157     case RX_ONLY:
00158
00159         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;

```

```

00160
00161     break ;
00162
00163     case TX_RX :
00164
00165         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00166         SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;
00167
00168     break ;
00169
00170 }
00171
00172 A_USARTx->CR2_REG = ( A_InitStruct->StopBits << MUSART_CR2_STOP_BIT )
00173             | ( A_ClockInitStruct->ClockOutput << MUSART_CR2_CLKEN_BIT )
00174             | ( A_ClockInitStruct->ClockPhase << MUSART_CR2_CPHA_BIT )
00175             | ( A_ClockInitStruct->ClockPolarity << MUSART_CR2_CPOL_BIT )
00176             | ( A_ClockInitStruct->LastBitClockPulse << MUSART_CR2_LBCL_BIT ) ;
00177
00178 A_USARTx->SR_REG = 0 ; // Clear all the flags.
00179
00180 SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ; // EN the peripheral.
00181
00182 }
```

References `_PCLK_`, 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, GPIOx_vInit(), MRCC_vEnablePeripheralCLK(), MUSART_CR1_M_BIT, MUSART_CR1_OVER8_BIT, MUSART_CR1_PCE_BIT, MUSART_CR1_PS_BIT, MUSART_CR1_RE_BIT, MUSART_CR1_TE_BIT, MUSART_CR1_UE_BIT, MUSART_CR2_CLKEN_BIT, MUSART_CR2_CPHA_BIT, MUSART_CR2_CPOL_BIT, MUSART_CR2_LBCL_BIT, MUSART_CR2_STOP_BIT, OVER_SAMPLING_16, OVER_SAMPLING_8, USART_InitType::Oversampling, USART_InitType::Parity_Enable, USART_InitType::Parity_Selection, GPIOx_ConfigType::Port, RCC_APB2_RX1_PIN, RX2_PIN, RX6_PIN, RX_ONLY, SET_BIT, USART_MemoryMapType::SR_REG, USART_InitType::StopBits, USART_InitType::TransferDirection, TX1_PIN, TX2_PIN, TX6_PIN, TX_ONLY, TX_RX, UART_BRR_SAMPLING16, USART1_PORT, USART1_REG, USART2_PORT, USART2_REG, USART6_PORT, and USART6_REG.

Referenced by [HBluetooth_vInit\(\)](#).

7.120.1.2 MUSART_vEnable()

```
void MUSART_vEnable (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 187 of file [UART_program.c](#).

```
00188 {
00189     SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00190 }
```

References [USART_MemoryMapType::CR1_REG](#), [MUSART_CR1_UE_BIT](#), and [SET_BIT](#).

Referenced by [HBluetooth_vEnable\(\)](#).

7.120.1.3 MUSART_vDisable()

```
void MUSART_vDisable (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 195 of file [UART_program.c](#).

```
00196 {
00197     CLR_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00198 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), and [MUSART_CR1_UE_BIT](#).

Referenced by [HBluetooth_vDisable\(\)](#).

7.120.1.4 MUSART_vTransmitByte()

```
void MUSART_vTransmitByte (
    USART_MemoryMapType * A_USARTTx,
    u8_t A_u8Byte )
```

Definition at line 203 of file [UART_program.c](#).

```
00204 {
00205
00206     while( GET_BIT( A_USARTTx->SR_REG, MUSART_SR_TXE_BIT ) == 0 ) ;
00207
00208     A_USARTTx->DR_REG = A_u8Byte ;
00209
00210     while( GET_BIT( A_USARTTx->SR_REG, MUSART_SR_TC_BIT ) == 0 ) ;
00211
00212     // Clear the TC flag:
00213     CLR_BIT( A_USARTTx->SR_REG, MUSART_SR_TC_BIT ) ;
00214
00215 }
```

References [CLR_BIT](#), [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_SR_TC_BIT](#), [MUSART_SR_TXE_BIT](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [HBluetooth_vSendByte\(\)](#), and [MUSART_vTransmitString\(\)](#).

7.120.1.5 MUSART_vTransmitString()

```
void MUSART_vTransmitString (
    USART_MemoryMapType * A_USARTTx,
    c8_t * A_ptrc8String )
```

Definition at line 220 of file [UART_program.c](#).

```
00221 {
00222
00223     u8_t loc_u8Iterator = INITIAL_ZERO ;
00224
00225     while( A_ptrc8String[loc_u8Iterator] != '\0' )
00226     {
00227         MUSART_vTransmitByte( A_USARTTx, A_ptrc8String[loc_u8Iterator] ) ;
00228
00229         loc_u8Iterator++ ;
00230
00231     }
00232
00233
00234 }
```

References [INITIAL_ZERO](#), and [MUSART_vTransmitByte\(\)](#).

Referenced by [HBluetooth_vSendString\(\)](#).

7.120.1.6 MUSART_u8ReceiveByteSynchNonBlocking()

```
u8_t MUSART_u8ReceiveByteSynchNonBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 239 of file [UART_program.c](#).

```
00240 {
00241     u8_t loc_u8Data = INITIAL_ZERO ;
00242     u32_t loc_u8TimeOut = INITIAL_ZERO ;
00243
00244     while( (GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0) && (loc_u8TimeOut < THRESHOLD_VALUE) )
00245     {
00246         loc_u8TimeOut++;
00247     }
00248
00249     if ( loc_u8TimeOut == THRESHOLD_VALUE )
00250     {
00251         loc_u8Data = 255 ; // To detect the error.
00252     }
00253     else
00254     {
00255         loc_u8Data = A_USARTx->DR_REG ;
00256     }
00257
00258     return loc_u8Data ;
00259
00260 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [INITIAL_ZERO](#), [MUSART_SR_RXNE_BIT](#), [USART_MemoryMapType::SR_REG](#), and [THRESHOLD_VALUE](#).

Referenced by [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#).

7.120.1.7 MUSART_u8ReceiveByteSynchBlocking()

```
u8_t MUSART_u8ReceiveByteSynchBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 267 of file [UART_program.c](#).

```
00268 {
00269     while( GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0 ) ;
00270
00271     return A_USARTx->DR_REG ;
00272
00273 }
```

References [USART_MemoryMapType::DR_REG](#), [GET_BIT](#), [MUSART_SR_RXNE_BIT](#), and [USART_MemoryMapType::SR_REG](#).

Referenced by [HBluetooth_u8ReceiveByte\(\)](#), and [MUSART_vRecieveString\(\)](#).

7.120.1.8 MUSART_ptrReceiveStringSynchNonBlocking()

```
u8_t * MUSART_ptrReceiveStringSynchNonBlocking (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 279 of file [UART_program.c](#).

```
00280 {
00281     u8_t loc_u8Iterator = INITIAL_ZERO ;
00282
00283     u8_t loc_u8DataCome ;
00284
00285     while( ( loc_u8DataCome = MUSART_u8ReceiveByteSynchNonBlocking(A_USARTx) ) != 13 )
00286     {
00287         G_u8String[ loc_u8Iterator ] = loc_u8DataCome ;
00288
00289         loc_u8Iterator++ ;
00290     }
00291
00292     G_u8String[loc_u8Iterator] = '\0' ;
00293
00294     return( G_u8String ) ;
00295
00296
00297 }
```

References [G_u8String](#), [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSynchNonBlocking\(\)](#).

7.120.1.9 MUSART_vRecieveString()

```
void MUSART_vRecieveString (
    USART_MemoryMapType * A_USARTx,
    c8_t A_c8YourString[] )
```

Definition at line 302 of file [UART_program.c](#).

```
00303 {
00304
00305     u32_t L_u32Counter = INITIAL_ZERO ;
00306
00307     A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00308
00309     while( A_c8YourString[L_u32Counter] != '\0' )
00310     {
00311
00312         L_u32Counter++ ;
00313
00314         A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00315
00316         if( (A_c8YourString[L_u32Counter] == '\n') || (A_c8YourString[L_u32Counter] == '\r') )
00317         {
00318             A_c8YourString[L_u32Counter] = '\0' ;
00319         }
00320
00321     }
00322
00323 }
```

References [INITIAL_ZERO](#), and [MUSART_u8ReceiveByteSynchBlocking\(\)](#).

Referenced by [HBluetooth_vReceiveString\(\)](#).

7.120.1.10 MUSART_u8CompareString()

```
u8_t MUSART_u8CompareString (
    c8_t * String1,
    c8_t * String2 )
```

Definition at line 328 of file [UART_program.c](#).

```
00329 {
00330
00331     u8_t L_u8TheComparingResult = INITIAL_ZERO ;
00332
00333     if( strcmp( String1, String2 ) == SAME_STRING )
00334     {
00335         L_u8TheComparingResult = SAME_STRING ;
00336     }
00337     else
00338     {
00339         L_u8TheComparingResult = DIFFERENT_STRING ;
00340     }
00341
00342     return L_u8TheComparingResult ;
00343
00344 }
```

References [DIFFERENT_STRING](#), [INITIAL_ZERO](#), and [SAME_STRING](#).

Referenced by [HBluetooth_u8CompStrings\(\)](#).

7.120.1.11 MUSART_u8ReadDataRegister()

```
u8_t MUSART_u8ReadDataRegister (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 349 of file [UART_program.c](#).

```
00350 {
00351     return A_USARTx->DR_REG ;
00352 }
```

References [USART_MemoryMapType::DR_REG](#).

7.120.1.12 MUSART_vClearFlags()

```
void MUSART_vClearFlags (
    USART_MemoryMapType * A_USARTx )
```

Definition at line 357 of file [UART_program.c](#).

```
00358 {
00359     A_USARTx->SR_REG = 0 ;
00360 }
```

References [USART_MemoryMapType::SR_REG](#).

7.120.1.13 MUSART_vRxIntSetStatus()

```
void MUSART_vRxIntSetStatus (
    USART_MemoryMapType * A_USARTx,
    u8_t A_u8Status )
```

Definition at line 365 of file [UART_program.c](#).

```
00366 {
00367
00368     switch(A_u8Status)
00369     {
00370
00371         case ENABLE : SET_BIT( A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00372
00373         case DISABLE: CLR_BIT( A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00374
00375     }
00376
00377 }
```

References [CLR_BIT](#), [USART_MemoryMapType::CR1_REG](#), [DISABLE](#), [ENABLE](#), [MUSART_CR1_RXNEIE_BIT](#), and [SET_BIT](#).

7.120.1.14 MUSART1_vSetCallBack()

```
void MUSART1_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 382 of file [UART_program.c](#).

```
00383 {
00384     MUSART1_CallBack = Fptr ;
00385 }
```

References [MUSART1_CallBack](#).

7.120.1.15 MUSART2_vSetCallBack()

```
void MUSART2_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 390 of file [UART_program.c](#).

```
00391 {
00392     MUSART2_CallBack = Fptr ;
00393 }
```

References [MUSART2_CallBack](#).

7.120.1.16 MUSART6_vSetCallBack()

```
void MUSART6_vSetCallBack (
    void(*)(void) Fptr )
```

Definition at line 398 of file [UART_program.c](#).

```
00399 {
00400     MUSART6_CallBack = Fptr ;
00401 }
```

References [MUSART6_CallBack](#).

7.120.1.17 USART1_IRQHandler()

```
void USART1_IRQHandler (
    void )
```

Definition at line 406 of file [UART_program.c](#).

```
00407 {
00408     USART1_REG->SR_REG = 0 ;
00410
00411     MUSART1_CallBack( ) ;
00412
00413 }
```

References [MUSART1_CallBack](#), and [USART1_REG](#).

7.120.1.18 USART2_IRQHandler()

```
void USART2_IRQHandler (
    void )
```

Definition at line 418 of file [UART_program.c](#).

```
00419 {
00420     USART2_REG -> SR_REG = 0 ;
00422
00423     MUSART2_CallBack( ) ;
00424
00425 }
```

References [MUSART2_CallBack](#), and [USART2_REG](#).

7.120.1.19 USART6_IRQHandler()

```
void USART6_IRQHandler (
    void )
```

Definition at line 430 of file [UART_program.c](#).

```
00431 {
00432     USART6_REG->SR_REG = 0 ;
00434
00435     MUSART6_CallBack( ) ;
00436
00437 }
```

References [MUSART6_CallBack](#), and [USART6_REG](#).

7.120.2 Variable Documentation

7.120.2.1 G_u8String

```
u8_t G_u8String[20]
```

Definition at line 28 of file [UART_program.c](#).

Referenced by [MUSART_ptrReceiveStringSynchNonBlocking\(\)](#).

7.120.2.2 MUSART1_CallBack

```
void(* MUSART1_CallBack) (void) (
    void )
```

Definition at line 30 of file [UART_program.c](#).

Referenced by [MUSART1_vSetCallBack\(\)](#), and [USART1_IRQHandler\(\)](#).

7.120.2.3 MUSART2_CallBack

```
void(* MUSART2_CallBack) (void) (
    void )
```

Definition at line 31 of file [UART_program.c](#).

Referenced by [MUSART2_vSetCallBack\(\)](#), and [USART2_IRQHandler\(\)](#).

7.120.2.4 MUSART6_CallBack

```
void(* MUSART6_CallBack) (void) (
    void )
```

Definition at line 32 of file [UART_program.c](#).

Referenced by [MUSART6_vSetCallBack\(\)](#), and [USART6_IRQHandler\(\)](#).

7.121 UART_program.c

[Go to the documentation of this file.](#)

```

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 /***** */
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 /***** */
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 /***** */
00035 /***** */
00036
00037 void MUSART_vInit( USART_InitType *A_InitStruct,
00038                     USART_ClockInitTypeDef *A_ClockInitStruct, USART_MemoryMapType *A_USARTx )
00039 {
00040
00041     if( A_USARTx == USART1_REG )
00042     {
00043
00044         MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART1EN ) ;
00045
00046         /*configer Tx1 as alt fun*/
00047         MGPIox_ConfigType TX1 =
00048         {
00049             .Port      = USART1_PORT      , .Pin       = TX1_PIN      ,
00050             .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00051             .OutputSpeed = GPIOx_MediumSpeed , .InputType  = GPIOx_NoPull ,
00052             .AF_Type   = GPIOx_AF7
00053         } ;
00054
00055         MGPIox_vInit( &TX1 ) ;
00056
00057         /*configer Rx1 as alt fun*/
00058         MGPIox_ConfigType RX1 =
00059         {
00060             .Port      = USART1_PORT      , .Pin       = RX1_PIN      ,
00061             .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00062             .OutputSpeed = GPIOx_MediumSpeed , .InputType  = GPIOx_NoPull ,
00063             .AF_Type   = GPIOx_AF7
00064         } ;
00065
00066         MGPIox_vInit( &RX1 ) ;
00067
00068     }
00069     else if( A_USARTx == USART2_REG )
00070     {
00071
00072         MRCC_vEnablePeriphralCLK( RCC_APB2, APB1ENR_USART2EN ) ;
00073
00074         /*configer Tx2 as alt fun*/
00075         MGPIox_ConfigType TX2 =
00076         {
00077             .Port      = USART2_PORT      , .Pin       = TX2_PIN      ,
00078             .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL ,
00079             .OutputSpeed = GPIOx_MediumSpeed , .InputType  = GPIOx_NoPull ,
00080             .AF_Type   = GPIOx_AF7
00081         } ;
00082

```

```

00083     MGPIOx_vInit( &TX2 ) ;
00084
00085     /*configer Rx2 as alt fun*/
00086     MGPIOx_ConfigType RX2 =
00087     {
00088         .Port      = USART2_PORT      , .Pin      = RX2_PIN
00089         .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL,
00090         .OutputSpeed = GPIOx_MediumSpeed , .InputType = GPIOx_NoPull ,
00091         .AF_Type   = GPIOx_AF7
00092     } ;
00093
00094     MGPIOx_vInit( &RX2 ) ;
00095
00096 }
00097 else if( A_USARTx == USART6_REG )
00098 {
00099
00100     MRCC_vEnablePeriphralCLK( RCC_APB2, APB2ENR_USART6EN ) ;
00101
00102     /*configer Tx6 as alt fun*/
00103     MGPIOx_ConfigType TX6 =
00104     {
00105         .Port      = USART6_PORT      , .Pin      = TX6_PIN
00106         .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL,
00107         .OutputSpeed = GPIOx_MediumSpeed , .InputType = GPIOx_NoPull ,
00108         .AF_Type   = GPIOx_AF7
00109     } ;
00110
00111     MGPIOx_vInit( &TX6 ) ;
00112
00113     /*configer Rx6 as alt fun*/
00114     MGPIOx_ConfigType RX6 =
00115     {
00116         .Port      = USART6_PORT      , .Pin      = RX6_PIN
00117         .Mode      = GPIOx_MODE_AF   , .OutputType = GPIOx_PUSH_PULL,
00118         .OutputSpeed = GPIOx_MediumSpeed , .InputType = GPIOx_NoPull ,
00119         .AF_Type   = GPIOx_AF7
00120     } ;
00121
00122     MGPIOx_vInit( &RX6 ) ;
00123
00124 }
00125
00126 // BuadRate/Over-sampling selections:
00127 switch( A_InitStruct->Oversampling )
00128 {
00129
00130     case OVER_SAMPLING_16:
00131
00132         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00133
00134         break;
00135
00136     case OVER_SAMPLING_8:
00137
00138         A_USARTx->BRR_REG = UART_BRR_SAMPLING16( __PCLK__ , A_InitStruct->BaudRate ) ;
00139
00140         break;
00141
00142     }
00143
00144     A_USARTx->CR1_REG = ( A_InitStruct->Oversampling << MUSART_CR1_OVER8_BIT ) |
00145             ( A_InitStruct->DataWidth << MUSART_CR1_M_BIT ) |
00146             ( A_InitStruct->Parity_Enable << MUSART_CR1_PCE_BIT ) |
00147             ( A_InitStruct->Parity_Selection << MUSART_CR1_PS_BIT ) ;
00148
00149     switch (A_InitStruct->TransferDirection)
00150     {
00151
00152         case TX_ONLY:
00153
00154             SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00155
00156             break ;
00157
00158         case RX_ONLY:
00159
00160             SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;
00161
00162             break ;
00163
00164         case TX_RX :
00165
00166             SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_TE_BIT ) ;
00167             SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_RE_BIT ) ;
00168
00169             break ;
00170

```

```

00170     }
00171
00172     A_USARTx->CR2_REG = ( A_InitStruct->StopBits << MUSART_CR2_STOP_BIT )
00173             ( A_ClockInitStruct->ClockOutput << MUSART_CR2_CLKEN_BIT ) |
00174             ( A_ClockInitStruct->ClockPhase << MUSART_CR2_CPHA_BIT ) |
00175             ( A_ClockInitStruct->ClockPolarity << MUSART_CR2_CPOL_BIT ) |
00176             ( A_ClockInitStruct->LastBitClockPulse << MUSART_CR2_LBCL_BIT ) ;
00177
00178     A_USARTx->SR_REG = 0 ; // Clear all the flags.
00179
00180     SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ; // EN the peripheral.
00181
00182 }
00183
00184 /*****
00185 *****/
00186
00187 void MUSART_vEnable( USART_MemoryMapType *A_USARTx )
00188 {
00189     SET_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00190 }
00191
00192 /*****
00193 *****/
00194
00195 void MUSART_vDisable( USART_MemoryMapType *A_USARTx )
00196 {
00197     CLR_BIT( A_USARTx->CR1_REG, MUSART_CR1_UE_BIT ) ;
00198 }
00199
00200 /*****
00201 *****/
00202
00203 void MUSART_vTransmitByte( USART_MemoryMapType *A_USARTx, u8_t A_u8Byte )
00204 {
00205
00206     while( GET_BIT( A_USARTx->SR_REG, MUSART_SR_TXE_BIT ) == 0 ) ;
00207
00208     A_USARTx->DR_REG = A_u8Byte ;
00209
00210     while( GET_BIT( A_USARTx->SR_REG, MUSART_SR_TC_BIT ) == 0 ) ;
00211
00212     // Clear the TC flag:
00213     CLR_BIT( A_USARTx->SR_REG, MUSART_SR_TC_BIT ) ;
00214
00215 }
00216
00217 /*****
00218 *****/
00219
00220 void MUSART_vTransmitString( USART_MemoryMapType *A_USARTx, c8_t *A_ptrc8String )
00221 {
00222
00223     u8_t loc_u8Iterator = INITIAL_ZERO ;
00224
00225     while( A_ptrc8String[loc_u8Iterator] != '\0' )
00226     {
00227
00228         MUSART_vTransmitByte( A_USARTx, A_ptrc8String[loc_u8Iterator] ) ;
00229
00230         loc_u8Iterator++ ;
00231
00232     }
00233
00234 }
00235
00236 /*****
00237 *****/
00238
00239 u8_t MUSART_u8ReceiveByteSyncNonBlocking ( USART_MemoryMapType *A_USARTx )
00240 {
00241
00242     u8_t loc_u8Data     = INITIAL_ZERO ;
00243
00244     u32_t loc_u8TimeOut = INITIAL_ZERO ;
00245
00246     while( (GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0) && (loc_u8TimeOut < THRESHOLD_VALUE) )
00247     {
00248         loc_u8TimeOut++ ;
00249     }
00250
00251     if ( loc_u8TimeOut == THRESHOLD_VALUE )
00252     {
00253         loc_u8Data = 255 ; // To detect the error.
00254     }
00255     else
00256     {

```

```

00257     loc_u8Data = A_USARTx->DR_REG ;
00258 }
00259
00260     return loc_u8Data ;
00261 }
00262 }
00263
00264 /*****
00265 ****
00266
00267 u8_t MUSART_u8ReceiveByteSynchBlocking ( USART_MemoryMapType *A_USARTx )
00268 {
00269
00270     while( GET_BIT(A_USARTx->SR_REG, MUSART_SR_RXNE_BIT) == 0 ) ;
00271
00272     return A_USARTx->DR_REG ;
00273 }
00274
00275
00276 /*****
00277 ****
00278
00279 u8_t * MUSART_ptrReceiveStringSynchNonBlocking( USART_MemoryMapType *A_USARTx )
00280 {
00281
00282     u8_t loc_u8Iterator = INITIAL_ZERO ;
00283
00284     u8_t loc_u8DataCome ;
00285
00286     while( ( loc_u8DataCome = MUSART_u8ReceiveByteSynchNonBlocking(A_USARTx) ) != 13 )
00287     {
00288         G_u8String[ loc_u8Iterator ] = loc_u8DataCome ;
00289
00290         loc_u8Iterator++ ;
00291     }
00292
00293     G_u8String[loc_u8Iterator] = '\0' ;
00294
00295     return( G_u8String ) ;
00296
00297 }
00298
00299 ****
00300 ****
00301
00302 void MUSART_vRecieveString( USART_MemoryMapType *A_USARTx, c8_t A_c8YourString[] )
00303 {
00304
00305     u32_t L_u32Counter = INITIAL_ZERO ;
00306
00307     A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00308
00309     while( A_c8YourString[L_u32Counter] != '\0' )
00310     {
00311
00312         L_u32Counter++ ;
00313
00314         A_c8YourString[L_u32Counter] = MUSART_u8ReceiveByteSynchBlocking( A_USARTx ) ;
00315
00316         if( (A_c8YourString[L_u32Counter] == '\n') || (A_c8YourString[L_u32Counter] == '\r') )
00317         {
00318             A_c8YourString[L_u32Counter] = '\0' ;
00319         }
00320
00321     }
00322
00323 }
00324
00325 ****
00326 ****
00327
00328 u8_t MUSART_u8CompareString( c8_t *String1, c8_t *String2 )
00329 {
00330
00331     u8_t L_u8TheComparingResult = INITIAL_ZERO ;
00332
00333     if( strcmp( String1, String2 ) == SAME_STRING )
00334     {
00335         L_u8TheComparingResult = SAME_STRING ;
00336     }
00337     else
00338     {
00339         L_u8TheComparingResult = DIFFERENT_STRING ;
00340     }
00341
00342     return L_u8TheComparingResult ;
00343

```

```
00344 }
00345
00346 /******
00347 *****/
00348
00349 u8_t MUSART_u8ReadDataRegister( USART_MemoryMapType *A_USARTx )
00350 {
00351     return A_USARTx->DR_REG ;
00352 }
00353
00354 *****/
00355 *****/
00356
00357 void MUSART_vClearFlags( USART_MemoryMapType *A_USARTx )
00358 {
00359     A_USARTx->SR_REG = 0 ;
00360 }
00361
00362 *****/
00363 *****/
00364
00365 void MUSART_vRxIntSetStatus(USART_MemoryMapType *A_USARTx, u8_t A_u8Status)
00366 {
00367
00368     switch(A_u8Status)
00369     {
00370
00371         case ENABLE : SET_BIT( A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00372
00373         case DISABLE: CLR_BIT( A_USARTx->CR1_REG, (MUSART_CR1_RXNEIE_BIT) ); break ;
00374
00375     }
00376
00377 }
00378
00379 *****/
00380 *****/
00381
00382 void MUSART1_vSetCallBack( void (*Fptr) (void) )
00383 {
00384     MUSART1_CallBack = Fptr ;
00385 }
00386
00387 *****/
00388 *****/
00389
00390 void MUSART2_vSetCallBack( void (*Fptr) (void) )
00391 {
00392     MUSART2_CallBack = Fptr ;
00393 }
00394
00395 *****/
00396 *****/
00397
00398 void MUSART6_vSetCallBack( void (*Fptr) (void) )
00399 {
00400     MUSART6_CallBack = Fptr ;
00401 }
00402
00403 *****/
00404 *****/
00405
00406 void USART1_IRQHandler(void)
00407 {
00408
00409     USART1_REG->SR_REG = 0 ;
00410
00411     MUSART1_CallBack( ) ;
00412
00413 }
00414
00415 *****/
00416 *****/
00417
00418 void USART2_IRQHandler(void)
00419 {
00420
00421     USART2_REG -> SR_REG = 0 ;
00422
00423     MUSART2_CallBack( ) ;
00424
00425 }
00426
00427 *****/
00428 *****/
00429
00430 void USART6_IRQHandler(void)
```

```

00431 {
00432     USART6_REG->SR_REG = 0 ;
00433     MUSART6_CallBack( ) ;
00434
00435 }
00436
00437 */
00438
00439 /*****
00440 *****/
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478

```

7.122 COTS/Services/UsersLogin/UsersLogin_config.h File Reference

7.123 UsersLogin_config.h

[Go to the documentation of this file.](#)

```

00001 #ifndef _USERSLOGIN_CONFIG_H
00002 #define _USERSLOGIN_CONFIG_H
00003
00004 #endif // _USERSLOGIN_CONFIG_H

```

7.124 COTS/Services/UsersLogin/UsersLogin_interface.h File Reference

Functions

- `bool_t UsersLogin_vUserLoginProcess (void)`

7.124.1 Function Documentation

7.124.1.1 UsersLogin_vUserLoginProcess()

```
bool_t UsersLogin_vUserLoginProcess (
    void )
```

Definition at line 65 of file [UsersLogin_program.c](#).

```
00066 {
00067     bool_t bHasLoggedIn = FALSE;
00068     c8_t L_strRecieveUsername[MAX_INFO_LENGTH] = "";
00069     c8_t L_strRecievePassword[MAX_INFO_LENGTH] = "";
00070
00071     do
00072     {
00073         HBluetooth_ExchangeString("\nEnter username (max: 10 letters): \n", L_strRecieveUsername);
00074         HBluetooth_ExchangeString("\nEnter password (max: 10 letters): \n", L_strRecievePassword);
00075         bHasLoggedIn = UsersLogin_boolValidateUserInfo(L_strRecieveUsername, L_strRecievePassword);
00076
00077         if (bHasLoggedIn)
00078         {
00079             HBluetooth_vSendString("\nLogged in successfully.\n");
00080         }
00081         else
00082         {
00083             HBluetooth_vSendString("\nCouldn't login, invalid login information.\n");
00084         }
00085     } while (!bHasLoggedIn);
00086
00087     return bHasLoggedIn;
00088 }
```

References [FALSE](#), [HBluetooth_ExchangeString\(\)](#), [HBluetooth_vSendString\(\)](#), and [MAX_INFO_LENGTH](#).

7.125 UsersLogin_interface.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _USERSLOGIN_INTERFACE_H
00002 #define _USERSLOGIN_INTERFACE_H
00003
00004 bool_t UsersLogin_vUserLoginProcess(void);
00005
00006 #endif // _USERSLOGIN_INTERFACE_H
```

7.126 COTS/Services/UsersLogin/UsersLogin_private.h File Reference

Data Structures

- struct [UserInfo_t](#)

Macros

- [#define MAX_INFO_LENGTH \(10\)](#)

7.126.1 Macro Definition Documentation

7.126.1.1 MAX_INFO_LENGTH

```
#define MAX_INFO_LENGTH (10)
```

Definition at line 4 of file [UsersLogin_private.h](#).

7.127 UsersLogin_private.h

[Go to the documentation of this file.](#)

```
00001 #ifndef _USERSLOGIN_PRIVATE_H
00002 #define _USERSLOGIN_PRIVATE_H
00003
00004 #define MAX_INFO_LENGTH (10)
00005
00006 typedef struct
00007 {
00008     c8_t strUsername[MAX_INFO_LENGTH];
00009     c8_t strPassword[MAX_INFO_LENGTH];
00010 } UserInfo_t;
00011
00012 #endif // _USERSLOGIN_PRIVATE_H
```

7.128 COTS/Services/UsersLogin/UsersLogin_program.c File Reference

```
#include "../../LIB/LSTD_TYPES.h"
#include "../../LIB/LSTD_COMPILER.h"
#include "../../LIB/LSTD_VALUES.h"
#include "../../LIB/LSTD_BITMATH.h"
#include "../../HAL/Bluetooth/Bluetooth_interface.h"
#include "UsersLogin_private.h"
#include "UsersLogin_interface.h"
#include "UsersLogin_config.h"
```

Functions

- [bool_t UsersLogin_vUserLoginProcess \(void\)](#)

7.128.1 Function Documentation

7.128.1.1 UsersLogin_vUserLoginProcess()

```
bool_t UsersLogin_vUserLoginProcess (
    void )
```

Definition at line 65 of file [UsersLogin_program.c](#).

```
00066 {
00067     bool_t bHasLoggedIn = FALSE;
00068     c8_t L_strRecieveUsername[MAX_INFO_LENGTH] = "";
00069     c8_t L_strRecievePassword[MAX_INFO_LENGTH] = "";
00070
00071     do
00072     {
00073         HBluetooth_ExchangeString("\nEnter username (max: 10 letters): \n", L_strRecieveUsername);
00074         HBluetooth_ExchangeString("\nEnter password (max: 10 letters): \n", L_strRecievePassword);
00075         bHasLoggedIn = UsersLogin_boolValidateUserInfo(L_strRecieveUsername, L_strRecievePassword);
00076
00077         if (bHasLoggedIn)
00078         {
00079             HBluetooth_vSendString("\nLogged in successfully.\n");
00080         }
00081         else
00082         {
00083             HBluetooth_vSendString("\nCouldn't login, invalid login information.\n");
00084         }
00085     } while (!bHasLoggedIn);
00086
00087     return bHasLoggedIn;
00088 }
```

References [FALSE](#), [HBluetooth_ExchangeString\(\)](#), [HBluetooth_vSendString\(\)](#), and [MAX_INFO_LENGTH](#).

7.129 UsersLogin_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 #include "../../LIB/LSTD_VALUES.h"
00008 #include "../../LIB/LSTD_BITMATH.h"
00009
00010 #include "../../HAL/Bluetooth/Bluetooth_interface.h"
00011
00012 #include "UsersLogin_private.h"
00013 #include "UsersLogin_interface.h"
00014 #include "UsersLogin_config.h"
00015
00016 STATIC FUNC(bool_t) UsersLogin_boolValidateUserInfo(P2VAR(c8_t) L_strUsername, P2VAR(c8_t)
00017     L_strPassword)
00018 {
00019     bool_t bHasCorrectInfo = FALSE;
00020
00021     UserInfo_t L_strUsersInfo[] =
00022     {
00023         { "Mohamed", "Mohamed" },
00024         { "Ahmed", "Ahmed" },
00025         { "Value", "Zagazig" }
00026     };
00027
00028     if (!L_strUsername[0] || !L_strPassword[0])
00029     {
00030         bHasCorrectInfo = FALSE;
00031         // HBluetooth_vSendString(L_strUsername);
00032     }
00033     else
00034     {
00035         for (u8_t L_intLoopCounter = INITIAL_ZERO; L_intLoopCounter < sizeof L_strUsersInfo;
00036             L_intLoopCounter++)
00037         {
00038             if (HBluetooth_u8CompStrings(L_strUsername,
00039                 L_strUsersInfo[L_intLoopCounter].strUsername,
00040                 L_strUsersInfo[L_intLoopCounter].strUsername) == SAME_STRING)
```

```
00039             {
00040                 if (HBluetooth_u8CompStrings(L_strPassword,
00041                     L_strUserInfo[L_intLoopCounter].strPassword) == SAME_STRING)
00042                 {
00043                     bHasCorrectInfo = TRUE;
00044                     break;
00045                 }
00046                 else
00047                 {
00048                     /* Do nothing */
00049                 }
00050             }
00051             else
00052             {
00053                 /* Do nothing */
00054             }
00055         }
00056     }
00057     /* Do nothing */
00058 }
00059 }
00060 }
00061
00062 return bHasCorrectInfo;
00063 }
00064
00065 FUNC(bool_t) UsersLogin_vUserLoginProcess(void)
00066 {
00067     bool_t bHasLoggedIn = FALSE;
00068     c8_t L_strRecieveUsername[MAX_INFO_LENGTH] = "";
00069     c8_t L_strRecievePassword[MAX_INFO_LENGTH] = "";
00070
00071     do
00072     {
00073         HBluetooth_ExchangeString("\nEnter username (max: 10 letters): \n", L_strRecieveUsername);
00074         HBluetooth_ExchangeString("\nEnter password (max: 10 letters): \n", L_strRecievePassword);
00075         bHasLoggedIn = UsersLogin_boolValidateUserInfo(L_strRecieveUsername, L_strRecievePassword);
00076
00077         if (bHasLoggedIn)
00078         {
00079             HBluetooth_vSendString("\nLogged in successfully.\n");
00080         }
00081         else
00082         {
00083             HBluetooth_vSendString("\nCouldn't login, invalid login information.\n");
00084         }
00085     } while (!bHasLoggedIn);
00086
00087     return bHasLoggedIn;
00088 }
00089 }
```

Index

- _10_BITS
 - ADC_private.h, 296
- _10_CONV
 - ADC_private.h, 309
- _112_CYCLES
 - ADC_private.h, 307
- _11_CONV
 - ADC_private.h, 309
- _12_BITS
 - ADC_private.h, 296
- _144_CYCLES
 - ADC_private.h, 307
- _15_CONV
 - ADC_private.h, 309
- _15_CYCLES
 - ADC_private.h, 306
- _16GROUP_NoSub_Priorities
 - Interrupt priority grouping, 51
- _16_CONV
 - ADC_private.h, 309
- _1_CHANNEL
 - ADC_private.h, 296
- _1_CONV
 - ADC_private.h, 307
- _28_CYCLES
 - ADC_private.h, 306
- _2GROUP_8Sub_Priorities
 - Interrupt priority grouping, 52
- _2_CHANNELS
 - ADC_private.h, 297
- _2_CONV
 - ADC_private.h, 308
- _3_CHANNELS
 - ADC_private.h, 297
- _3_CONV
 - ADC_private.h, 308
- _3_CYCLES
 - ADC_private.h, 306
- _480_CYCLES
 - ADC_private.h, 307
- _4GROUP_4Sub_Priorities
 - Interrupt priority grouping, 51
- _4_CHANNELS
 - ADC_private.h, 297
- _4_CONV
 - ADC_private.h, 308
- _56_CYCLES
 - ADC_private.h, 307
- _5_CHANNELS
 - ADC_private.h, 297
- _5_CONV
 - ADC_private.h, 308
- _6_BITS
 - ADC_private.h, 296
- _6_CHANNELS
 - ADC_private.h, 297
- _6_CONV
 - ADC_private.h, 308
- _7_CHANNELS
 - ADC_private.h, 297
- _7_CONV
 - ADC_private.h, 308
- _84_CYCLES
 - ADC_private.h, 307
- _8GROUP_2Sub_Priorities
 - Interrupt priority grouping, 51
- _8_BITS
 - ADC_private.h, 296
- _8_CHANNELS
 - ADC_private.h, 298
- _8_CONV
 - ADC_private.h, 309
- _9_CONV
 - ADC_private.h, 309
- _BAUDRATE__
 - UART_interface.h, 582
- _PCLK__
 - UART_config.h, 574
- ADC
 - ADC_private.h, 262
 - NVIC Vector Table, 71
- ADC1EN
 - MRCC_config.h, 418
- ADC_BASE_ADDRESS
 - ADC_private.h, 262
- ADC_config.h
 - ADC_ON, 233
 - ADC_PRESCALER, 237
 - ANALOG_WDT, 229
 - AWD_CHANNEL, 231
 - AWDIE_MODE, 231
 - AWDSGL_MODE, 230
 - CONT_CONV, 233
 - DATA_ALIGN, 232
 - DISC_CHANNEL_NUM, 230
 - DISC_MODE, 230
 - DMA_DIS_SELECTION, 233
 - DMA_MODE, 233

EOC_SELECTION, 233
 EOCIE_MODE, 231
 INJ_ANALOG_WDT, 230
 INJ_AUTO_MODE, 230
 INJ_DISC_MODE, 230
 INJ_EXT_EVENT_SRC, 232
 INJ_EXTERNAL_TRIGGER, 232
 INJ_START_CONV, 232
 JEOCIE_MODE, 231
 OVERRUN_INT, 229
 REG_EXT_EVENT_SRC, 232
 REG_EXTERNAL_TRIGGER, 232
 REG_SQ_LENGTH, 237
 REG_START_CONV, 231
 RESOLUTION, 229
 SCAN_MODE, 231
 SMP0, 235
 SMP1, 235
 SMP10, 233
 SMP11, 234
 SMP12, 234
 SMP13, 234
 SMP14, 234
 SMP15, 234
 SMP16, 234
 SMP17, 235
 SMP18, 235
 SMP2, 235
 SMP3, 235
 SMP4, 236
 SMP5, 236
 SMP6, 236
 SMP7, 236
 SMP8, 236
 SMP9, 236
 SQ1_BIT_MANIPULATION, 237
 TEMP_SENSOR_AND_VREF, 237
 V_BATTERY, 237
ADC_interface.h
 CHANNEL0, 245
 CHANNEL1, 246
 CHANNEL10, 247
 CHANNEL11, 247
 CHANNEL12, 247
 CHANNEL13, 248
 CHANNEL14, 248
 CHANNEL15, 248
 CHANNEL16, 248
 CHANNEL17, 248
 CHANNEL18, 248
 CHANNEL2, 246
 CHANNEL3, 246
 CHANNEL4, 246
 CHANNEL5, 246
 CHANNEL6, 246
 CHANNEL7, 247
 CHANNEL8, 247
 CHANNEL9, 247
MADC.h
 MADC_u16ConvertToDigital, 253
 MADC_u16GetADCData, 255
 MADC_vDisable, 253
 MADC_vEnable, 253
 MADC_vInit, 249
 MADC_vInitStruct, 255
 MADC_vRegINT_Disable, 253
 MADC_vRegINTEnable, 253
 MADC_vSelectChannel, 254
 MADC_vSelectChannelsOrder, 254
 MADC_vSelectMode, 254
 MADC_vSelectResolution, 254
 MADC_vSelectSampleTime, 254
 MADC_vSetCallBack, 255
 MADC_vStartConversion, 254
ADC_IRQHandler
 ADC_program.c, 322
ADC_MemoryMapType, 103
 CCR, 108
 CR1, 104
 CR2, 104
 DR, 108
 HTR, 106
 JDR1, 107
 JDR2, 107
 JDR3, 107
 JDR4, 108
 JOFR1, 105
 JOFR2, 105
 JOFR3, 105
 JOFR4, 106
 JSQR, 107
 LTR, 106
 SMPR1, 105
 SMPR2, 105
 SQR1, 106
 SQR2, 106
 SQR3, 107
 SR, 104
ADC_ON
 ADC_config.h, 233
ADC_PRESCALER
 ADC_config.h, 237
ADC_private.h
 _10_BITS, 296
 _10_CONV, 309
 _112_CYCLES, 307
 _11_CONV, 309
 _12_BITS, 296
 _144_CYCLES, 307
 _15_CONV, 309
 _15_CYCLES, 306
 _16_CONV, 309
 _1_CHANNEL, 296
 _1_CONV, 307
 _28_CYCLES, 306
 _2_CHANNELS, 297
 _2_CONV, 308

_3_CHANNELS, 297
_3_CONV, 308
_3_CYCLES, 306
_480_CYCLES, 307
_4_CHANNELS, 297
_4_CONV, 308
_56_CYCLES, 307
_5_CHANNELS, 297
_5_CONV, 308
_6_BITS, 296
_6_CHANNELS, 297
_6_CONV, 308
_7_CHANNELS, 297
_7_CONV, 308
_84_CYCLES, 307
_8_BITS, 296
_8_CHANNELS, 298
_8_CONV, 309
_9_CONV, 309
ADC, 262
ADC_BASE_ADDRESS, 262
ADCPRE0, 295
ADCPRE1, 295
ADON, 270
ALIGN, 269
AWD, 263
AWDCH0, 266
AWDCH1, 266
AWDCH2, 266
AWDCH3, 267
AWDCH4, 267
AWDEN, 264
AWDIE, 266
AWDSGL, 265
CONT, 270
DDS, 270
DISABLE, 296
DISCEN, 265
DISCNUM0, 264
DISCNUM1, 264
DISCNUM2, 265
DIV_BY_2, 310
DIV_BY_4, 310
DIV_BY_6, 310
DIV_BY_8, 310
DMA, 270
EIGHT, 299
EIGHTEEN, 301
ELEVEN, 299
ENABLE, 295
EOC, 263
EOCIE, 266
EOCS, 269
EXTEN0, 267
EXTEN1, 267
EXTI_LINE11, 303
EXTI_LINE15, 306
EXTSEL0, 267
EXTSEL1, 268
EXTSEL2, 268
EXTSEL3, 268
FALLING, 303
FIFTEEN, 300
FIVE, 298
FOUR, 298
FOURTEEN, 300
INJ_T2_TRGO, 304
JAUTO, 265
JAWDEN, 264
JDISCEN, 265
JEOC, 263
JEOCIE, 266
JEXTEN0, 268
JEXTEN1, 268
JEXTSEL0, 269
JEXTSEL1, 269
JEXTSEL2, 269
JEXTSEL3, 269
JSTRT, 263
JSWSTART, 268
L0, 281
L1, 281
L2, 281
L3, 281
LEFT, 306
NINE, 299
ON_CHANGE, 303
ONE, 298
OVR, 262
OVRIE, 263
REG_T2_TRGO, 302
RES0, 264
RES1, 264
RIGHT, 306
RISING, 303
SCAN, 265
SEVEN, 299
SEVENTEEN, 300
SIX, 299
SIXTEEN, 300
SMP0_b0, 275
SMP0_b1, 275
SMP0_b2, 276
SMP10_b0, 270
SMP10_b1, 270
SMP10_b2, 271
SMP11_b0, 271
SMP11_b1, 271
SMP11_b2, 271
SMP12_b0, 271
SMP12_b1, 271
SMP12_b2, 272
SMP13_b0, 272
SMP13_b1, 272
SMP13_b2, 272
SMP14_b0, 272

SMP14_b1, 272
SMP14_b2, 273
SMP15_b0, 273
SMP15_b1, 273
SMP15_b2, 273, 274
SMP16_b0, 274
SMP16_b1, 274
SMP16_b2, 274
SMP17_b0, 274
SMP17_b1, 274
SMP17_b2, 275
SMP18_b0, 275
SMP18_b1, 275
SMP18_b2, 275
SMP1_b0, 276
SMP1_b1, 276
SMP1_b2, 276
SMP2_b0, 276
SMP2_b1, 276
SMP2_b2, 277
SMP3_b0, 277
SMP3_b1, 277
SMP3_b2, 277
SMP4_b0, 277
SMP4_b1, 277
SMP4_b2, 278
SMP5_b0, 278
SMP5_b1, 278
SMP5_b2, 278, 279
SMP6_b0, 279
SMP6_b1, 279
SMP6_b2, 279
SMP7_b0, 279
SMP7_b1, 279
SMP7_b2, 280
SMP8_b0, 280
SMP8_b1, 280
SMP8_b2, 280
SMP9_b0, 280
SMP9_b1, 280
SMP9_b2, 281
SQ10_b0, 286
SQ10_b1, 287
SQ10_b2, 287
SQ10_b3, 287
SQ10_b4, 287
SQ11_b0, 286
SQ11_b1, 286
SQ11_b2, 286
SQ11_b3, 286
SQ11_b4, 286
SQ12_b0, 285
SQ12_b1, 285
SQ12_b2, 285
SQ12_b3, 285
SQ12_b4, 285
SQ13_b0, 284
SQ13_b1, 284
SQ13_b2, 284
SQ13_b3, 284
SQ13_b4, 285
SQ14_b0, 283
SQ14_b1, 283
SQ14_b2, 283
SQ14_b3, 284
SQ14_b4, 284
SQ15_b0, 282
SQ15_b1, 282
SQ15_b2, 283
SQ15_b3, 283
SQ15_b4, 283
SQ16_b0, 281
SQ16_b1, 282
SQ16_b2, 282
SQ16_b3, 282
SQ16_b4, 282
SQ1_b0, 294
SQ1_b1, 294
SQ1_b2, 294
SQ1_b3, 294
SQ1_b4, 295
SQ2_b0, 293
SQ2_b1, 293
SQ2_b2, 293
SQ2_b3, 294
SQ2_b4, 294
SQ3_b0, 292
SQ3_b1, 292
SQ3_b2, 293
SQ3_b3, 293
SQ3_b4, 293
SQ4_b0, 291
SQ4_b1, 292
SQ4_b2, 292
SQ4_b3, 292
SQ4_b4, 292
SQ5_b0, 291
SQ5_b1, 291
SQ5_b2, 291
SQ5_b3, 291
SQ5_b4, 291
SQ6_b0, 290
SQ6_b1, 290
SQ6_b2, 290
SQ6_b3, 290
SQ6_b4, 290
SQ7_b0, 289
SQ7_b1, 289
SQ7_b2, 289
SQ7_b3, 289
SQ7_b4, 290
SQ8_b0, 288
SQ8_b1, 288
SQ8_b2, 288
SQ8_b3, 289
SQ8_b4, 289

SQ9_b0, 287
SQ9_b1, 287
SQ9_b2, 288
SQ9_b3, 288
SQ9_b4, 288
STRT, 263
SWSTART, 267
T1_CC1, 301
T1_CC2, 301
T1_CC3, 301
T1_CC4, 304
T1_TRGO, 304
T2_CC1, 304
T2_CC2, 301
T2_CC3, 301
T2_CC4, 302
T3_CC1, 302
T3_CC2, 304
T3_CC4, 304
T3_TRGO, 302
T4_CC1, 305
T4_CC2, 305
T4_CC3, 305
T4_CC4, 302
T4_TRGO, 305
T5_CC1, 302
T5_CC2, 303
T5_CC3, 303
T5_CC4, 305
T5_TRGO, 305
TEN, 299
THERTEEN, 300
THREE, 298
TSVREFE, 295
TWELVE, 300
VBATE, 295
ZERO, 298
ADC_program.c
 ADC_IRQHandler, 322
 MADC_u16ConvertToDigital, 320
 MADC_vDisable, 321
 MADC_vEnable, 321
 MADC_vInit, 316
 MADC_vpPointerToFunction, 323
 MADC_vRegINT_Disable, 321
 MADC_vRegINTEnable, 322
 MADC_vSelectChannel, 320
 MADC_vSetCallBack, 322
ADCPRE0
 ADC_private.h, 295
ADCPRE1
 ADC_private.h, 295
ADON
 ADC_private.h, 270
AF_Type
 GPIOx_ConfigType, 119
AFRHx
 GPIOx_MemoryMapType, 116
AFRLx
 GPIOx_MemoryMapType, 116
AHB
 Systick Clock Sources, 100
AHB1ENR
 RCC_MemoryMapType, 129
AHB1ENR_CRCEN
 ID options, 88
AHB1ENR_DMA1EN
 ID options, 87
AHB1ENR_DMA2EN
 ID options, 87
AHB1ENR_GPIOAEN
 ID options, 89
AHB1ENR_GPIOBEN
 ID options, 89
AHB1ENR_GPIOCEN
 ID options, 89
AHB1ENR_GPIODEN
 ID options, 88
AHB1ENR_GPIOEEN
 ID options, 88
AHB1ENR_GPIOHEN
 ID options, 88
AHB1LPENR
 RCC_MemoryMapType, 131
AHB1LPENR_FLITFLPEN
 ID options, 95
AHB1RSTR
 RCC_MemoryMapType, 128
AHB2ENR
 RCC_MemoryMapType, 130
AHB2ENR_OTGFSEN
 ID options, 89
AHB2LPENR
 RCC_MemoryMapType, 131
AHB2RSTR
 RCC_MemoryMapType, 128
AHB_DividedBy8
 Systick Clock Sources, 100
AHBby16
 MRCC_private.h, 467
AHBby2
 MRCC_private.h, 466
AHBby4
 MRCC_private.h, 466
AHBby8
 MRCC_private.h, 466
AIRCR
 SCB_MemoryMapType, 136
ALIGN
 ADC_private.h, 269
ANALOG_WDT
 ADC_config.h, 229
APB1ENR
 RCC_MemoryMapType, 130
APB1ENR_I2C1EN
 ID options, 90

APB1ENR_I2C2EN
 ID options, 90
 APB1ENR_I2C3EN
 ID options, 90
 APB1ENR_PWREN
 ID options, 90
 APB1ENR_SPI2EN
 ID options, 91
 APB1ENR_SPI3EN
 ID options, 91
 APB1ENR_TIM2EN
 ID options, 92
 APB1ENR_TIM3EN
 ID options, 92
 APB1ENR_TIM4EN
 ID options, 92
 APB1ENR_TIM5EN
 ID options, 92
 APB1ENR_USART2EN
 ID options, 91
 APB1ENR_WWDGEN
 ID options, 91
 APB1LPENR
 RCC_MemoryMapType, 132
 APB1RSTR
 RCC_MemoryMapType, 129
 APB2ENR
 RCC_MemoryMapType, 130
 APB2ENR_ADC1EN
 ID options, 94
 APB2ENR_SDIOEN
 ID options, 94
 APB2ENR_SPI1EN
 ID options, 94
 APB2ENR_SPI4EN
 ID options, 94
 APB2ENR_SYSCFGEN
 ID options, 93
 APB2ENR_TIM10EN
 ID options, 93
 APB2ENR_TIM11EN
 ID options, 93
 APB2ENR_TIM1EN
 ID options, 95
 APB2ENR_TIM9EN
 ID options, 93
 APB2ENR_USART1EN
 ID options, 95
 APB2ENR_USART6EN
 ID options, 95
 APB2LPENR
 RCC_MemoryMapType, 132
 APB2RSTR
 RCC_MemoryMapType, 129
 AssertRequest
 Systick Exception (Interrupt) Status, 101
 AWD
 ADC_private.h, 263
 AWD_CHANNEL
 ADC_config.h, 231
 AWDCH0
 ADC_private.h, 266
 AWDCH1
 ADC_private.h, 266
 AWDCH2
 ADC_private.h, 266
 AWDCH3
 ADC_private.h, 267
 AWDCH4
 ADC_private.h, 267
 AWDEN
 ADC_private.h, 264
 AWDIE
 ADC_private.h, 266
 AWDIE_MODE
 ADC_config.h, 231
 AWDSGL
 ADC_private.h, 265
 AWDSGL_MODE
 ADC_config.h, 230
 BACKWARD
 DC Motor Directions, 9
 BaudRate
 USART_InitType, 146
 BDCR
 RCC_MemoryMapType, 133
 BFAR
 SCB_MemoryMapType, 140
 BIDIMODE
 SPI_private.h, 520
 BIDIOE
 SPI_private.h, 519
 BIDIOE_MODE
 SPI_config.h, 497
 BIDMODE_MODE
 SPI_config.h, 497
 Bit Group Manipulation Math Macros, 12
 CLR_BITs, 12
 GET_BITs, 13
 SET_BITs, 12
 TOGGLE_BITs, 13
 Bit Manipulation Math Macros, 10
 CLR_BIT, 11
 GET_BIT, 11
 SET_BIT, 11
 TOGGLE_BIT, 11
 Bluetooth_CLK
 Bluetooth_config.c, 151
 Bluetooth_program.c, 160
 Bluetooth_Config
 Bluetooth_config.c, 151
 Bluetooth_program.c, 160
 Bluetooth_config.c
 Bluetooth_CLK, 151
 Bluetooth_Config, 151
 Bluetooth_UART_ID, 152

Bluetooth_interface.h
HBluetooth_ExchangeByte, 155
HBluetooth_ExchangeString, 155
HBluetooth_u8CompStrings, 154
HBluetooth_u8ReceiveByte, 154
HBluetooth_vDisable, 155
HBluetooth_vEnable, 154
HBluetooth_vInit, 153
HBluetooth_vReceiveString, 154
HBluetooth_vSendByte, 153
HBluetooth_vSendString, 154

Bluetooth_program.c
Bluetooth_CLK, 160
Bluetooth_Config, 160
Bluetooth_UART_ID, 160
HBluetooth_ExchangeByte, 159
HBluetooth_ExchangeString, 159
HBluetooth_u8CompStrings, 158
HBluetooth_u8ReceiveByte, 157
HBluetooth_vDisable, 159
HBluetooth_vEnable, 159
HBluetooth_vInit, 157
HBluetooth_vReceiveString, 158
HBluetooth_vSendByte, 157
HBluetooth_vSendString, 158

Bluetooth_UART_ID
Bluetooth_config.c, 152
Bluetooth_program.c, 160

bool_t
Standard types, 17

BR_MODE
SPI_config.h, 496

BRR_REG
USART_MemoryMapType, 148

BSRRx
GPIOx_MemoryMapType, 115

BSY
SPI_private.h, 522

BSY_MODE
SPI_config.h, 499

BUS_FAULT
NVIC Settable Priorities, 84

BUSY
SPI_private.h, 529

BUSY_TICK_TIME
SysTick_interface.h, 553

BYBASED
MRCC_private.h, 457

c8_t
Standard types, 18

CALIB
SysTick_Type, 143

CarControl_interface.h
HCarControl_vInitCar, 163
HCarControl_vMoveBackward, 163
HCarControl_vMoveForward, 163
HCarControl_vStopCar, 164
HCarControl_vTurnLeft, 164

HCarControl_vTurnRight, 164

CarControl_program.c
dcmMotor1, 167
dcmMotor2, 168
dcmMotor3, 168
dcmMotor4, 168
HCarControl_vInitCar, 166
HCarControl_vMoveBackward, 166
HCarControl_vMoveForward, 166
HCarControl_vStopCar, 167
HCarControl_vTurnLeft, 167
HCarControl_vTurnRight, 166

CCR
ADC_MemoryMapType, 108
SCB_MemoryMapType, 137

CCW
DC Motor Rotation Directions, 10

CFGR
RCC_MemoryMapType, 127

CFSR
SCB_MemoryMapType, 138

CGRAM_AddressOfPattern0
LCD_interface.h, 190

CGRAM_AddressOfPattern1
LCD_interface.h, 190

CGRAM_AddressOfPattern2
LCD_interface.h, 190

CGRAM_AddressOfPattern3
LCD_interface.h, 190

CGRAM_AddressOfPattern4
LCD_interface.h, 190

CGRAM_AddressOfPattern5
LCD_interface.h, 191

CGRAM_AddressOfPattern7
LCD_interface.h, 191

CGRAM_AddressOfPattern8
LCD_interface.h, 191

CHANNEL0
ADC_interface.h, 245

CHANNEL1
ADC_interface.h, 246

CHANNEL10
ADC_interface.h, 247

CHANNEL11
ADC_interface.h, 247

CHANNEL12
ADC_interface.h, 247

CHANNEL13
ADC_interface.h, 248

CHANNEL14
ADC_interface.h, 248

CHANNEL15
ADC_interface.h, 248

CHANNEL16
ADC_interface.h, 248

CHANNEL17
ADC_interface.h, 248

CHANNEL18

ADC_interface.h, 248
 CHANNEL2
 ADC_interface.h, 246
 CHANNEL3
 ADC_interface.h, 246
 CHANNEL4
 ADC_interface.h, 246
 CHANNEL5
 ADC_interface.h, 246
 CHANNEL6
 ADC_interface.h, 246
 CHANNEL7
 ADC_interface.h, 247
 CHANNEL8
 ADC_interface.h, 247
 CHANNEL9
 ADC_interface.h, 247
 CHSIDE
 SPI_private.h, 521
 CIR
 RCC_MemoryMapType, 128
 CLK_By128
 SPI_private.h, 524
 CLK_By16
 SPI_private.h, 524
 CLK_By2
 SPI_private.h, 523
 CLK_By256
 SPI_private.h, 525
 CLK_By32
 SPI_private.h, 524
 CLK_By4
 SPI_private.h, 524
 CLK_By64
 SPI_private.h, 524
 CLK_By8
 SPI_private.h, 524
 CLK_IdleAt0
 SPI_private.h, 523
 CLK_IdleAt1
 SPI_private.h, 523
 CLK_SOURCE
 GPIO Addresses, 49
 CLKSOURCE
 Systick Register Bits Positions, 99
 ClockOutput
 USART_ClockInitTypeDef, 144
 ClockPhase
 USART_ClockInitTypeDef, 145
 ClockPolarity
 USART_ClockInitTypeDef, 144
 CLR_BIT
 Bit Manipulation Math Macros, 11
 CLR_BITS
 Bit Group Manipulation Math Macros, 12
 CMPPCR
 MSYSCFG_MemMap_t, 120
 Compiler standard macros, 14
 CONST, 16
 CONSTP2CONST, 15
 CONSTP2VAR, 15
 FUNC, 14
 P2CONST, 15
 P2FUNC, 15
 P2VAR, 14
 STATIC, 16
 VAR, 14
 CONST
 Compiler standard macros, 16
 CONSTP2CONST
 Compiler standard macros, 15
 CONSTP2VAR
 Compiler standard macros, 15
 CONT
 ADC_private.h, 270
 CONT_CONV
 ADC_config.h, 233
 COTS/HAL/Bluetooth/Bluetooth_config.c, 151, 152
 COTS/HAL/Bluetooth/Bluetooth_config.h, 153
 COTS/HAL/Bluetooth/Bluetooth_interface.h, 153, 156
 COTS/HAL/Bluetooth/Bluetooth_private.h, 156
 COTS/HAL/Bluetooth/Bluetooth_program.c, 156, 161
 COTS/HAL/CarControl/CarControl_config.h, 163
 COTS/HAL/CarControl/CarControl_interface.h, 163, 165
 COTS/HAL/CarControl/CarControl_private.h, 165
 COTS/HAL/CarControl/CarControl_program.c, 165, 169
 COTS/HAL/DCMOTOR/DCM_config.h, 170, 171
 COTS/HAL/DCMOTOR/DCM_interface.h, 171, 173
 COTS/HAL/DCMOTOR/DCM_private.h, 174, 175
 COTS/HAL/DCMOTOR/DCM_program.c, 175, 177
 COTS/HAL/LCD/LCD_config.h, 178, 184
 COTS/HAL/LCD/LCD_interface.h, 185, 197
 COTS/HAL/LCD/LCD_private.h, 198, 200
 COTS/HAL/LCD/LCD_program.c, 201, 209
 COTS/HAL/LDR/LDR_config.c, 215, 216
 COTS/HAL/LDR/LDR_config.h, 216, 217
 COTS/HAL/LDR/LDR_interface.h, 217, 219
 COTS/HAL/LDR/LDR_private.h, 220
 COTS/HAL/LDR/LDR_program.c, 220, 222
 COTS/LIB/LSTD_BITMATH.h, 222, 223
 COTS/LIB/LSTD_COMPILER.h, 223, 224
 COTS/LIB/LSTD MCU UTILITIES.h, 225
 COTS/LIB/LSTD_TYPES.h, 225, 226
 COTS/LIB/LSTD_VALUES.h, 226, 227
 COTS/MCAL/ADC/ADC_config.h, 228, 238
 COTS/MCAL/ADC/ADC_interface.h, 245, 255
 COTS/MCAL/ADC/ADC_private.h, 257, 311
 COTS/MCAL/ADC/ADC_program.c, 316, 323
 COTS/MCAL/EXTI/EXTI_config.h, 328, 335
 COTS/MCAL/EXTI/EXTI_interface.h, 337, 344
 COTS/MCAL/EXTI/EXTI_private.h, 345, 346
 COTS/MCAL/EXTI/EXTI_program.c, 346, 355
 COTS/MCAL/EXTI/SYSCFG_private.h, 360, 361
 COTS/MCAL/GPIO/GPIO_config.h, 361, 363
 COTS/MCAL/GPIO/GPIO_interface.h, 363, 372
 COTS/MCAL/GPIO/GPIO_private.h, 374, 375

COTS/MCAL/GPIO/GPIO_program.c, 375, 384
COTS/MCAL/NVIC/NVIC_config.h, 390
COTS/MCAL/NVIC/NVIC_interface.h, 390, 398
COTS/MCAL/NVIC/NVIC_private.h, 399, 401
COTS/MCAL/NVIC/NVIC_program.c, 402, 406
COTS/MCAL/RCC/MRCC_config.h, 408, 419
COTS/MCAL/RCC/MRCC_interface.h, 427, 434
COTS/MCAL/RCC/MRCC_private.h, 435, 478
COTS/MCAL/RCC/MRCC_program.c, 482, 489
COTS/MCAL/SPI/SPI_config.h, 495, 501
COTS/MCAL/SPI/SPI_interface.h, 504, 515
COTS/MCAL/SPI/SPI_private.h, 516, 531
COTS/MCAL/SPI/SPI_program.c, 532, 543
COTS/MCAL/SysTick/SysTick_config.h, 550, 551
COTS/MCAL/SysTick/SysTick_interface.h, 551, 559
COTS/MCAL/SysTick/SysTick_private.h, 559, 560
COTS/MCAL/SysTick/SysTick_program.c, 561, 569
COTS/MCAL/UART/UART_config.h, 574, 577
COTS/MCAL/UART/UART_interface.h, 578, 591
COTS/MCAL/UART/UART_private.h, 593, 604
COTS/MCAL/UART/UART_program.c, 606, 617
COTS/Services/UsersLogin/UsersLogin_config.h, 622
COTS/Services/UsersLogin/UsersLogin_interface.h,
 622, 623
COTS/Services/UsersLogin/UsersLogin_private.h, 623,
 624
COTS/Services/UsersLogin/UsersLogin_program.c,
 624, 625
COUNTER_ENABLE
 Systick Register Bits Positions, 99
COUNTFLAG
 Systick Register Bits Positions, 99
CPHA
 SPI_private.h, 518
CPHA_MODE
 SPI_config.h, 495
CPOL
 SPI_private.h, 518
CPOL_MODE
 SPI_config.h, 496
CPUID
 SCB_MemoryMapType, 135
CR
 RCC_MemoryMapType, 127
CR1
 ADC_MemoryMapType, 104
 SPI_MemoryMapType, 141
CR1_REG
 USART_MemoryMapType, 148
CR2
 ADC_MemoryMapType, 104
 SPI_MemoryMapType, 141
CR2_REG
 USART_MemoryMapType, 149
CR3_REG
 USART_MemoryMapType, 149
CRC_PHASE
 SPI_private.h, 526
CRCCLK
 MRCC_config.h, 414
CRCEN
 SPI_private.h, 519
CRCEN_MODE
 SPI_config.h, 497
CRCERR
 SPI_private.h, 522
CRCERR_MODE
 SPI_config.h, 500
CRCNEXT
 SPI_private.h, 519
CRCNEXT_MODE
 SPI_config.h, 497
CRCPR
 SPI_MemoryMapType, 141
CSR
 RCC_MemoryMapType, 133
CSS
 MRCC_config.h, 410
CTRL
 SysTick_Type, 143
CW
 DC Motor Rotation Directions, 10
D0_PIN
 LCD_config.h, 180
D1_PIN
 LCD_config.h, 180
D2_PIN
 LCD_config.h, 180
D3_PIN
 LCD_config.h, 180
D4_PIN
 LCD_config.h, 181
D5_PIN
 LCD_config.h, 181
D6_PIN
 LCD_config.h, 181
D7_PIN
 LCD_config.h, 181
D:/GProject/ADAS_Project/README.md, 151
DARK
 LDR_interface.h, 218
DATA_ALIGN
 ADC_config.h, 232
DataWidth
 USART_InitType, 146
DC Motor Directions, 9
 BACKWARD, 9
 FORWARD, 9
DC Motor Rotation Directions, 10
 CCW, 10
 CW, 10
DCKCFG
 RCC_MemoryMapType, 134
DCM_interface.h
 HDCM_vInitMotor, 172
 HDCM_vMotorSpeedCntrl, 173

HDCM_vMoveBackward, 173
HDCM_vMoveForward, 172
HDCM_vStopMotor, 173
DCM_MotorConfiguration, 108
 u8Direction, 110
 u8Pin1, 109
 u8Pin2, 109
 u8Port, 109
DCM_program.c
 HDCM_vInitMotor, 176
 HDCM_vMoveBackward, 176
 HDCM_vMoveForward, 176
 HDCM_vStopMotor, 177
dcmMotor1
 CarControl_program.c, 167
dcmMotor2
 CarControl_program.c, 168
dcmMotor3
 CarControl_program.c, 168
dcmMotor4
 CarControl_program.c, 168
DDS
 ADC_private.h, 270
Delay Units, 97
 MICRO_SEC, 97
 MILLI_SEC, 97
 SEC, 97
DFF
 SPI_private.h, 519
DFF_MODE
 SPI_config.h, 497
DIFFERENT_STRING
 Standard values, 23
DISABLE
 ADC_private.h, 296
 EXTI Lines Status, 31
 MRCC_private.h, 457
 SPI_private.h, 525
 UART_interface.h, 582
DISC_CHANNEL_NUM
 ADC_config.h, 230
DISC_MODE
 ADC_config.h, 230
DISCEN
 ADC_private.h, 265
DISCNUM0
 ADC_private.h, 264
DISCNUM1
 ADC_private.h, 264
DISCNUM2
 ADC_private.h, 265
DispClearCmd
 LCD_private.h, 199
DispCursorWithBlinkingCmd
 LCD_private.h, 199
DispOffCmd
 LCD_private.h, 199
DispOnOffCTRLCmd

LCD_private.h, 199
DIV_BY_2
 ADC_private.h, 310
DIV_BY_4
 ADC_private.h, 310
DIV_BY_6
 ADC_private.h, 310
DIV_BY_8
 ADC_private.h, 310
DivisionBy2
 MRCC_private.h, 458
DivisionBy3
 MRCC_private.h, 458
DivisionBy4
 MRCC_private.h, 458
DivisionBy5
 MRCC_private.h, 459
DMA
 ADC_private.h, 270
DMA1_STREAM0
 NVIC Vector Table, 70
DMA1_STREAM1
 NVIC Vector Table, 70
DMA1_STREAM2
 NVIC Vector Table, 70
DMA1_STREAM3
 NVIC Vector Table, 70
DMA1_STREAM4
 NVIC Vector Table, 71
DMA1_STREAM5
 NVIC Vector Table, 71
DMA1_STREAM6
 NVIC Vector Table, 71
DMA1_STREAM7
 NVIC Vector Table, 76
DMA1CLK
 MRCC_config.h, 413
DMA2_STREAM0
 NVIC Vector Table, 77
DMA2_STREAM1
 NVIC Vector Table, 78
DMA2_STREAM2
 NVIC Vector Table, 78
DMA2_STREAM3
 NVIC Vector Table, 78
DMA2_STREAM4
 NVIC Vector Table, 79
DMA2_STREAM5
 NVIC Vector Table, 78
DMA2_STREAM6
 NVIC Vector Table, 79
DMA2_STREAM7
 NVIC Vector Table, 79
DMA2CLK
 MRCC_config.h, 413
DMA_DIS_SELECTION
 ADC_config.h, 233
DMA_MODE

ADC_config.h, 233
Dont AssertRequest
 Systick Exception (Interrupt) Status, 101
DR
 ADC_MemoryMapType, 108
 SPI_MemoryMapType, 141
DR_REG
 USART_MemoryMapType, 148

EIGHT
 ADC_private.h, 299
EIGHT_BITS
 SPI_private.h, 526
EIGHTEEN
 ADC_private.h, 301
ELEVEN
 ADC_private.h, 299
EMPTY
 SPI_private.h, 527
EMR
 EXTI_Type, 112
ENABLE
 ADC_private.h, 295
 EXTI Lines Status, 30
 MRCC_private.h, 457
 SPI_private.h, 525
 UART_interface.h, 582
EntryModeSet_Cmd
 LCD_private.h, 199
EntryModeSet_ShiftLeftOn_Cmd
 LCD_private.h, 200
EOC
 ADC_private.h, 263
EOC_SELECTION
 ADC_config.h, 233
EOCIE
 ADC_private.h, 266
EOCIE_MODE
 ADC_config.h, 231
EOCS
 ADC_private.h, 269
Equal_0
 MRCC_private.h, 467
Equal_1
 MRCC_private.h, 467
Equal_10
 MRCC_private.h, 468
Equal_11
 MRCC_private.h, 469
Equal_12
 MRCC_private.h, 469
Equal_13
 MRCC_private.h, 469
Equal_14
 MRCC_private.h, 469
Equal_15
 MRCC_private.h, 469
Equal_16
 MRCC_private.h, 469
Equal_17
 MRCC_private.h, 470
Equal_18
 MRCC_private.h, 470
Equal_19
 MRCC_private.h, 470
Equal_2
 MRCC_private.h, 467
Equal_20
 MRCC_private.h, 470
Equal_21
 MRCC_private.h, 470
Equal_22
 MRCC_private.h, 470
Equal_23
 MRCC_private.h, 471
Equal_24
 MRCC_private.h, 471
Equal_25
 MRCC_private.h, 471
Equal_26
 MRCC_private.h, 471
Equal_27
 MRCC_private.h, 471
Equal_28
 MRCC_private.h, 471
Equal_29
 MRCC_private.h, 472
Equal_3
 MRCC_private.h, 467
Equal_30
 MRCC_private.h, 472
Equal_31
 MRCC_private.h, 472
Equal_32
 MRCC_private.h, 472
Equal_33
 MRCC_private.h, 472
Equal_34
 MRCC_private.h, 472
Equal_35
 MRCC_private.h, 473
Equal_36
 MRCC_private.h, 473
Equal_37
 MRCC_private.h, 473
Equal_38
 MRCC_private.h, 473
Equal_39
 MRCC_private.h, 473
Equal_4
 MRCC_private.h, 467
Equal_40
 MRCC_private.h, 473
Equal_41
 MRCC_private.h, 474
Equal_42
 MRCC_private.h, 474

Equal_43
 MRCC_private.h, 474
Equal_44
 MRCC_private.h, 474
Equal_45
 MRCC_private.h, 474
Equal_46
 MRCC_private.h, 474
Equal_47
 MRCC_private.h, 475
Equal_48
 MRCC_private.h, 475
Equal_49
 MRCC_private.h, 475
Equal_5
 MRCC_private.h, 468
Equal_50
 MRCC_private.h, 475
Equal_51
 MRCC_private.h, 475
Equal_52
 MRCC_private.h, 475
Equal_53
 MRCC_private.h, 476
Equal_54
 MRCC_private.h, 476
Equal_55
 MRCC_private.h, 476
Equal_56
 MRCC_private.h, 476
Equal_57
 MRCC_private.h, 476
Equal_58
 MRCC_private.h, 476
Equal_59
 MRCC_private.h, 477
Equal_6
 MRCC_private.h, 468
Equal_60
 MRCC_private.h, 477
Equal_61
 MRCC_private.h, 477
Equal_62
 MRCC_private.h, 477
Equal_63
 MRCC_private.h, 477
Equal_7
 MRCC_private.h, 468
Equal_8
 MRCC_private.h, 468
Equal_9
 MRCC_private.h, 468
ERRIE
 SPI_private.h, 520
ERRIE_MODE
 SPI_config.h, 499
ERROR
 SPI_private.h, 529

EVEN_PARITY
 UART_interface.h, 581
Exception_Request
 SysTick_config.h, 550
EXTENO
 ADC_private.h, 267
EXTEN1
 ADC_private.h, 267
EXTI Line Settings, 31
 EXTI_MAX_EXTI_NUM, 31
EXTI Lines Status, 30
 DISABLE, 31
 ENABLE, 30
EXTI Memory Addresses, 29
 EXTI_BASE_ADDRESS, 29
EXTI Registers, 30
 MEXTI, 30
EXTI0
 NVIC Vector Table, 68
EXTI0_IRQHandler
 EXTI_program.c, 352
EXTI1
 NVIC Vector Table, 69
EXTI15_10
 NVIC Vector Table, 76
EXTI15_10_IRQHandler
 EXTI_program.c, 354
EXTI16
 NVIC Vector Table, 67
EXTI17
 NVIC Vector Table, 76
EXTI18
 NVIC Vector Table, 76
EXTI1_IRQHandler
 EXTI_program.c, 353
EXTI2
 NVIC Vector Table, 69
EXTI21
 NVIC Vector Table, 67
EXTI22
 NVIC Vector Table, 68
EXTI2_IRQHandler
 EXTI_program.c, 353
EXTI3
 NVIC Vector Table, 69
EXTI3_IRQHandler
 EXTI_program.c, 353
EXTI4
 NVIC Vector Table, 69
EXTI4_IRQHandler
 EXTI_program.c, 354
EXTI9
 NVIC Vector Table, 72
EXTI9_5_IRQHandler
 EXTI_program.c, 354
EXTI_BASE_ADDRESS
 EXTI Memory Addresses, 29
EXTI_config.h

EXTI_LINE0_EN, 329
EXTI_LINE0_TRIGGER, 329
EXTI_LINE10_EN, 332
EXTI_LINE10_TRIGGER, 333
EXTI_LINE11_EN, 333
EXTI_LINE11_TRIGGER, 333
EXTI_LINE12_EN, 333
EXTI_LINE12_TRIGGER, 333
EXTI_LINE13_EN, 333
EXTI_LINE13_TRIGGER, 334
EXTI_LINE14_EN, 334
EXTI_LINE14_TRIGGER, 334
EXTI_LINE15_EN, 334
EXTI_LINE15_TRIGGER, 334
EXTI_LINE1_EN, 329
EXTI_LINE1_TRIGGER, 330
EXTI_LINE2_EN, 330
EXTI_LINE2_TRIGGER, 330
EXTI_LINE3_EN, 330
EXTI_LINE3_TRIGGER, 330
EXTI_LINE4_EN, 330
EXTI_LINE4_TRIGGER, 331
EXTI_LINE5_EN, 331
EXTI_LINE5_TRIGGER, 331
EXTI_LINE6_EN, 331
EXTI_LINE6_TRIGGER, 331
EXTI_LINE7_EN, 331
EXTI_LINE7_TRIGGER, 332
EXTI_LINE8_EN, 332
EXTI_LINE8_TRIGGER, 332
EXTI_LINE9_EN, 332
EXTI_LINE9_TRIGGER, 332
EXTI_ConfigType, 110
LineNum, 111
PortNum, 111
TriggerStatus, 111
EXTI_FallingEdge
Interrupt trigger status, 28
EXTI_interface.h
MEXTI_vDisableLine, 342
MEXTI_vEnableLine, 342
MEXTI_vInit, 339
MEXTI_vInit_WithStruct, 342
MEXTI_vSetCallback, 343
MEXTI_vSetTrigger, 343
MEXTI_vSWITrigger, 343
MSYSCFG_vSetEXTIPort, 343
EXTI IRQs
EXTI_private.h, 345
EXTI_LINE0
Interrupt line IDs, 24
EXTI_LINE0_EN
EXTI_config.h, 329
EXTI_LINE0_TRIGGER
EXTI_config.h, 329
EXTI_LINE1
Interrupt line IDs, 24
EXTI_LINE10
Interrupt line IDs, 27
EXTI_LINE10_EN
EXTI_config.h, 332
EXTI_LINE10_TRIGGER
EXTI_config.h, 333
EXTI_LINE11
ADC_private.h, 303
Interrupt line IDs, 27
EXTI_LINE11_EN
EXTI_config.h, 333
EXTI_LINE11_TRIGGER
EXTI_config.h, 333
EXTI_LINE12
Interrupt line IDs, 27
EXTI_LINE12_EN
EXTI_config.h, 333
EXTI_LINE12_TRIGGER
EXTI_config.h, 333
EXTI_LINE13
Interrupt line IDs, 27
EXTI_LINE13_EN
EXTI_config.h, 333
EXTI_LINE13_TRIGGER
EXTI_config.h, 334
EXTI_LINE14
Interrupt line IDs, 28
EXTI_LINE14_EN
EXTI_config.h, 334
EXTI_LINE14_TRIGGER
EXTI_config.h, 334
EXTI_LINE15
ADC_private.h, 306
Interrupt line IDs, 28
EXTI_LINE15_EN
EXTI_config.h, 334
EXTI_LINE15_TRIGGER
EXTI_config.h, 334
EXTI_LINE1_EN
EXTI_config.h, 329
EXTI_LINE1_TRIGGER
EXTI_config.h, 330
EXTI_LINE2
Interrupt line IDs, 25
EXTI_LINE2_EN
EXTI_config.h, 330
EXTI_LINE2_TRIGGER
EXTI_config.h, 330
EXTI_LINE3
Interrupt line IDs, 25
EXTI_LINE3_EN
EXTI_config.h, 330
EXTI_LINE3_TRIGGER
EXTI_config.h, 330
EXTI_LINE4
Interrupt line IDs, 25
EXTI_LINE4_EN
EXTI_config.h, 330
EXTI_LINE4_TRIGGER

EXTI_config.h, 331
EXTI_LINE5
 Interrupt line IDs, 25
EXTI_LINE5_EN
 EXTI_config.h, 331
EXTI_LINE5_TRIGGER
 EXTI_config.h, 331
EXTI_LINE6
 Interrupt line IDs, 26
EXTI_LINE6_EN
 EXTI_config.h, 331
EXTI_LINE6_TRIGGER
 EXTI_config.h, 331
EXTI_LINE7
 Interrupt line IDs, 26
EXTI_LINE7_EN
 EXTI_config.h, 331
EXTI_LINE7_TRIGGER
 EXTI_config.h, 332
EXTI_LINE8
 Interrupt line IDs, 26
EXTI_LINE8_EN
 EXTI_config.h, 332
EXTI_LINE8_TRIGGER
 EXTI_config.h, 332
EXTI_LINE9
 Interrupt line IDs, 26
EXTI_LINE9_EN
 EXTI_config.h, 332
EXTI_LINE9_TRIGGER
 EXTI_config.h, 332
EXTI_MAX_EXTI_NUM
 EXTI Line Settings, 31
EXTI_OnChange
 Interrupt trigger status, 29
EXTI_private.h
 EXTI IRQs, 345
EXTI_program.c
 EXTI0_IRQHandler, 352
 EXTI15_10_IRQHandler, 354
 EXTI1_IRQHandler, 353
 EXTI2_IRQHandler, 353
 EXTI3_IRQHandler, 353
 EXTI4_IRQHandler, 354
 EXTI9_5_IRQHandler, 354
 MEXTI_vDisableLine, 351
 MEXTI_vEnableLine, 350
 MEXTI_vInit, 347
 MEXTI_vInit_WithStruct, 350
 MEXTI_vSetCallback, 352
 MEXTI_vSetTrigger, 351
 MEXTI_vSWITrigger, 351
 MSYSCFG_vSetEXTIPort, 347
EXTI_RisingEdge
 Interrupt trigger status, 29
EXTI_Type, 111
 EMR, 112
 FTSR, 113
 IMR, 112
 PR, 113
 RTSR, 112
 SWIER, 113
EXTICR
 MSYSCFG_MemMap_t, 120
EXTSEL0
 ADC_private.h, 267
EXTSEL1
 ADC_private.h, 268
EXTSEL2
 ADC_private.h, 268
EXTSEL3
 ADC_private.h, 268
f32_t
 Standard types, 20
f64_t
 Standard types, 20
FALLING
 ADC_private.h, 303
FALSE
 Standard values, 21
FAULT_OCCURRED
 SPI_private.h, 528
FIFTEEN
 ADC_private.h, 300
FIRST_CLK_Captured
 SPI_private.h, 522
FIRST_ROW_IDX
 LCD_config.h, 183
FIRST_ROW_START
 LCD_config.h, 183
FirstByteInCGRAM_Pattern
 LCD_config.h, 184
FIVE
 ADC_private.h, 298
FLAG_CLEARED
 Standard values, 22
FLAG_SET
 Standard values, 21
FLASH
 NVIC Vector Table, 68
FORWARD
 DC Motor Directions, 9
FOUR
 ADC_private.h, 298
FOURTEEN
 ADC_private.h, 300
FPU
 NVIC Vector Table, 80
FRE
 SPI_private.h, 522
FRE_MODE
 SPI_config.h, 499
FRF
 SPI_private.h, 520
FRF_MODE
 SPI_config.h, 498

FTSR
 EXTI_Type, 113

FuctionSetCmd
 LCD_private.h, 198

FULL_DUPLEX
 SPI_private.h, 525

FUNC
 Compiler standard macros, 14

G_u8String
 UART_program.c, 615

GET_BIT
 Bit Manipulation Math Macros, 11

GET_BITs
 Bit Group Manipulation Math Macros, 13

GPIO Addresses, 48
 CLK_SOURCE, 49
 GPIOA_BASE_ADDRESS, 48
 GPIOB_BASE_ADDRESS, 48
 GPIOC_BASE_ADDRESS, 49

GPIO Alternate Functions, 43
 GPIOx_AF0, 44
 GPIOx_AF1, 44
 GPIOx_AF10, 46
 GPIOx_AF11, 47
 GPIOx_AF12, 47
 GPIOx_AF13, 47
 GPIOx_AF14, 47
 GPIOx_AF15, 48
 GPIOx_AF2, 44
 GPIOx_AF3, 45
 GPIOx_AF4, 45
 GPIOx_AF5, 45
 GPIOx_AF6, 45
 GPIOx_AF7, 46
 GPIOx_AF8, 46
 GPIOx_AF9, 46

GPIO Modes, 33
 GPIOx_MODE_AF, 33
 GPIOx_MODE_ANALOG, 33
 GPIOx_MODE_INPUT, 33
 GPIOx_MODE_OUTPUT, 33

GPIO Output PINs, 39
 GPIOx_PIN0, 39
 GPIOx_PIN1, 39
 GPIOx_PIN10, 42
 GPIOx_PIN11, 42
 GPIOx_PIN12, 42
 GPIOx_PIN13, 42
 GPIOx_PIN14, 43
 GPIOx_PIN15, 43
 GPIOx_PIN2, 40
 GPIOx_PIN3, 40
 GPIOx_PIN4, 40
 GPIOx_PIN5, 40
 GPIOx_PIN6, 41
 GPIOx_PIN7, 41
 GPIOx_PIN8, 41
 GPIOx_PIN9, 41

GPIO Output Types, 35
 GPIOx_OPENDRAIN, 35
 GPIOx_PUSH_PULL, 35

GPIO Output Values, 38
 GPIOx_HIGH, 38
 GPIOx_LOW, 38

GPIO PIN Speed, 36
 GPIOx_HighSpeed, 36
 GPIOx_LowSpeed, 36
 GPIOx_MediumSpeed, 36
 GPIOx_VeryHighSpeed, 36

GPIO Ports, 34
 GPIO_PORTA, 34
 GPIO_PORTB, 34
 GPIO_PORTC, 34

GPIO Pull Types, 37
 GPIOx_NoPull, 37
 GPIOx_PullDown, 37
 GPIOx_PullUp, 37

GPIO Registers, 49
 GPIOA, 49
 GPIOB, 50
 GPIOC, 50

GPIO_config.h
 GPIOA_PIN_POS, 362
 GPIOB_PIN_POS, 362
 LCKK_BIT_POS, 362
 PORTA_BIT_MANIPULATION, 363
 PORTB_BIT_MANIPULATION, 363

GPIO_interface.h
 GPIO_vSetValue, 372
 GPIOA_LOW_NIBBLE_HIGH, 367
 GPIOA_LOW_NIBBLE_LOW, 367
 MGPIOx_u8GetPinValue, 369
 MGPIOx_vInit, 371
 MGPIOx_vLockedPins, 367
 MGPIOx_vSetAlternateFunctionON, 370
 MGPIOx_vSetPinInputPullType, 369
 MGPIOx_vSetPinMode, 368
 MGPIOx_vSetPinOutputSpeed, 369
 MGPIOx_vSetPinOutputType, 368
 MGPIOx_vSetPinValue, 370
 MGPIOx_vSetPortConfigLock, 371
 MGPIOx_vSetResetAtomic, 370
 MGPIOx_vTogglePinValue, 371

GPIO_PORTA
 GPIO Ports, 34

GPIO_PORTB
 GPIO Ports, 34

GPIO_PORTC
 GPIO Ports, 34

GPIO_program.c
 GPIO_vSetValue, 383
 MGPIOx_u8GetPinValue, 379
 MGPIOx_vInit, 383
 MGPIOx_vLockedPins, 376
 MGPIOx_vSetAlternateFunctionON, 381
 MGPIOx_vSetPinInputPullType, 379

MGPIOx_vSetPinMode, 377
MGPIOx_vSetPinOutputSpeed, 378
MGPIOx_vSetPinOutputType, 377
MGPIOx_vSetPinValue, 380
MGPIOx_vSetResetAtomic, 380
MGPIOx_vTogglePinValue, 383
GPIO_vSetPortValue
 GPIO_interface.h, 372
 GPIO_program.c, 383
GPIOA
 GPIO Registers, 49
GPIOA_BASE_ADDRESS
 GPIO Addresses, 48
GPIOA_LOW_NIBBLE_HIGH
 GPIO_interface.h, 367
GPIOA_LOW_NIBBLE_LOW
 GPIO_interface.h, 367
GPIOA_PIN_POS
 GPIO_config.h, 362
GPIOACLK
 MRCC_config.h, 415
GPIOB
 GPIO Registers, 50
GPIOB_BASE_ADDRESS
 GPIO Addresses, 48
GPIOB_PIN_POS
 GPIO_config.h, 362
GPIOBCLK
 MRCC_config.h, 414
GPIOC
 GPIO Registers, 50
GPIOC_BASE_ADDRESS
 GPIO Addresses, 49
GPIOCCLK
 MRCC_config.h, 414
GPIODCLK
 MRCC_config.h, 414
GPIOECLK
 MRCC_config.h, 414
GPIOHCLK
 MRCC_config.h, 414
GPIOx_AF0
 GPIO Alternate Functions, 44
GPIOx_AF1
 GPIO Alternate Functions, 44
GPIOx_AF10
 GPIO Alternate Functions, 46
GPIOx_AF11
 GPIO Alternate Functions, 47
GPIOx_AF12
 GPIO Alternate Functions, 47
GPIOx_AF13
 GPIO Alternate Functions, 47
GPIOx_AF14
 GPIO Alternate Functions, 47
GPIOx_AF15
 GPIO Alternate Functions, 48
GPIOx_AF2

GPIO Alternate Functions, 44
GPIOx_AF3
 GPIO Alternate Functions, 45
GPIOx_AF4
 GPIO Alternate Functions, 45
GPIOx_AF5
 GPIO Alternate Functions, 45
GPIOx_AF6
 GPIO Alternate Functions, 45
GPIOx_AF7
 GPIO Alternate Functions, 46
GPIOx_AF8
 GPIO Alternate Functions, 46
GPIOx_AF9
 GPIO Alternate Functions, 46
GPIOx_HIGH
 GPIO Output Values, 38
GPIOx_HighSpeed
 GPIO PIN Speed, 36
GPIOx_LOW
 GPIO Output Values, 38
GPIOx_LowSpeed
 GPIO PIN Speed, 36
GPIOx_MediumSpeed
 GPIO PIN Speed, 36
GPIOx_MemoryMapType, 113
 AFRHx, 116
 AFRLx, 116
 BSRRx, 115
 IDRx, 115
 LCKRx, 115
 MODERx, 114
 ODRx, 115
 OSPEEDRx, 114
 OTYPERx, 114
 PUPDRx, 115
GPIOx_MODE_AF
 GPIO Modes, 33
GPIOx_MODE_ANALOG
 GPIO Modes, 33
GPIOx_MODE_INPUT
 GPIO Modes, 33
GPIOx_MODE_OUTPUT
 GPIO Modes, 33
GPIOx_NoPull
 GPIO Pull Types, 37
GPIOx_OPENDRAIN
 GPIO Output Types, 35
GPIOx_PIN0
 GPIO Output PINs, 39
GPIOx_PIN1
 GPIO Output PINs, 39
GPIOx_PIN10
 GPIO Output PINs, 42
GPIOx_PIN11
 GPIO Output PINs, 42
GPIOx_PIN12
 GPIO Output PINs, 42

GPIOx_PIN13
 GPIO Output PINs, 42

GPIOx_PIN14
 GPIO Output PINs, 43

GPIOx_PIN15
 GPIO Output PINs, 43

GPIOx_PIN2
 GPIO Output PINs, 40

GPIOx_PIN3
 GPIO Output PINs, 40

GPIOx_PIN4
 GPIO Output PINs, 40

GPIOx_PIN5
 GPIO Output PINs, 40

GPIOx_PIN6
 GPIO Output PINs, 41

GPIOx_PIN7
 GPIO Output PINs, 41

GPIOx_PIN8
 GPIO Output PINs, 41

GPIOx_PIN9
 GPIO Output PINs, 41

GPIOx_PullDown
 GPIO Pull Types, 37

GPIOx_PullUp
 GPIO Pull Types, 37

GPIOx_PUSH_PULL
 GPIO Output Types, 35

GPIOx_VeryHighSpeed
 GPIO PIN Speed, 36

Group priorities, 54

- GROUP_PRIORITY_0, 55
- GROUP_PRIORITY_1, 55
- GROUP_PRIORITY_10, 57
- GROUP_PRIORITY_11, 58
- GROUP_PRIORITY_12, 58
- GROUP_PRIORITY_13, 58
- GROUP_PRIORITY_14, 58
- GROUP_PRIORITY_15, 59
- GROUP_PRIORITY_2, 55
- GROUP_PRIORITY_3, 56
- GROUP_PRIORITY_4, 56
- GROUP_PRIORITY_5, 56
- GROUP_PRIORITY_6, 56
- GROUP_PRIORITY_7, 57
- GROUP_PRIORITY_8, 57
- GROUP_PRIORITY_9, 57
- NO_GROUP_PRIORITY, 55

GROUP_0BITS
 Interrupt priority grouping, 53

GROUP_1BITS
 Interrupt priority grouping, 53

GROUP_2BITS
 Interrupt priority grouping, 53

GROUP_3BITS
 Interrupt priority grouping, 53

GROUP_4BITS
 Interrupt priority grouping, 52

GROUP_PRIORITY_0
 Group priorities, 55

GROUP_PRIORITY_1
 Group priorities, 55

GROUP_PRIORITY_10
 Group priorities, 57

GROUP_PRIORITY_11
 Group priorities, 58

GROUP_PRIORITY_12
 Group priorities, 58

GROUP_PRIORITY_13
 Group priorities, 58

GROUP_PRIORITY_14
 Group priorities, 58

GROUP_PRIORITY_15
 Group priorities, 59

GROUP_PRIORITY_2
 Group priorities, 55

GROUP_PRIORITY_3
 Group priorities, 56

GROUP_PRIORITY_4
 Group priorities, 56

GROUP_PRIORITY_5
 Group priorities, 56

GROUP_PRIORITY_6
 Group priorities, 56

GROUP_PRIORITY_7
 Group priorities, 57

GROUP_PRIORITY_8
 Group priorities, 57

GROUP_PRIORITY_9
 Group priorities, 57

GTPR_REG
 USART_MemoryMapType, 149

HALF_DUPLEX_MODE
 SPI_config.h, 498

HALF_DUPLEX_RX
 SPI_private.h, 530

HALF_DUPLEX_TX
 SPI_private.h, 529

HardwareFlowControl
 USART_InitType, 147

HBluetooth_ExchangeByte
 Bluetooth_interface.h, 155

 Bluetooth_program.c, 159

HBluetooth_ExchangeString
 Bluetooth_interface.h, 155

 Bluetooth_program.c, 159

HBluetooth_u8CompStrings
 Bluetooth_interface.h, 154

 Bluetooth_program.c, 158

HBluetooth_u8ReceiveByte
 Bluetooth_interface.h, 154

 Bluetooth_program.c, 157

HBluetooth_vDisable
 Bluetooth_interface.h, 155

 Bluetooth_program.c, 159

HBluetooth_vEnable

Bluetooth_interface.h, 154
 Bluetooth_program.c, 159
HBluetooth_vInit
 Bluetooth_interface.h, 153
 Bluetooth_program.c, 157
HBluetooth_vReceiveString
 Bluetooth_interface.h, 154
 Bluetooth_program.c, 158
HBluetooth_vSendByte
 Bluetooth_interface.h, 153
 Bluetooth_program.c, 157
HBluetooth_vSendString
 Bluetooth_interface.h, 154
 Bluetooth_program.c, 158
HCarControl_vInitCar
 CarControl_interface.h, 163
 CarControl_program.c, 166
HCarControl_vMoveBackward
 CarControl_interface.h, 163
 CarControl_program.c, 166
HCarControl_vMoveForward
 CarControl_interface.h, 163
 CarControl_program.c, 166
HCarControl_vStopCar
 CarControl_interface.h, 164
 CarControl_program.c, 167
HCarControl_vTurnLeft
 CarControl_interface.h, 164
 CarControl_program.c, 167
HCarControl_vTurnRight
 CarControl_interface.h, 164
 CarControl_program.c, 166
HDCM_vInitMotor
 DCM_interface.h, 172
 DCM_program.c, 176
HDCM_vMotorSpeedCntrl
 DCM_interface.h, 173
HDCM_vMoveBackward
 DCM_interface.h, 173
 DCM_program.c, 176
HDCM_vMoveForward
 DCM_interface.h, 172
 DCM_program.c, 176
HDCM_vStopMotor
 DCM_interface.h, 173
 DCM_program.c, 177
HFSR
 SCB_MemoryMapType, 139
HLCD_CharactersNums
 LCD_config.h, 183
HLCD_CTRL_PORT
 LCD_config.h, 179
HLCD_DATA_PORT
 LCD_config.h, 180
HLCD_Dispclear_Cmd
 LCD_config.h, 182
HLCD_DispcursorWithBlinking_Cmd
 LCD_config.h, 182
HLCD_Disppoff_Cmd
 LCD_config.h, 182
HLCD_Disponoffctrl_Cmd
 LCD_config.h, 181
HLCD_EN_PIN
 LCD_config.h, 180
HLCD_EntryModeSet_Cmd
 LCD_config.h, 182
HLCD_EntryModeSet_ShiftLeftOn_Cmd
 LCD_config.h, 182
HLCD_FuctionSet_Cmd
 LCD_config.h, 181
HLCD_LINE1
 LCD_interface.h, 187
HLCD_LINE2
 LCD_interface.h, 187
HLCD_RS_PIN
 LCD_config.h, 179
HLCD_RW_PIN
 LCD_config.h, 179
HLCD_SetCursorBlinkingOFF_Cmd
 LCD_config.h, 182
HLCD_Square1
 LCD_interface.h, 187
HLCD_Square10
 LCD_interface.h, 189
HLCD_Square11
 LCD_interface.h, 189
HLCD_Square12
 LCD_interface.h, 189
HLCD_Square13
 LCD_interface.h, 189
HLCD_Square14
 LCD_interface.h, 189
HLCD_Square15
 LCD_interface.h, 189
HLCD_Square16
 LCD_interface.h, 190
HLCD_Square2
 LCD_interface.h, 187
HLCD_Square3
 LCD_interface.h, 187
HLCD_Square4
 LCD_interface.h, 188
HLCD_Square5
 LCD_interface.h, 188
HLCD_Square6
 LCD_interface.h, 188
HLCD_Square7
 LCD_interface.h, 188
HLCD_Square8
 LCD_interface.h, 188
HLCD_Square9
 LCD_interface.h, 188
HLCD_vClear
 LCD_interface.h, 194
 LCD_program.c, 204
HLCD_vClearChar

LCD_interface.h, 196
LCD_program.c, 208
HLCDD_vDispCursorWithBlinking
 LCD_interface.h, 195
 LCD_program.c, 207
HLCDD_vDispNumber
 LCD_interface.h, 194
 LCD_program.c, 205
HLCDD_vDispShiftLeftString
 LCD_interface.h, 196
 LCD_program.c, 208
HLCDD_vDispString
 LCD_interface.h, 194
 LCD_program.c, 204
HLCDD_vGoTo
 LCD_interface.h, 195
 LCD_program.c, 206
HLCDD_vInit
 LCD_interface.h, 191
 LCD_program.c, 201
HLCDD_vSaveCustomChar
 LCD_interface.h, 194
 LCD_program.c, 205
HLCDD_vSaveDispChar
 LCD_interface.h, 194
HLCDD_vSendCommand
 LCD_interface.h, 193
 LCD_program.c, 203
HLCDD_vSendData
 LCD_interface.h, 193
 LCD_program.c, 204
HLCDD_vSetCursorBlinkingOFF
 LCD_interface.h, 196
 LCD_program.c, 207
HLCDD_vSetDispOFF
 LCD_interface.h, 195
 LCD_program.c, 207
HLCDD_vSetShiftLeftOn
 LCD_interface.h, 195
 LCD_program.c, 206
HLDR_u16DigitalOutputValue
 LDR_interface.h, 219
 LDR_program.c, 221
HLDR_vInit
 LDR_interface.h, 218
 LDR_program.c, 220
HPRE
 MRCC_config.h, 413
HSE
 MRCC_private.h, 458
HSE_EN
 MRCC_config.h, 410
HSEby10
 MRCC_private.h, 462
HSEby11
 MRCC_private.h, 463
HSEby12
 MRCC_private.h, 463
 HSEby13
 MRCC_private.h, 463
 HSEby14
 MRCC_private.h, 463
 HSEby15
 MRCC_private.h, 463
 HSEby16
 MRCC_private.h, 463
 HSEby17
 MRCC_private.h, 464
 HSEby18
 MRCC_private.h, 464
 HSEby19
 MRCC_private.h, 464
 HSEby2
 MRCC_private.h, 461
 HSEby20
 MRCC_private.h, 464
 HSEby21
 MRCC_private.h, 464
 HSEby22
 MRCC_private.h, 464
 HSEby23
 MRCC_private.h, 465
 HSEby24
 MRCC_private.h, 465
 HSEby25
 MRCC_private.h, 465
 HSEby26
 MRCC_private.h, 465
 HSEby27
 MRCC_private.h, 465
 HSEby28
 MRCC_private.h, 465
 HSEby29
 MRCC_private.h, 466
 HSEby3
 MRCC_private.h, 461
 HSEby30
 MRCC_private.h, 466
 HSEby31
 MRCC_private.h, 466
 HSEby4
 MRCC_private.h, 461
 HSEby5
 MRCC_private.h, 462
 HSEby6
 MRCC_private.h, 462
 HSEby7
 MRCC_private.h, 462
 HSEby8
 MRCC_private.h, 462
 HSEby9
 MRCC_private.h, 462
 HSEBYP
 MRCC_config.h, 410
 HSI
 MRCC_private.h, 459

HSI_EN
 MRCC_config.h, 410
 HTR
 ADC_MemoryMapType, 106

 I2C1_ER
 NVIC Vector Table, 74
 I2C1_EV
 NVIC Vector Table, 74
 I2C1EN
 MRCC_config.h, 415
 I2C2_ER
 NVIC Vector Table, 74
 I2C2_EV
 NVIC Vector Table, 74
 I2C2EN
 MRCC_config.h, 415
 I2C3_ER
 NVIC Vector Table, 80
 I2C3_EV
 NVIC Vector Table, 80
 I2C3EN
 MRCC_config.h, 415
 I2S_CKIN
 MRCC_private.h, 459
 I2SCFGR
 SPI_MemoryMapType, 142
 I2SPR
 SPI_MemoryMapType, 142
 I2SSRC
 MRCC_config.h, 412
 IABRx
 NVIC_MemoryMapType, 124
 ICERx
 NVIC_MemoryMapType, 122
 ICPRx
 NVIC_MemoryMapType, 123
 ICSR
 SCB_MemoryMapType, 136
 ID options, 84, 86
 AHB1ENR_CRCEN, 88
 AHB1ENR_DMA1EN, 87
 AHB1ENR_DMA2EN, 87
 AHB1ENR_GPIOAEN, 89
 AHB1ENR_GPIOBEN, 89
 AHB1ENR_GPIOCEN, 89
 AHB1ENR_GPIODEN, 88
 AHB1ENR_GPIOEEN, 88
 AHB1ENR_GPIOHEN, 88
 AHB1LPENR_FLITFLPEN, 95
 AHB2ENR_OTGFSEN, 89
 APB1ENR_I2C1EN, 90
 APB1ENR_I2C2EN, 90
 APB1ENR_I2C3EN, 90
 APB1ENR_PWREN, 90
 APB1ENR_SPI2EN, 91
 APB1ENR_SPI3EN, 91
 APB1ENR_TIM2EN, 92
 APB1ENR_TIM3EN, 92
 APB1ENR_TIM4EN, 92
 APB1ENR_TIM5EN, 92
 APB1ENR_USART2EN, 91
 APB1ENR_WWDGEN, 91
 APB2ENR_ADC1EN, 94
 APB2ENR_SDIOEN, 94
 APB2ENR_SPI1EN, 94
 APB2ENR_SPI4EN, 94
 APB2ENR_SYSCFGEN, 93
 APB2ENR_TIM10EN, 93
 APB2ENR_TIM11EN, 93
 APB2ENR_TIM1EN, 95
 APB2ENR_TIM9EN, 93
 APB2ENR_USART1EN, 95
 APB2ENR_USART6EN, 95
 RCC_AHB1, 84
 RCC_AHB1LPENR, 85
 RCC_AHB2, 85
 RCC_APB1, 85
 RCC_APB2, 85
 IDRx
 GPIOx_MemoryMapType, 115
 IMR
 EXTI_Type, 112
 INITIAL_ZERO
 Standard values, 21
 INJ_ANALOG_WDT
 ADC_config.h, 230
 INJ_AUTO_MODE
 ADC_config.h, 230
 INJ_DISC_MODE
 ADC_config.h, 230
 INJ_EXT_EVENT_SRC
 ADC_config.h, 232
 INJ_EXTERNAL_TRIGGER
 ADC_config.h, 232
 INJ_START_CONV
 ADC_config.h, 232
 INJ_T2_TRGO
 ADC_private.h, 304
 InputType
 MGPIOx_ConfigType, 119
 Interrupt line IDs, 24
 EXTI_LINE0, 24
 EXTI_LINE1, 24
 EXTI_LINE10, 27
 EXTI_LINE11, 27
 EXTI_LINE12, 27
 EXTI_LINE13, 27
 EXTI_LINE14, 28
 EXTI_LINE15, 28
 EXTI_LINE2, 25
 EXTI_LINE3, 25
 EXTI_LINE4, 25
 EXTI_LINE5, 25
 EXTI_LINE6, 26
 EXTI_LINE7, 26
 EXTI_LINE8, 26

EXTI_LINE9, 26
Interrupt priority grouping, 50
 _16GROUP_NoSub_Priorities, 51
 _2GROUP_8Sub_Priorities, 52
 _4GROUP_4Sub_Priorities, 51
 _8GROUP_2Sub_Priorities, 51
GROUP_0BITS, 53
GROUP_1BITS, 53
GROUP_2BITS, 53
GROUP_3BITS, 53
GROUP_4BITS, 52
NoGROUP_16Sub_Priorities, 52
Interrupt trigger status, 28
 EXTI_FallingEdge, 28
 EXTI_OnChange, 29
 EXTI_RisingEdge, 29
IPRx
 NVIC_MemoryMapType, 125
ISERx
 NVIC_MemoryMapType, 121
ISPRx
 NVIC_MemoryMapType, 123
JAUTO
 ADC_private.h, 265
JAWDEN
 ADC_private.h, 264
JDISCEN
 ADC_private.h, 265
JDR1
 ADC_MemoryMapType, 107
JDR2
 ADC_MemoryMapType, 107
JDR3
 ADC_MemoryMapType, 107
JDR4
 ADC_MemoryMapType, 108
JEOC
 ADC_private.h, 263
JEOCIE
 ADC_private.h, 266
JEOCIE_MODE
 ADC_config.h, 231
JEXTEN0
 ADC_private.h, 268
JEXTEN1
 ADC_private.h, 268
JEXTSEL0
 ADC_private.h, 269
JEXTSEL1
 ADC_private.h, 269
JEXTSEL2
 ADC_private.h, 269
JEXTSEL3
 ADC_private.h, 269
JOFR1
 ADC_MemoryMapType, 105
JOFR2
 ADC_MemoryMapType, 105
JOFR3
 ADC_MemoryMapType, 105
JOFR4
 ADC_MemoryMapType, 106
JSQR
 ADC_MemoryMapType, 107
JSTRT
 ADC_private.h, 263
JSWSTART
 ADC_private.h, 268
L0
 ADC_private.h, 281
L1
 ADC_private.h, 281
L2
 ADC_private.h, 281
L3
 ADC_private.h, 281
LastBitClockPulse
 USART_ClockInitTypeDef, 145
LastByteInCGRAM_Pattern
 LCD_config.h, 184
LCD_config.h
 D0_PIN, 180
 D1_PIN, 180
 D2_PIN, 180
 D3_PIN, 180
 D4_PIN, 181
 D5_PIN, 181
 D6_PIN, 181
 D7_PIN, 181
 FIRST_ROW_IDX, 183
 FIRST_ROW_START, 183
FirstByteInCGRAM_Pattern, 184
HLCDD_CharactersNums, 183
HLCDD_CTRL_PORT, 179
HLCDD_DATA_PORT, 180
HLCDD_Dispclear_Cmd, 182
HLCDD_DispcursorWithBlinking_Cmd, 182
HLCDD_Dispoff_Cmd, 182
HLCDD_DisponoffCTRL_Cmd, 181
HLCDD_EN_PIN, 180
HLCDD_EntryModeSet_Cmd, 182
HLCDD_EntryModeSet_ShiftLeftOn_Cmd, 182
HLCDD_FuctionSet_Cmd, 181
HLCDD_RS_PIN, 179
HLCDD_RW_PIN, 179
HLCDD_SetCursorBlinkingOFF_Cmd, 182
LastByteInCGRAM_Pattern, 184
NumOf_CGRAM_Patterns, 184
SEC_ROW_START, 183
SET_CGRAM_AC_MASK, 183
SET_DDRAM_AC_MASK, 183
LCD_interface.h
 CGRAM_AddressOfPattern0, 190
 CGRAM_AddressOfPattern1, 190
 CGRAM_AddressOfPattern2, 190
 CGRAM_AddressOfPattern3, 190

CGRAM_AddressOfPattern4, 190
 CGRAM_AddressOfPattern5, 191
 CGRAM_AddressOfPattern7, 191
 CGRAM_AddressOfPattern8, 191
 LCD_LINE1, 187
 LCD_LINE2, 187
 LCD_Square1, 187
 LCD_Square10, 189
 LCD_Square11, 189
 LCD_Square12, 189
 LCD_Square13, 189
 LCD_Square14, 189
 LCD_Square15, 189
 LCD_Square16, 190
 LCD_Square2, 187
 LCD_Square3, 187
 LCD_Square4, 188
 LCD_Square5, 188
 LCD_Square6, 188
 LCD_Square7, 188
 LCD_Square8, 188
 LCD_Square9, 188
 LCD_vClear, 194
 LCD_vClearChar, 196
 LCD_vDispCursorWithBlinking, 195
 LCD_vDispNumber, 194
 LCD_vDispShiftLeftString, 196
 LCD_vDispString, 194
 LCD_vGoTo, 195
 LCD_vInit, 191
 LCD_vSaveCustomChar, 194
 LCD_vSaveDispChar, 194
 LCD_vSendCommand, 193
 LCD_vSendData, 193
 LCD_vSetCursorBlinkingOFF, 196
 LCD_vSetDispOFF, 195
 LCD_vSetShiftLeftOn, 195
 MAX_IDX_OF_COL, 187
 MAX_IDX_OF_ROWS, 186
 MIN_IDX_OF_COL, 186
 MIN_IDX_OF_ROWS, 186
 LCD_private.h
 DispClearCmd, 199
 DispCursorWithBlinkingCmd, 199
 DispOffCmd, 199
 DispOnOffCTRLCmd, 199
 EntryModeSet_Cmd, 199
 EntryModeSet_ShiftLeftOn_Cmd, 200
 FuctionSetCmd, 198
 SetCursorBlinkingOFFCmd, 199
 LCD_program.c
 LCD_vClear, 204
 LCD_vClearChar, 208
 LCD_vDispCursorWithBlinking, 207
 LCD_vDispNumber, 205
 LCD_vDispShiftLeftString, 208
 LCD_vDispString, 204
 LCD_vGoTo, 206
 LCD_vInit, 201
 LCD_vSaveCustomChar, 205
 LCD_vSendCommand, 203
 LCD_vSendData, 204
 LCD_vSetCursorBlinkingOFF, 207
 LCD_vSetDispOFF, 207
 LCD_vSetShiftLeftOn, 206
 LCKK_BIT_POS
 GPIO_config.h, 362
 LCKRx
 GPIOx_MemoryMapType, 115
 LDR_CHANNEL
 LDR_config.c, 215
 LDR_program.c, 221
 LDR_config.c
 LDR_CHANNEL, 215
 LDR_Vo, 215
 LDR_config.h
 LDR_Vo_PIN, 217
 LDR_Vo_PORT, 217
 LDR_interface.h
 DARK, 218
 HLDRL_u16DigitalOutputValue, 219
 HLDRL_vInit, 218
 MODERATE, 218
 OVERCAST, 218
 VERY_BRIGHT, 218
 VERY_DIM, 218
 LDR_program.c
 HLDRL_u16DigitalOutputValue, 221
 HLDRL_vInit, 220
 LDR_CHANNEL, 221
 LDR_Vo, 221
 LDR_Vo
 LDR_config.c, 215
 LDR_program.c, 221
 LDR_Vo_PIN
 LDR_config.h, 217
 LDR_Vo_PORT
 LDR_config.h, 217
 LEFT
 ADC_private.h, 306
 SPI_private.h, 527
 LineNum
 EXTI_ConfigType, 111
 LOAD
 SysTick_Type, 143
 LSB_First
 SPI_private.h, 525
 LSBFIRST
 SPI_private.h, 518
 LSBFIRST_MODE
 SPI_config.h, 496
 LSE
 MRCC_private.h, 459
 LTR
 ADC_MemoryMapType, 106
 MADC_ConfigType, 116

Port, 117
MADC_u16ConvertToDigital
 ADC_interface.h, 253
 ADC_program.c, 320
MADC_u16GetADCData
 ADC_interface.h, 255
MADC_vDisable
 ADC_interface.h, 253
 ADC_program.c, 321
MADC_vEnable
 ADC_interface.h, 253
 ADC_program.c, 321
MADC_vInit
 ADC_interface.h, 249
 ADC_program.c, 316
MADC_vInitStruct
 ADC_interface.h, 255
MADC_vpPointerToFunction
 ADC_program.c, 323
MADC_vRegINT_Disable
 ADC_interface.h, 253
 ADC_program.c, 321
MADC_vRegINTEnable
 ADC_interface.h, 253
 ADC_program.c, 322
MADC_vSelectChannel
 ADC_interface.h, 254
 ADC_program.c, 320
MADC_vSelectChannelsOrder
 ADC_interface.h, 254
MADC_vSelectMode
 ADC_interface.h, 254
MADC_vSelectResolution
 ADC_interface.h, 254
MADC_vSelectSampleTime
 ADC_interface.h, 254
MADC_vSetCallBack
 ADC_interface.h, 255
 ADC_program.c, 322
MADC_vStartConversion
 ADC_interface.h, 254
MASTER
 SPI_private.h, 523
MATCHED
 SPI_private.h, 528
MAX_IDX_OF_COL
 LCD_interface.h, 187
MAX_IDX_OF_ROWS
 LCD_interface.h, 186
MAX_INFO_LENGTH
 UsersLogin_private.h, 623
MAX_TICKS
 SysTick_config.h, 550
MCO1
 MRCC_config.h, 412
MCO1PRE
 MRCC_config.h, 412
MCO2
 MRCC_config.h, 411
MCO2PRE
 MRCC_config.h, 412
MEMORY_MANAGE
 NVIC_Settable Priorities, 83
MEMRMP
 MSYSCFG_MemMap_t, 120
MEXTI
 EXTI Registers, 30
MEXTI_vDisableLine
 EXTI_interface.h, 342
 EXTI_program.c, 351
MEXTI_vEnableLine
 EXTI_interface.h, 342
 EXTI_program.c, 350
MEXTI_vInit
 EXTI_interface.h, 339
 EXTI_program.c, 347
MEXTI_vInit_WithStruct
 EXTI_interface.h, 342
 EXTI_program.c, 350
MEXTI_vSetCallback
 EXTI_interface.h, 343
 EXTI_program.c, 352
MEXTI_vSetTrigger
 EXTI_interface.h, 343
 EXTI_program.c, 351
MEXTI_vSWITrigger
 EXTI_interface.h, 343
 EXTI_program.c, 351
MGPIOx_ConfigType, 117
 AF_Type, 119
 InputType, 119
 Mode, 118
 OutputSpeed, 118
 OutputType, 118
 Pin, 118
 Port, 117
MGPIOx_u8GetPinValue
 GPIO_interface.h, 369
 GPIO_program.c, 379
MGPIOx_vInit
 GPIO_interface.h, 371
 GPIO_program.c, 383
MGPIOx_vLockedPins
 GPIO_interface.h, 367
 GPIO_program.c, 376
MGPIOx_vSetAlternateFunctionON
 GPIO_interface.h, 370
 GPIO_program.c, 381
MGPIOx_vSetPinInputPullType
 GPIO_interface.h, 369
 GPIO_program.c, 379
MGPIOx_vSetPinMode
 GPIO_interface.h, 368
 GPIO_program.c, 377
MGPIOx_vSetPinOutputSpeed
 GPIO_interface.h, 369

GPIO_program.c, 378
MGPIOx_vSetPinOutputType
 GPIO_interface.h, 368
 GPIO_program.c, 377
MGPIOx_vSetPinValue
 GPIO_interface.h, 370
 GPIO_program.c, 380
MGPIOx_vSetPortConfigLock
 GPIO_interface.h, 371
MGPIOx_vSetResetAtomic
 GPIO_interface.h, 370
 GPIO_program.c, 380
MGPIOx_vTogglePinValue
 GPIO_interface.h, 371
 GPIO_program.c, 383
MICRO_SEC
 Delay Units, 97
MILLI_SEC
 Delay Units, 97
MIN_IDX_OF_COL
 LCD_interface.h, 186
MIN_IDX_OF_ROWS
 LCD_interface.h, 186
MMFAR
 SCB_MemoryMapType, 139
MNVIC
 NVIC Registers, 82
MNVIC_STIR
 NVIC_private.h, 400
MNVIC_u8GetActive
 NVIC_interface.h, 396
 NVIC_program.c, 404
MNVIC_vClearPendingFlag
 NVIC_interface.h, 396
 NVIC_program.c, 403
MNVIC_vDisablePeriphral
 NVIC_interface.h, 395
 NVIC_program.c, 403
MNVIC_vEnablePeriphral
 NVIC_interface.h, 395
 NVIC_program.c, 403
MNVIC_vSetPendingFlag
 NVIC_interface.h, 395
 NVIC_program.c, 403
MNVIC_vSetPriority
 NVIC_interface.h, 397
 NVIC_program.c, 404
MNVIC_vSetPriorityConfig
 NVIC_interface.h, 397
 NVIC_program.c, 404
Mode
 MGPIOx_ConfigType, 118
MODE_8BIT
 UART_interface.h, 581
MODE_9BIT
 UART_interface.h, 581
MODERATE
 LDR_interface.h, 218
MODERx
 GPIOx_MemoryMapType, 114
MODF
 SPI_private.h, 522
MODF_MODE
 SPI_config.h, 500
MOTOROLA
 SPI_private.h, 527
MRCC_config.h
 ADC1EN, 418
 CRCCLK, 414
 CSS, 410
 DMA1CLK, 413
 DMA2CLK, 413
 GPIOACLK, 415
 GPIOBCLK, 414
 GPIOCCLK, 414
 GPIODCLK, 414
 GPIOECLK, 414
 GPIOHCLK, 414
 HPRE, 413
 HSE_EN, 410
 HSEBYP, 410
 HSI_EN, 410
 I2C1EN, 415
 I2C2EN, 415
 I2C3EN, 415
 I2SSRC, 412
 MCO1, 412
 MCO1PRE, 412
 MCO2, 411
 MCO2PRE, 412
 OTGFS, 415
 PLL, 410
 PLLI2S, 410
 PLLM, 411
 PLLN, 411
 PLLP, 411
 PLLQ, 411
 PLLSRC, 411
 PPRE1, 413
 PPRE2, 412
 PWREN, 415
 RTCPRE, 412
 SDIOEN, 418
 SPI1EN, 418
 SPI2EN, 416
 SPI3EN, 416
 SPI4EN, 418
 SW, 413
 SWS, 413
 SYSCFGEN, 417
 TIM10EN, 417
 TIM11EN, 417
 TIM1EN, 419
 TIM2EN, 417
 TIM3EN, 417
 TIM4EN, 416

TIM5EN, 416
TIM9EN, 417
USART1EN, 418
USART2EN, 416
USART6EN, 418
WWDGEN, 416
MRCC_interface.h
 MRCC_vDisablePeriphralCLK, 434
 MRCC_vEnablePeriphralCLK, 433
 MRCC_vInit, 429
MRCC_private.h
 AHBby16, 467
 AHBby2, 466
 AHBby4, 466
 AHBby8, 466
 BYBASED, 457
 DISABLE, 457
 DivisionBy2, 458
 DivisionBy3, 458
 DivisionBy4, 458
 DivisionBy5, 459
 ENABLE, 457
 Equal_0, 467
 Equal_1, 467
 Equal_10, 468
 Equal_11, 469
 Equal_12, 469
 Equal_13, 469
 Equal_14, 469
 Equal_15, 469
 Equal_16, 469
 Equal_17, 470
 Equal_18, 470
 Equal_19, 470
 Equal_2, 467
 Equal_20, 470
 Equal_21, 470
 Equal_22, 470
 Equal_23, 471
 Equal_24, 471
 Equal_25, 471
 Equal_26, 471
 Equal_27, 471
 Equal_28, 471
 Equal_29, 472
 Equal_3, 467
 Equal_30, 472
 Equal_31, 472
 Equal_32, 472
 Equal_33, 472
 Equal_34, 472
 Equal_35, 473
 Equal_36, 473
 Equal_37, 473
 Equal_38, 473
 Equal_39, 473
 Equal_4, 467
 Equal_40, 473
 Equal_41, 474
 Equal_42, 474
 Equal_43, 474
 Equal_44, 474
 Equal_45, 474
 Equal_46, 474
 Equal_47, 475
 Equal_48, 475
 Equal_49, 475
 Equal_5, 468
 Equal_50, 475
 Equal_51, 475
 Equal_52, 475
 Equal_53, 476
 Equal_54, 476
 Equal_55, 476
 Equal_56, 476
 Equal_57, 476
 Equal_58, 476
 Equal_59, 477
 Equal_6, 468
 Equal_60, 477
 Equal_61, 477
 Equal_62, 477
 Equal_63, 477
 Equal_7, 468
 Equal_8, 468
 Equal_9, 468
 HSE, 458
 HSEby10, 462
 HSEby11, 463
 HSEby12, 463
 HSEby13, 463
 HSEby14, 463
 HSEby15, 463
 HSEby16, 463
 HSEby17, 464
 HSEby18, 464
 HSEby19, 464
 HSEby2, 461
 HSEby20, 464
 HSEby21, 464
 HSEby22, 464
 HSEby23, 465
 HSEby24, 465
 HSEby25, 465
 HSEby26, 465
 HSEby27, 465
 HSEby28, 465
 HSEby29, 466
 HSEby3, 461
 HSEby30, 466
 HSEby31, 466
 HSEby4, 461
 HSEby5, 462
 HSEby6, 462
 HSEby7, 462
 HSEby8, 462

HSEby9, 462
 HSI, 459
 I2S_CKIN, 459
 LSE, 459
 NoCLK0, 461
 NoCLK1, 461
 NoDivision, 458
 NOTBYBASED, 457
 PLLCLK, 458, 459
 PLLI2SCLK, 457
 RCC, 440
 RCC_AHB1ENR_CRCEN, 451
 RCC_AHB1ENR_DMA1EN, 451
 RCC_AHB1ENR_DMA2EN, 451
 RCC_AHB1ENR_GPIOAEN, 452
 RCC_AHB1ENR_GPIOBEN, 452
 RCC_AHB1ENR_GPIOCEN, 452
 RCC_AHB1ENR_GPIODEN, 452
 RCC_AHB1ENR_GPIOEEN, 452
 RCC_AHB1ENR_GPIOHEN, 451
 RCC_AHB1LPENR_FLITFLPEN, 456
 RCC_AHB2ENR_OTGFSEN, 452
 RCC_APB1ENR_I2C1EN, 453
 RCC_APB1ENR_I2C2EN, 453
 RCC_APB1ENR_I2C3EN, 453
 RCC_APB1ENR_PWREN, 453
 RCC_APB1ENR_SPI2EN, 454
 RCC_APB1ENR_SPI3EN, 453
 RCC_APB1ENR_TIM2EN, 454
 RCC_APB1ENR_TIM3EN, 454
 RCC_APB1ENR_TIM4EN, 454
 RCC_APB1ENR_TIM5EN, 454
 RCC_APB1ENR_USART2EN, 453
 RCC_APB1ENR_WWDGEN, 454
 RCC_APB2ENR_ADC1EN, 456
 RCC_APB2ENR_SDIOEN, 456
 RCC_APB2ENR_SPI1EN, 455
 RCC_APB2ENR_SPI4EN, 455
 RCC_APB2ENR_SYSCFGEN, 455
 RCC_APB2ENR_TIM10EN, 455
 RCC_APB2ENR_TIM11EN, 455
 RCC_APB2ENR_TIM1EN, 456
 RCC_APB2ENR_TIM9EN, 455
 RCC_APB2ENR_USART1EN, 456
 RCC_APB2ENR_USART6EN, 456
 RCC_BASE_ADDRESS, 440
 RCC_CFGR_HPRE_b0, 450
 RCC_CFGR_HPRE_b1, 450
 RCC_CFGR_HPRE_b2, 450
 RCC_CFGR_HPRE_b3, 450
 RCC_CFGR_I2SSRC, 447
 RCC_CFGR_MOC1_b0, 447
 RCC_CFGR_MOC1_b1, 448
 RCC_CFGR_MOC1PRE_b0, 447
 RCC_CFGR_MOC1PRE_b1, 447
 RCC_CFGR_MOC1PRE_b2, 447
 RCC_CFGR_MOC2_b0, 446
 RCC_CFGR_MOC2_b1, 446
 RCC_CFGR_MOC2PRE_b0, 446
 RCC_CFGR_MOC2PRE_b1, 446
 RCC_CFGR_MOC2PRE_b2, 447
 RCC_CFGR_PPRE1_b0, 449
 RCC_CFGR_PPRE1_b1, 449
 RCC_CFGR_PPRE1_b2, 449
 RCC_CFGR_PPRE2_b0, 449
 RCC_CFGR_PPRE2_b1, 449
 RCC_CFGR_PPRE2_b2, 449
 RCC_CFGR_RTCPRE_b0, 448
 RCC_CFGR_RTCPRE_b1, 448
 RCC_CFGR_RTCPRE_b2, 448
 RCC_CFGR_RTCPRE_b3, 448
 RCC_CFGR_RTCPRE_b4, 448
 RCC_CFGR_SW_b0, 451
 RCC_CFGR_SW_b1, 451
 RCC_CFGR_SWS_b0, 450
 RCC_CFGR_SWS_b1, 450
 RCC_CR_CSSON, 441
 RCC_CR_HSEBYP, 441
 RCC_CR_HSEON, 442
 RCC_CR_HSERDY, 442
 RCC_CR_HSION, 442
 RCC_CR_HSIRDY, 442
 RCC_CR_PLLI2SON, 441
 RCC_CR_PLLI2SRDY, 441
 RCC_CR_PLLON, 441
 RCC_CR_PLLRDY, 441
 RCC_PLLCFGR_PLLM_b0, 445
 RCC_PLLCFGR_PLLM_b1, 445
 RCC_PLLCFGR_PLLM_b2, 445
 RCC_PLLCFGR_PLLM_b3, 445
 RCC_PLLCFGR_PLLM_b4, 446
 RCC_PLLCFGR_PLLM_b5, 446
 RCC_PLLCFGR_PLLN_b0, 443
 RCC_PLLCFGR_PLLN_b1, 444
 RCC_PLLCFGR_PLLN_b2, 444
 RCC_PLLCFGR_PLLN_b3, 444
 RCC_PLLCFGR_PLLN_b4, 444
 RCC_PLLCFGR_PLLN_b5, 444
 RCC_PLLCFGR_PLLN_b6, 444
 RCC_PLLCFGR_PLLN_b7, 445
 RCC_PLLCFGR_PLLN_b8, 445
 RCC_PLLCFGR_PLLP_b0, 443
 RCC_PLLCFGR_PLLP_b1, 443
 RCC_PLLCFGR_PLLQ_b0, 442
 RCC_PLLCFGR_PLLQ_b1, 442
 RCC_PLLCFGR_PLLQ_b2, 443
 RCC_PLLCFGR_PLLQ_b3, 443
 RCC_PLLCFGR_PLLSRC, 443
 SYSCLK, 457
 SYSCLKby128, 460
 SYSCLKby16, 460
 SYSCLKby2, 459
 SYSCLKby256, 460
 SYSCLKby4, 460
 SYSCLKby512, 461
 SYSCLKby64, 460

SYSCLKby8, 460
MRCC_program.c
 MRCC_vDisablePeriphralCLK, 488
 MRCC_vEnablePeriphralCLK, 487
 MRCC_vInit, 483
MRCC_vDisablePeriphralCLK
 MRCC_interface.h, 434
 MRCC_program.c, 488
MRCC_vEnablePeriphralCLK
 MRCC_interface.h, 433
 MRCC_program.c, 487
MRCC_vInit
 MRCC_interface.h, 429
 MRCC_program.c, 483
MSB_First
 SPI_private.h, 525
MSCB
 NVIC Registers, 82
MSPI1_CallBack
 SPI_program.c, 542
MSPI1_vSetCallBack
 SPI_interface.h, 514
 SPI_program.c, 540
MSPI2_CallBack
 SPI_program.c, 542
MSPI2_vSetCallBack
 SPI_interface.h, 514
 SPI_program.c, 540
MSPI3_CallBack
 SPI_program.c, 542
MSPI3_vSetCallBack
 SPI_interface.h, 515
 SPI_program.c, 541
MSPI_u16MasterRecieve
 SPI_interface.h, 511
 SPI_program.c, 537
MSPI_u16SlaveRecieve
 SPI_interface.h, 512
 SPI_program.c, 538
MSPI_u16Transcieve
 SPI_interface.h, 510
 SPI_program.c, 537
MSPI_vDISABLE
 SPI_interface.h, 512
 SPI_program.c, 538
MSPI_vInit
 SPI_interface.h, 507
 SPI_program.c, 533
MSPI_vMasterTransmit
 SPI_interface.h, 511
 SPI_program.c, 537
MSPI_vSlaveTransmit
 SPI_interface.h, 512
 SPI_program.c, 538
MSTR
 SPI_private.h, 518
MSTR_MODE
 SPI_config.h, 496
MSYSCFG
 SYSFCFG Memory Registers, 32
MSYSCFG_MemMap_t, 119
 CMPCR, 120
 EXTICR, 120
 MEMRMP, 120
 PMC, 120
MSYSCFG_vSetEXTIPort
 EXTI_interface.h, 343
 EXTI_program.c, 347
MSysTick_u32GetElapsedTime
 SysTick_interface.h, 557
 SysTick_program.c, 567
MSysTick_u32GetRemainingTime
 SysTick_interface.h, 557
 SysTick_program.c, 567
MSysTick_vDelay
 SysTick_interface.h, 554
 SysTick_program.c, 563
MSysTick_vDelayMicroSec
 SysTick_interface.h, 555
 SysTick_program.c, 564
MSysTick_vDelayMilliSec
 SysTick_interface.h, 555
 SysTick_program.c, 564
MSysTick_vDelaySec
 SysTick_interface.h, 555
 SysTick_program.c, 565
MSysTick_vDisable
 SysTick_interface.h, 558
 SysTick_program.c, 568
MSysTick_vEnable
 SysTick_interface.h, 557
 SysTick_program.c, 568
MSysTick_vEnableException
 SysTick_interface.h, 558
 SysTick_program.c, 562
MSysTick_vInit
 SysTick_interface.h, 553
 SysTick_program.c, 562
MSysTick_vSetBusyWait
 SysTick_interface.h, 554
 SysTick_program.c, 562
MSysTick_vSetPeriodicInterval
 SysTick_interface.h, 556
 SysTick_program.c, 566
MSysTick_vSetSingleInterval
 SysTick_interface.h, 556
 SysTick_program.c, 565
MSysTick_vStopInterval
 SysTick_interface.h, 556
 SysTick_program.c, 566
MULTI_MSTR_MODE
 SPI_config.h, 498
MUSART1_CallBack
 UART_program.c, 616
MUSART1_vSetCallBack
 UART_interface.h, 590

UART_program.c, 614
MUSART2_CallBack
 UART_program.c, 616
MUSART2_vSetCallBack
 UART_interface.h, 590
 UART_program.c, 614
MUSART6_CallBack
 UART_program.c, 616
MUSART6_vSetCallBack
 UART_interface.h, 590
 UART_program.c, 614
MUSART_CR1_IDLEIE_BIT
 UART_private.h, 598
MUSART_CR1_M_BIT
 UART_private.h, 599
MUSART_CR1_OVER8_BIT
 UART_private.h, 600
MUSART_CR1_PCE_BIT
 UART_private.h, 599
MUSART_CR1_PEIE_BIT
 UART_private.h, 599
MUSART_CR1_PS_BIT
 UART_private.h, 599
MUSART_CR1_RE_BIT
 UART_private.h, 598
MUSART_CR1_RWU_BIT
 UART_private.h, 598
MUSART_CR1_RXNEIE_BIT
 UART_private.h, 598
MUSART_CR1_SBK_BIT
 UART_private.h, 597
MUSART_CR1_TCIE_BIT
 UART_private.h, 598
MUSART_CR1_TE_BIT
 UART_private.h, 598
MUSART_CR1_TXEIE_BIT
 UART_private.h, 599
MUSART_CR1_UE_BIT
 UART_private.h, 600
MUSART_CR1_WAKE_BIT
 UART_private.h, 599
MUSART_CR2_ADD0_BIT
 UART_private.h, 600
MUSART_CR2_ADD1_BIT
 UART_private.h, 600
MUSART_CR2_ADD2_BIT
 UART_private.h, 600
MUSART_CR2_ADD3_BIT
 UART_private.h, 600
MUSART_CR2_CLKEN_BIT
 UART_private.h, 601
MUSART_CR2_CPHA_BIT
 UART_private.h, 601
MUSART_CR2_CPOL_BIT
 UART_private.h, 601
MUSART_CR2_LBCL_BIT
 UART_private.h, 601
MUSART_CR2_LBDIE_BIT
 UART_private.h, 601
MUSART_CR2_LBDL_BIT
 UART_private.h, 601
MUSART_CR2_LINEN_BIT
 UART_private.h, 602
MUSART_CR2_STOP0_BIT
 UART_private.h, 602
MUSART_CR2_STOP1_BIT
 UART_private.h, 602
MUSART_CR2_STOP_BIT
 UART_private.h, 602
MUSART_CR3_CTSE_BIT
 UART_private.h, 602
MUSART_CR3_CTSIE_BIT
 UART_private.h, 602
MUSART_CR3_DMAR_BIT
 UART_private.h, 603
MUSART_CR3_DMAT_BIT
 UART_private.h, 603
MUSART_CR3_EIE_BIT
 UART_private.h, 604
MUSART_CR3_HDSEL_BIT
 UART_private.h, 603
MUSART_CR3_IREN_BIT
 UART_private.h, 604
MUSART_CR3_IRLP_BIT
 UART_private.h, 604
MUSART_CR3_NACK_BIT
 UART_private.h, 603
MUSART_CR3_RTSE_BIT
 UART_private.h, 603
MUSART_CR3_SCEN_BIT
 UART_private.h, 603
MUSART_ptrReceiveStringSynchNonBlocking
 UART_interface.h, 587
 UART_program.c, 611
MUSART_SR_CTS_BIT
 UART_private.h, 597
MUSART_SR_FE_BIT
 UART_private.h, 596
MUSART_SR_IDLE_BIT
 UART_private.h, 596
MUSART_SR_LBD_BIT
 UART_private.h, 597
MUSART_SR_NE_BIT
 UART_private.h, 596
MUSART_SR_ORE_BIT
 UART_private.h, 596
MUSART_SR_PE_BIT
 UART_private.h, 596
MUSART_SR_RXNE_BIT
 UART_private.h, 597
MUSART_SR_TC_BIT
 UART_private.h, 597
MUSART_SR_TXE_BIT
 UART_private.h, 597
MUSART_u8CompareString
 UART_interface.h, 588

UART_program.c, 612
MUSART_u8ReadDataRegister
 UART_interface.h, 589
 UART_program.c, 613
MUSART_u8ReceiveByteSynchBlocking
 UART_interface.h, 587
 UART_program.c, 611
MUSART_u8ReceiveByteSynchNonBlocking
 UART_interface.h, 586
 UART_program.c, 610
MUSART_vClearFlags
 UART_interface.h, 589
 UART_program.c, 613
MUSART_vDisable
 UART_interface.h, 585
 UART_program.c, 609
MUSART_vEnable
 UART_interface.h, 585
 UART_program.c, 609
MUSART_vInit
 UART_interface.h, 583
 UART_program.c, 607
MUSART_vRecieveString
 UART_interface.h, 588
 UART_program.c, 612
MUSART_vRxIntSetStatus
 UART_interface.h, 589
 UART_program.c, 613
MUSART_vTransmitByte
 UART_interface.h, 585
 UART_program.c, 609
MUSART_vTransmitString
 UART_interface.h, 586
 UART_program.c, 610
MY_MS_DELAY
 Utilities macros, 16

NINE
 ADC_private.h, 299
NO_CRC_PHASE
 SPI_private.h, 526
NO_ERROR
 SPI_private.h, 529
NO_FAULT_OCCURRED
 SPI_private.h, 529
NO_GROUP_PRIORITY
 Group priorities, 55
NO_SUB_PRIORITY
 Sub-Group priorities, 60
NoCLK0
 MRCC_private.h, 461
NoCLK1
 MRCC_private.h, 461
NoDivision
 MRCC_private.h, 458
NoGROUP_16Sub_Priorities
 Interrupt priority grouping, 52
NOT_BUSY
 SPI_private.h, 529

NOT_MATCHED
 SPI_private.h, 528
NOT_OCCURRED
 SPI_private.h, 528
NOTBYBASED
 MRCC_private.h, 457
NOTEEMPTY
 SPI_private.h, 527
NULL
 Standard values, 21
NumOf_CGRAM_Patterns
 LCD_config.h, 184
NVIC Addresses, 81
 NVIC_BASE_ADDRESS, 81
 SCB_BASE_ADDRESS, 81
NVIC Registers, 82
 MNVIC, 82
 MSCB, 82
NVIC Settable Priorities, 82
 BUSFAULT, 84
 MEMORYMANAGE, 83
 PENDSV, 83
 SVCALL, 83
 SYSTICK, 83
 USAGEFAULT, 84
NVIC Vector Table, 64
 ADC, 71
 DMA1_STREAM0, 70
 DMA1_STREAM1, 70
 DMA1_STREAM2, 70
 DMA1_STREAM3, 70
 DMA1_STREAM4, 71
 DMA1_STREAM5, 71
 DMA1_STREAM6, 71
 DMA1_STREAM7, 76
 DMA2_STREAM0, 77
 DMA2_STREAM1, 78
 DMA2_STREAM2, 78
 DMA2_STREAM3, 78
 DMA2_STREAM4, 79
 DMA2_STREAM5, 78
 DMA2_STREAM6, 79
 DMA2_STREAM7, 79
 EXTI0, 68
 EXTI1, 69
 EXTI15_10, 76
 EXTI16, 67
 EXTI17, 76
 EXTI18, 76
 EXTI2, 69
 EXTI21, 67
 EXTI22, 68
 EXTI3, 69
 EXTI4, 69
 EXTI9, 72
 FLASH, 68
 FPU, 80
 I2C1_ER, 74

I2C1_EV, 74
 I2C2_ER, 74
 I2C2_EV, 74
 I2C3_ER, 80
 I2C3_EV, 80
 OTG_FS, 79
 RCC, 68
 SDIO, 77
 SPI1, 75
 SPI2, 75
 SPI3, 77
 SPI4, 80
 TIM1_BRK_TIM9, 72
 TIM1_CC, 73
 TIM1_TRG_COM_TIM11, 72
 TIM1_UP_TIM10, 72
 TIM2, 73
 TIM3, 73
 TIM4, 73
 TIM5, 77
 USART1, 75
 USART2, 75
 USART6, 80
 WWDG, 67
 NVIC_BASE_ADDRESS
 NVIC Addresses, 81
 NVIC_GetPriority
 NVIC_interface.h, 397
 NVIC_program.c, 405
 NVIC_interface.h
 MNVIC_u8GetActive, 396
 MNVIC_vClearPendingFlag, 396
 MNVIC_vDisablePeriphral, 395
 MNVIC_vEnablePeriphral, 395
 MNVIC_vSetPendingFlag, 395
 MNVIC_vSetPriority, 397
 MNVIC_vSetPriorityConfig, 397
 NVIC_GetPriority, 397
 NVIC_MemoryMapType, 121
 IABRx, 124
 ICERx, 122
 ICPRx, 123
 IPRx, 125
 ISERx, 121
 ISPRx, 123
 RESERVED0x, 122
 RESERVED2x, 123
 RESERVED3x, 124
 RESERVED4x, 124
 RSERVED1x, 122
 NVIC_private.h
 MNVIC_STIR, 400
 VECTKEY_PASSWORD, 401
 NVIC_program.c
 MNVIC_u8GetActive, 404
 MNVIC_vClearPendingFlag, 403
 MNVIC_vDisablePeriphral, 403
 MNVIC_vEnablePeriphral, 403
 MNVIC_vSetPendingFlag, 403
 MNVIC_vSetPriority, 404
 MNVIC_vSetPriorityConfig, 404
 PEND_SV

MNVIC_vSetPendingFlag, 403
 MNVIC_vSetPriority, 404
 MNVIC_vSetPriorityConfig, 404
 NVIC_GetPriority, 405
 REGISTER_SIZE, 403
 OCCURRED
 SPI_private.h, 528
 ODD_PARITY
 UART_interface.h, 581
 ODRx
 GPIOx_MemoryMapType, 115
 ON_CHANGE
 ADC_private.h, 303
 ONE
 ADC_private.h, 298
 OneLine
 SPI_private.h, 526
 OSPEEDRx
 GPIOx_MemoryMapType, 114
 OTG_FS
 NVIC Vector Table, 79
 OTGFS
 MRCC_config.h, 415
 OTYPERx
 GPIOx_MemoryMapType, 114
 OutputSpeed
 MGPIOx_ConfigType, 118
 OutputType
 MGPIOx_ConfigType, 118
 OVER_SAMPLING_16
 UART_interface.h, 580
 OVER_SAMPLING_8
 UART_interface.h, 580
 OVERCAST
 LDR_interface.h, 218
 OVERRUN_INT
 ADC_config.h, 229
 Oversampling
 USART_InitType, 147
 OVR
 ADC_private.h, 262
 SPI_private.h, 522
 OVR_MODE
 SPI_config.h, 499
 OVRIE
 ADC_private.h, 263
 P2CONST
 Compiler standard macros, 15
 P2FUNC
 Compiler standard macros, 15
 P2VAR
 Compiler standard macros, 14
 Parity_Enable
 USART_InitType, 146
 Parity_Selection
 USART_InitType, 146

NVIC Settable Priorities, 83
PERIODIC_INTERVAL_MODE
 Systick Interval Mode Configuration, 96
PERIODIC_INTERVAL_TICK_TIME
 SysTick_interface.h, 553
Pin
 MGPIOx_ConfigType, 118
PLL
 MRCC_config.h, 410
PLLCFGR
 RCC_MemoryMapType, 127
PLLCLK
 MRCC_private.h, 458, 459
PLLI2S
 MRCC_config.h, 410
PLLI2SCFGR
 RCC_MemoryMapType, 134
PLL2SCLK
 MRCC_private.h, 457
PLLM
 MRCC_config.h, 411
PLLN
 MRCC_config.h, 411
PLLP
 MRCC_config.h, 411
PLLQ
 MRCC_config.h, 411
PLLSRC
 MRCC_config.h, 411
PMC
 MSYSCFG_MemMap_t, 120
Port
 MADC_ConfigType, 117
 MGPIOx_ConfigType, 117
PORTA_BIT_MANIPULATION
 GPIO_config.h, 363
PORTB_BIT_MANIPULATION
 GPIO_config.h, 363
PortNum
 EXTI_ConfigType, 111
PPRE1
 MRCC_config.h, 413
PPRE2
 MRCC_config.h, 412
PR
 EXTI_Type, 113
PRESSED
 Standard values, 22
PUPDRx
 GPIOx_MemoryMapType, 115
PWREN
 MRCC_config.h, 415
RCC
 MRCC_private.h, 440
 NVIC Vector Table, 68
RCC_AHB1
 ID options, 84
RCC_AHB1ENR_CRCEN
 MRCC_private.h, 451
RCC_AHB1ENR_DMA1EN
 MRCC_private.h, 451
RCC_AHB1ENR_DMA2EN
 MRCC_private.h, 451
RCC_AHB1ENR_GPIOAEN
 MRCC_private.h, 452
RCC_AHB1ENR_GPIOBEN
 MRCC_private.h, 452
RCC_AHB1ENR_GPIOCEN
 MRCC_private.h, 452
RCC_AHB1ENR_GPIODEN
 MRCC_private.h, 452
RCC_AHB1ENR_GPIOEEN
 MRCC_private.h, 452
RCC_AHB1ENR_GPIOHEN
 MRCC_private.h, 451
RCC_AHB1LPENR
 ID options, 85
RCC_AHB1LPENR_FLITFLPEN
 MRCC_private.h, 456
RCC_AHB2
 ID options, 85
RCC_AHB2ENR_OTGFSEN
 MRCC_private.h, 452
RCC_APB1
 ID options, 85
RCC_APB1ENR_I2C1EN
 MRCC_private.h, 453
RCC_APB1ENR_I2C2EN
 MRCC_private.h, 453
RCC_APB1ENR_I2C3EN
 MRCC_private.h, 453
RCC_APB1ENR_PWREN
 MRCC_private.h, 453
RCC_APB1ENR_SPI2EN
 MRCC_private.h, 454
RCC_APB1ENR_SPI3EN
 MRCC_private.h, 453
RCC_APB1ENR_TIM2EN
 MRCC_private.h, 454
RCC_APB1ENR_TIM3EN
 MRCC_private.h, 454
RCC_APB1ENR_TIM4EN
 MRCC_private.h, 454
RCC_APB1ENR_TIM5EN
 MRCC_private.h, 454
RCC_APB1ENR_USART2EN
 MRCC_private.h, 453
RCC_APB1ENR_WWDGEN
 MRCC_private.h, 454
RCC_APB2
 ID options, 85
RCC_APB2ENR_ADC1EN
 MRCC_private.h, 456
RCC_APB2ENR_SDIOEN
 MRCC_private.h, 456
RCC_APB2ENR_SPI1EN

MRCC_private.h, 455
 RCC_APB2ENR_SPI4EN
 MRCC_private.h, 455
 RCC_APB2ENR_SYSCFGEN
 MRCC_private.h, 455
 RCC_APB2ENR_TIM10EN
 MRCC_private.h, 455
 RCC_APB2ENR_TIM11EN
 MRCC_private.h, 455
 RCC_APB2ENR_TIM1EN
 MRCC_private.h, 456
 RCC_APB2ENR_TIM9EN
 MRCC_private.h, 455
 RCC_APB2ENR_USART1EN
 MRCC_private.h, 456
 RCC_APB2ENR_USART6EN
 MRCC_private.h, 456
 RCC_BASE_ADDRESS
 MRCC_private.h, 440
 RCC_CFGR_HPRE_b0
 MRCC_private.h, 450
 RCC_CFGR_HPRE_b1
 MRCC_private.h, 450
 RCC_CFGR_HPRE_b2
 MRCC_private.h, 450
 RCC_CFGR_HPRE_b3
 MRCC_private.h, 450
 RCC_CFGR_I2SSRC
 MRCC_private.h, 447
 RCC_CFGR_MOC1_b0
 MRCC_private.h, 447
 RCC_CFGR_MOC1_b1
 MRCC_private.h, 448
 RCC_CFGR_MOC1PRE_b0
 MRCC_private.h, 447
 RCC_CFGR_MOC1PRE_b1
 MRCC_private.h, 447
 RCC_CFGR_MOC1PRE_b2
 MRCC_private.h, 447
 RCC_CFGR_MOC2_b0
 MRCC_private.h, 446
 RCC_CFGR_MOC2_b1
 MRCC_private.h, 446
 RCC_CFGR_MOC2PRE_b0
 MRCC_private.h, 446
 RCC_CFGR_MOC2PRE_b1
 MRCC_private.h, 446
 RCC_CFGR_MOC2PRE_b2
 MRCC_private.h, 447
 RCC_CFGR_PPREG1_b0
 MRCC_private.h, 449
 RCC_CFGR_PPREG1_b1
 MRCC_private.h, 449
 RCC_CFGR_PPREG1_b2
 MRCC_private.h, 449
 RCC_CFGR_PPREG2_b0
 MRCC_private.h, 449
 RCC_CFGR_PPREG2_b1

MRCC_private.h, 449
 RCC_CFGR_PPREG2_b2
 MRCC_private.h, 449
 RCC_CFGR_RTCPRE_b0
 MRCC_private.h, 448
 RCC_CFGR_RTCPRE_b1
 MRCC_private.h, 448
 RCC_CFGR_RTCPRE_b2
 MRCC_private.h, 448
 RCC_CFGR_RTCPRE_b3
 MRCC_private.h, 448
 RCC_CFGR_RTCPRE_b4
 MRCC_private.h, 448
 RCC_CFGR_SW_b0
 MRCC_private.h, 451
 RCC_CFGR_SW_b1
 MRCC_private.h, 451
 RCC_CFGR_SWS_b0
 MRCC_private.h, 450
 RCC_CFGR_SWS_b1
 MRCC_private.h, 450
 RCC_CR_CSSON
 MRCC_private.h, 441
 RCC_CR_HSEBYP
 MRCC_private.h, 441
 RCC_CR_HSEON
 MRCC_private.h, 442
 RCC_CR_HSIRDY
 MRCC_private.h, 442
 RCC_CR_HSION
 MRCC_private.h, 442
 RCC_CR_HSIRDY
 MRCC_private.h, 442
 RCC_CR_PLLI2SON
 MRCC_private.h, 441
 RCC_CR_PLLI2SRDY
 MRCC_private.h, 441
 RCC_CR_PLLON
 MRCC_private.h, 441
 RCC_CR_PLLRDY
 MRCC_private.h, 441
 RCC_MemoryMapType, 125
 AHB1ENR, 129
 AHB1LPENR, 131
 AHB1RSTR, 128
 AHB2ENR, 130
 AHB2LPENR, 131
 AHB2RSTR, 128
 APB1ENR, 130
 APB1LPENR, 132
 APB1RSTR, 129
 APB2ENR, 130
 APB2LPENR, 132
 APB2RSTR, 129
 BDCR, 133
 CFGR, 127
 CIR, 128
 CR, 127

CSR, 133
DCKCFGR, 134
PLLCFGR, 127
PLL12SCFGR, 134
Reserved1, 128
Reserved10, 132
Reserved11, 132
Reserved12, 132
Reserved13, 133
Reserved14, 133
Reserved2, 128
Reserved3, 129
Reserved4, 129
Reserved5, 130
Reserved6, 130
Reserved7, 131
Reserved8, 131
Reserved9, 131
SSCGR, 133
RCC_PLLCFGR_PLLM_b0
 MRCC_private.h, 445
RCC_PLLCFGR_PLLM_b1
 MRCC_private.h, 445
RCC_PLLCFGR_PLLM_b2
 MRCC_private.h, 445
RCC_PLLCFGR_PLLM_b3
 MRCC_private.h, 445
RCC_PLLCFGR_PLLM_b4
 MRCC_private.h, 446
RCC_PLLCFGR_PLLM_b5
 MRCC_private.h, 446
RCC_PLLCFGR_PLLN_b0
 MRCC_private.h, 443
RCC_PLLCFGR_PLLN_b1
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b2
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b3
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b4
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b5
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b6
 MRCC_private.h, 444
RCC_PLLCFGR_PLLN_b7
 MRCC_private.h, 445
RCC_PLLCFGR_PLLN_b8
 MRCC_private.h, 445
RCC_PLLCFGR_PLLP_b0
 MRCC_private.h, 443
RCC_PLLCFGR_PLLP_b1
 MRCC_private.h, 443
RCC_PLLCFGR_PLLQ_b0
 MRCC_private.h, 442
RCC_PLLCFGR_PLLQ_b1
 MRCC_private.h, 442
RCC_PLLCFGR_PLLQ_b2
 MRCC_private.h, 443
RCC_PLLCFGR_PLLQ_b3
 MRCC_private.h, 443
REG_EXT_EVENT_SRC
 ADC_config.h, 232
REG_EXTERNAL_TRIGGER
 ADC_config.h, 232
REG_SQ_LENGTH
 ADC_config.h, 237
REG_START_CONV
 ADC_config.h, 231
REG_T2_TRGO
 ADC_private.h, 302
REGISTER_SIZE
 NVIC_program.c, 403
RELEASED
 Standard values, 23
RES0
 ADC_private.h, 264
RES1
 ADC_private.h, 264
RESERVED
 SCB_MemoryMapType, 139
RESERVED0x
 NVIC_MemoryMapType, 122
Reserved1
 RCC_MemoryMapType, 128
Reserved10
 RCC_MemoryMapType, 132
Reserved11
 RCC_MemoryMapType, 132
Reserved12
 RCC_MemoryMapType, 132
Reserved13
 RCC_MemoryMapType, 133
Reserved14
 RCC_MemoryMapType, 133
Reserved2
 RCC_MemoryMapType, 128
RESERVED2x
 NVIC_MemoryMapType, 123
Reserved3
 RCC_MemoryMapType, 129
RESERVED3x
 NVIC_MemoryMapType, 124
Reserved4
 RCC_MemoryMapType, 129
RESERVED4x
 NVIC_MemoryMapType, 124
Reserved5
 RCC_MemoryMapType, 130
Reserved6
 RCC_MemoryMapType, 130
Reserved7
 RCC_MemoryMapType, 131
Reserved8

RCC_MemoryMapType, 131
 Reserved9
 RCC_MemoryMapType, 131
 RESOLUTION
 ADC_config.h, 229
 RIGHT
 ADC_private.h, 306
 SPI_private.h, 528
 RISING
 ADC_private.h, 303
 RSERVED1x
 NVIC_MemoryMapType, 122
 RTCPRE
 MRCC_config.h, 412
 RTSR
 EXTI_Type, 112
 RUN
 Standard values, 22
 RX1_PIN
 UART_config.h, 575
 RX2_PIN
 UART_config.h, 576
 RX6_PIN
 UART_config.h, 576
 RX_ONLY
 SPI_private.h, 526
 UART_interface.h, 580
 RXCRCR
 SPI_MemoryMapType, 141
 RXDMAEN
 SPI_private.h, 520
 RXDMAEN_MODE
 SPI_config.h, 498
 RXNE
 SPI_private.h, 521
 RXNEIE
 SPI_private.h, 521
 RXNEIE_MODE
 SPI_config.h, 499, 500
 RXONLY
 SPI_private.h, 519
 RXONLY_MODE
 SPI_config.h, 497
 s16_t
 Standard types, 19
 s32_t
 Standard types, 19
 s64_t
 Standard types, 19
 s8_t
 Standard types, 19
 SAME_STRING
 Standard values, 23
 SCAN
 ADC_private.h, 265
 SCAN_MODE
 ADC_config.h, 231
 SCB_BASE_ADDRESS
 NVIC Addresses, 81
 SCB_MemoryMapType, 134
 AIRCR, 136
 BFAR, 140
 CCR, 137
 CFSR, 138
 CPUID, 135
 HFSR, 139
 ICSR, 136
 MMFAR, 139
 RESERVED, 139
 SCR, 136
 SHCSR, 138
 SHPR1, 137
 SHPR2, 137
 SHPR3, 138
 VTOR, 136
 SCR
 SCB_MemoryMapType, 136
 SDIO
 NVIC Vector Table, 77
 SDIOEN
 MRCC_config.h, 418
 SEC
 Delay Units, 97
 SEC_ROW_START
 LCD_config.h, 183
 SECOND_CLK_Captured
 SPI_private.h, 523
 SET_BIT
 Bit Manipulation Math Macros, 11
 SET_BITS
 Bit Group Manipulation Math Macros, 12
 SET_CGRAM_AC_MASK
 LCD_config.h, 183
 SET_DDRAM_AC_MASK
 LCD_config.h, 183
 SetCursorBlinkingOFFCmd
 LCD_private.h, 199
 SEVEN
 ADC_private.h, 299
 SEVENTEEN
 ADC_private.h, 300
 SHCSR
 SCB_MemoryMapType, 138
 SHPR1
 SCB_MemoryMapType, 137
 SHPR2
 SCB_MemoryMapType, 137
 SHPR3
 SCB_MemoryMapType, 138
 SIMPLEX_MODE
 SPI_config.h, 498
 SIMPLEX_RX
 SPI_private.h, 530
 SIMPLEX_TX
 SPI_private.h, 530
 SINGLE_INTERVAL_MODE

Systick Interval Mode Configuration, 96
SINGLE_INTERVAL_TICK_TIME
 SysTick_interface.h, 553
SIX
 ADC_private.h, 299
SIXTEEN
 ADC_private.h, 300
SIXTEEN_BITS
 SPI_private.h, 526
SLAVE
 SPI_private.h, 523
SMP0
 ADC_config.h, 235
SMP0_b0
 ADC_private.h, 275
SMP0_b1
 ADC_private.h, 275
SMP0_b2
 ADC_private.h, 276
SMP1
 ADC_config.h, 235
SMP10
 ADC_config.h, 233
SMP10_b0
 ADC_private.h, 270
SMP10_b1
 ADC_private.h, 270
SMP10_b2
 ADC_private.h, 271
SMP11
 ADC_config.h, 234
SMP11_b0
 ADC_private.h, 271
SMP11_b1
 ADC_private.h, 271
SMP11_b2
 ADC_private.h, 271
SMP12
 ADC_config.h, 234
SMP12_b0
 ADC_private.h, 271
SMP12_b1
 ADC_private.h, 271
SMP12_b2
 ADC_private.h, 272
SMP13
 ADC_config.h, 234
SMP13_b0
 ADC_private.h, 272
SMP13_b1
 ADC_private.h, 272
SMP13_b2
 ADC_private.h, 272
SMP14
 ADC_config.h, 234
SMP14_b0
 ADC_private.h, 272
SMP14_b1
 ADC_private.h, 272
SMP15
 ADC_config.h, 234
SMP15_b0
 ADC_private.h, 273
SMP15_b1
 ADC_private.h, 273
SMP15_b2
 ADC_private.h, 273, 274
SMP16
 ADC_config.h, 234
SMP16_b0
 ADC_private.h, 274
SMP16_b1
 ADC_private.h, 274
SMP16_b2
 ADC_private.h, 274
SMP17
 ADC_config.h, 235
SMP17_b0
 ADC_private.h, 274
SMP17_b1
 ADC_private.h, 274
SMP17_b2
 ADC_private.h, 275
SMP18
 ADC_config.h, 235
SMP18_b0
 ADC_private.h, 275
SMP18_b1
 ADC_private.h, 275
SMP18_b2
 ADC_private.h, 275
SMP1_b0
 ADC_private.h, 276
SMP1_b1
 ADC_private.h, 276
SMP1_b2
 ADC_private.h, 276
SMP2
 ADC_config.h, 235
SMP2_b0
 ADC_private.h, 276
SMP2_b1
 ADC_private.h, 276
SMP2_b2
 ADC_private.h, 277
SMP3
 ADC_config.h, 235
SMP3_b0
 ADC_private.h, 277
SMP3_b1
 ADC_private.h, 277
SMP3_b2
 ADC_private.h, 277
SMP4
 ADC_private.h, 277

ADC_config.h, 236
 SMP4_b0
 ADC_private.h, 277
 SMP4_b1
 ADC_private.h, 277
 SMP4_b2
 ADC_private.h, 278
 SMP5
 ADC_config.h, 236
 SMP5_b0
 ADC_private.h, 278
 SMP5_b1
 ADC_private.h, 278
 SMP5_b2
 ADC_private.h, 278, 279
 SMP6
 ADC_config.h, 236
 SMP6_b0
 ADC_private.h, 279
 SMP6_b1
 ADC_private.h, 279
 SMP6_b2
 ADC_private.h, 279
 SMP7
 ADC_config.h, 236
 SMP7_b0
 ADC_private.h, 279
 SMP7_b1
 ADC_private.h, 279
 SMP7_b2
 ADC_private.h, 280
 SMP8
 ADC_config.h, 236
 SMP8_b0
 ADC_private.h, 280
 SMP8_b1
 ADC_private.h, 280
 SMP8_b2
 ADC_private.h, 280
 SMP9
 ADC_config.h, 236
 SMP9_b0
 ADC_private.h, 280
 SMP9_b1
 ADC_private.h, 280
 SMP9_b2
 ADC_private.h, 281
 SMPR1
 ADC_MemoryMapType, 105
 SMPR2
 ADC_MemoryMapType, 105
 SPE
 SPI_private.h, 518
 SPE_MODE
 SPI_config.h, 496
 SPI1
 NVIC Vector Table, 75
 SPI_interface.h, 505
 SPI1_BASE_ADDRESS
 SPI_interface.h, 505
 SPI1_IRQHandler
 SPI_program.c, 541
 SPI1EN
 MRCC_config.h, 418
 SPI2
 NVIC Vector Table, 75
 SPI_interface.h, 505
 SPI2_BASE_ADDRESS
 SPI_interface.h, 505
 SPI2_IRQHandler
 SPI_program.c, 541
 SPI2EN
 MRCC_config.h, 416
 SPI3
 NVIC Vector Table, 77
 SPI_interface.h, 506
 SPI3_BASE_ADDRESS
 SPI_interface.h, 505
 SPI3_IRQHandler
 SPI_program.c, 541
 SPI3EN
 MRCC_config.h, 416
 SPI4
 NVIC Vector Table, 80
 SPI4_BASE_ADDRESS
 SPI_interface.h, 505
 SPI4EN
 MRCC_config.h, 418
 SPI_config.h
 BIDIOE_MODE, 497
 BIDMODE_MODE, 497
 BR_MODE, 496
 BSY_MODE, 499
 CPHA_MODE, 495
 CPOL_MODE, 496
 CRCEN_MODE, 497
 CRCERR_MODE, 500
 CRCNEXT_MODE, 497
 DFF_MODE, 497
 ERRIE_MODE, 499
 FRE_MODE, 499
 FRF_MODE, 498
 HALF_DUPLEX_MODE, 498
 LSBFIRST_MODE, 496
 MODF_MODE, 500
 MSTR_MODE, 496
 MULTI_MSTR_MODE, 498
 OVR_MODE, 499
 RXDMAEN_MODE, 498
 RXNEIE_MODE, 499, 500
 RXONLY_MODE, 497
 SIMPLEX_MODE, 498
 SPE_MODE, 496
 SSM_MODE, 496
 TXDMAEN_MODE, 498
 TXNEIE_MODE, 499, 500

UDR_MODE, 500
SPI_interface.h
 MSPI1_vSetCallBack, 514
 MSPI2_vSetCallBack, 514
 MSPI3_vSetCallBack, 515
 MSPI_u16MasterRecieve, 511
 MSPI_u16SlaveReceive, 512
 MSPI_u16Transcieve, 510
 MSPI_vDISABLE, 512
 MSPI_vInit, 507
 MSPI_vMasterTransmit, 511
 MSPI_vSlaveTransmit, 512
 SPI1, 505
 SPI1_BASE_ADDRESS, 505
 SPI2, 505
 SPI2_BASE_ADDRESS, 505
 SPI3, 506
 SPI3_BASE_ADDRESS, 505
 SPI4_BASE_ADDRESS, 505
 SPIx_FULL_DUPLEX, 506
 SPIx_HALF_DUPLEX, 506
 SPIx_MSTR, 506
 SPIx_SIMPLEX, 506
 SPIx_SLAVE, 506
SPI_MemoryMapType, 140
 CR1, 141
 CR2, 141
 CRCPR, 141
 DR, 141
 I2SCFGR, 142
 I2SPR, 142
 RXCRCR, 141
 SR, 141
 TXCRCR, 142
SPI_private.h
 BIDIMODE, 520
 BIDIOE, 519
 BSY, 522
 BUSY, 529
 CHSIDE, 521
 CLK_By128, 524
 CLK_By16, 524
 CLK_By2, 523
 CLK_By256, 525
 CLK_By32, 524
 CLK_By4, 524
 CLK_By64, 524
 CLK_By8, 524
 CLK_IdleAt0, 523
 CLK_IdleAt1, 523
 CPHA, 518
 CPOL, 518
 CRC_PHASE, 526
 CRCEN, 519
 CRCERR, 522
 CRCNEXT, 519
 DFF, 519
 DISABLE, 525
 EIGHT_BITS, 526
 EMPTY, 527
 ENABLE, 525
 ERRIE, 520
 ERROR, 529
 FAULT_OCCURRED, 528
 FIRST_CLK_Captured, 522
 FRE, 522
 FRF, 520
 FULL_DUPLEX, 525
 HALF_DUPLEX_RX, 530
 HALF_DUPLEX_TX, 529
 LEFT, 527
 LSB_First, 525
 LSBFIRST, 518
 MASTER, 523
 MATCHED, 528
 MODF, 522
 MOTOROLA, 527
 MSB_First, 525
 MSTR, 518
 NO_CRC_PHASE, 526
 NO_ERROR, 529
 NO_FAULT_OCCURRED, 529
 NOT_BUSY, 529
 NOT_MATCHED, 528
 NOT_OCCURRED, 528
 NOTEEMPTY, 527
 OCCURRED, 528
 OneLine, 526
 OVR, 522
 RIGHT, 528
 RX_ONLY, 526
 RXDMAEN, 520
 RXNE, 521
 RXNEIE, 521
 RXONLY, 519
 SECOND_CLK_Captured, 523
 SIMPLEX_RX, 530
 SIMPLEX_TX, 530
 SIXTEEN_BITS, 526
 SLAVE, 523
 SPE, 518
 SR_RESET, 530
 SSI, 518
 SSM, 519
 SSOE, 520
 TI, 527
 TwoLines, 527
 TXDMAEN, 520
 TXE, 521
 TXEIE, 521
 UDR, 521
SPI_program.c
 MSPI1_CallBack, 542
 MSPI1_vSetCallBack, 540
 MSPI2_CallBack, 542
 MSPI2_vSetCallBack, 540

MSPI3_CallBack, 542
MSPI3_vSetCallBack, 541
MSPI_u16MasterRecieve, 537
MSPI_u16SlaveRecieve, 538
MSPI_u16Transcieve, 537
MSPI_vDISABLE, 538
MSPI_vInit, 533
MSPI_vMasterTransmit, 537
MSPI_vSlaveTransmit, 538
SPI1_IRQHandler, 541
SPI2_IRQHandler, 541
SPI3_IRQHandler, 541
SPIx_FULL_DUPLEX
 SPI_interface.h, 506
SPIx_HALF_DUPLEX
 SPI_interface.h, 506
SPIx_MSTR
 SPI_interface.h, 506
SPIx_SIMPLEX
 SPI_interface.h, 506
SPIx_SLAVE
 SPI_interface.h, 506
SQ10_b0
 ADC_private.h, 286
SQ10_b1
 ADC_private.h, 287
SQ10_b2
 ADC_private.h, 287
SQ10_b3
 ADC_private.h, 287
SQ10_b4
 ADC_private.h, 287
SQ11_b0
 ADC_private.h, 286
SQ11_b1
 ADC_private.h, 286
SQ11_b2
 ADC_private.h, 286
SQ11_b3
 ADC_private.h, 286
SQ11_b4
 ADC_private.h, 286
SQ12_b0
 ADC_private.h, 285
SQ12_b1
 ADC_private.h, 285
SQ12_b2
 ADC_private.h, 285
SQ12_b3
 ADC_private.h, 285
SQ12_b4
 ADC_private.h, 285
SQ13_b0
 ADC_private.h, 284
SQ13_b1
 ADC_private.h, 284
SQ13_b2
 ADC_private.h, 284
SQ13_b3
 ADC_private.h, 284
SQ13_b4
 ADC_private.h, 285
SQ14_b0
 ADC_private.h, 283
SQ14_b1
 ADC_private.h, 283
SQ14_b2
 ADC_private.h, 283
SQ14_b3
 ADC_private.h, 284
SQ14_b4
 ADC_private.h, 284
SQ15_b0
 ADC_private.h, 282
SQ15_b1
 ADC_private.h, 282
SQ15_b2
 ADC_private.h, 283
SQ15_b3
 ADC_private.h, 283
SQ15_b4
 ADC_private.h, 283
SQ16_b0
 ADC_private.h, 281
SQ16_b1
 ADC_private.h, 282
SQ16_b2
 ADC_private.h, 282
SQ16_b3
 ADC_private.h, 282
SQ16_b4
 ADC_private.h, 282
SQ1_b0
 ADC_private.h, 294
SQ1_b1
 ADC_private.h, 294
SQ1_b2
 ADC_private.h, 294
SQ1_b3
 ADC_private.h, 294
SQ1_b4
 ADC_private.h, 295
SQ1_BIT_MANIPULATION
 ADC_config.h, 237
SQ2_b0
 ADC_private.h, 293
SQ2_b1
 ADC_private.h, 293
SQ2_b2
 ADC_private.h, 293
SQ2_b3
 ADC_private.h, 294
SQ2_b4
 ADC_private.h, 294
SQ3_b0
 ADC_private.h, 292

SQ3_b1
 ADC_private.h, 292
SQ3_b2
 ADC_private.h, 293
SQ3_b3
 ADC_private.h, 293
SQ3_b4
 ADC_private.h, 293
SQ4_b0
 ADC_private.h, 291
SQ4_b1
 ADC_private.h, 292
SQ4_b2
 ADC_private.h, 292
SQ4_b3
 ADC_private.h, 292
SQ4_b4
 ADC_private.h, 292
SQ5_b0
 ADC_private.h, 291
SQ5_b1
 ADC_private.h, 291
SQ5_b2
 ADC_private.h, 291
SQ5_b3
 ADC_private.h, 291
SQ5_b4
 ADC_private.h, 291
SQ6_b0
 ADC_private.h, 290
SQ6_b1
 ADC_private.h, 290
SQ6_b2
 ADC_private.h, 290
SQ6_b3
 ADC_private.h, 290
SQ6_b4
 ADC_private.h, 290
SQ7_b0
 ADC_private.h, 289
SQ7_b1
 ADC_private.h, 289
SQ7_b2
 ADC_private.h, 289
SQ7_b3
 ADC_private.h, 289
SQ7_b4
 ADC_private.h, 290
SQ8_b0
 ADC_private.h, 288
SQ8_b1
 ADC_private.h, 288
SQ8_b2
 ADC_private.h, 288
SQ8_b3
 ADC_private.h, 289
SQ8_b4
 ADC_private.h, 289

SQ9_b0
 ADC_private.h, 287
SQ9_b1
 ADC_private.h, 287
SQ9_b2
 ADC_private.h, 288
SQ9_b3
 ADC_private.h, 288
SQ9_b4
 ADC_private.h, 288

SQR1
 ADC_MemoryMapType, 106
SQR2
 ADC_MemoryMapType, 106
SQR3
 ADC_MemoryMapType, 107

SR
 ADC_MemoryMapType, 104
 SPI_MemoryMapType, 141
SR_REG
 USART_MemoryMapType, 148
SR_RESET
 SPI_private.h, 530

SSCGR
 RCC_MemoryMapType, 133

SSI
 SPI_private.h, 518

SSM
 SPI_private.h, 519

SSM_MODE
 SPI_config.h, 496

SSOE
 SPI_private.h, 520

Standard types, 17

- bool_t, 17
- c8_t, 18
- f32_t, 20
- f64_t, 20
- s16_t, 19
- s32_t, 19
- s64_t, 19
- s8_t, 19
- u16_t, 18
- u32_t, 18
- u64_t, 18
- u8_t, 17

Standard values, 20

- DIFFERENT_STRING, 23
- FALSE, 21
- FLAG_CLEARED, 22
- FLAG_SET, 21
- INITIAL_ZERO, 21
- NULL, 21
- PRESSED, 22
- RELEASED, 23
- RUN, 22
- SAME_STRING, 23
- STOP, 22

TRUE, 21
STATIC Compiler standard macros, 16
STOP Standard values, 22
STOP_BIT_0_5 UART_interface.h, 582
STOP_BIT_1 Standard values, 22
STOP_BIT_1_5 UART_interface.h, 582
STOP_BIT_2 Standard values, 22
StopBits USART_InitType, 146
strPassword UserInfo_t, 150
STRT ADC_private.h, 263
strUsername UserInfo_t, 150
Sub-Group priorities, 59
 NO_SUB_PRIORITY, 60
 SUB_PRIORITY_0, 60
 SUB_PRIORITY_1, 60
 SUB_PRIORITY_10, 63
 SUB_PRIORITY_11, 63
 SUB_PRIORITY_12, 63
 SUB_PRIORITY_13, 63
 SUB_PRIORITY_14, 64
 SUB_PRIORITY_15, 64
 SUB_PRIORITY_2, 61
 SUB_PRIORITY_3, 61
 SUB_PRIORITY_4, 61
 SUB_PRIORITY_5, 61
 SUB_PRIORITY_6, 62
 SUB_PRIORITY_7, 62
 SUB_PRIORITY_8, 62
 SUB_PRIORITY_9, 62
SUB_PRIORITY_0 Sub-Group priorities, 60
SUB_PRIORITY_1 Sub-Group priorities, 60
SUB_PRIORITY_10 Sub-Group priorities, 63
SUB_PRIORITY_11 Sub-Group priorities, 63
SUB_PRIORITY_12 Sub-Group priorities, 63
SUB_PRIORITY_13 Sub-Group priorities, 63
SUB_PRIORITY_14 Sub-Group priorities, 64
SUB_PRIORITY_15 Sub-Group priorities, 64
SUB_PRIORITY_2 Sub-Group priorities, 61
SUB_PRIORITY_3 Sub-Group priorities, 61
Sub-Group priorities, 61
 SUB_PRIORITY_4 Sub-Group priorities, 61
 SUB_PRIORITY_5 Sub-Group priorities, 61
 SUB_PRIORITY_6 Sub-Group priorities, 62
 SUB_PRIORITY_7 Sub-Group priorities, 62
 SUB_PRIORITY_8 Sub-Group priorities, 62
 SUB_PRIORITY_9 Sub-Group priorities, 62
SV_CALL NVIC Settable Priorities, 83
SW MRCC_config.h, 413
SWIER EXTI_Type, 113
SWS MRCC_config.h, 413
SWSTART ADC_private.h, 267
SYSCFG Memory Addresses, 32
 SYSCFG_BASE_ADDR, 32
SYSCFG Memory Registers, 32
 MSYSCFG, 32
SYSCFG_BASE_ADDR SYSCFG Memory Addresses, 32
SYSCFGEN MRCC_config.h, 417
SYSCLK MRCC_private.h, 457
SYSCLKby128 MRCC_private.h, 460
SYSCLKby16 MRCC_private.h, 460
SYSCLKby2 MRCC_private.h, 459
SYSCLKby256 MRCC_private.h, 460
SYSCLKby4 MRCC_private.h, 460
SYSCLKby512 MRCC_private.h, 461
SYSCLKby64 MRCC_private.h, 460
SYSCLKby8 MRCC_private.h, 460
SYSTICK NVIC Settable Priorities, 83
SysTick Systick Registers, 98
 Systick Addresses, 98
 SysTick_BASE_ADDRESS, 98
Systick Clock Sources, 100
 AHB, 100
 AHB_DividedBy8, 100

Systick Exception (Interrupt) Status, 101
 AssertRequest, 101
 Dont AssertRequest, 101
Systick Interval Mode Configuration, 96
 PERIODIC_INTERVAL_MODE, 96
 SINGLE_INTERVAL_MODE, 96
Systick Register Bits Positions, 99
 CLKSOURCE, 99
 COUNTER_ENABLE, 99
 COUNTFLAG, 99
 TICKINT, 99
Systick Registers, 98
 SysTick, 98
SysTick_BASE_ADDRESS
 Systick Addresses, 98
SysTick_config.h
 Exception_Request, 550
 MAX_TICKS, 550
SysTick_Handler
 SysTick_program.c, 568
SysTick_interface.h
 BUSY_TICK_TIME, 553
 MSysTick_u32GetElapsedTime, 557
 MSysTick_u32GetRemainingTime, 557
 MSysTick_vDelay, 554
 MSysTick_vDelayMicroSec, 555
 MSysTick_vDelayMilliSec, 555
 MSysTick_vDelaySec, 555
 MSysTick_vDisable, 558
 MSysTick_vEnable, 557
 MSysTick_vEnableException, 558
 MSysTick_vInit, 553
 MSysTick_vSetBusyWait, 554
 MSysTick_vSetPeriodicInterval, 556
 MSysTick_vSetSingleInterval, 556
 MSysTick_vStopInterval, 556
 PERIODIC_INTERVAL_TICK_TIME, 553
 SINGLE_INTERVAL_TICK_TIME, 553
SysTick_program.c
 MSysTick_u32GetElapsedTime, 567
 MSysTick_u32GetRemainingTime, 567
 MSysTick_vDelay, 563
 MSysTick_vDelayMicroSec, 564
 MSysTick_vDelayMilliSec, 564
 MSysTick_vDelaySec, 565
 MSysTick_vDisable, 568
 MSysTick_vEnable, 568
 MSysTick_vEnableException, 562
 MSysTick_vInit, 562
 MSysTick_vSetBusyWait, 562
 MSysTick_vSetPeriodicInterval, 566
 MSysTick_vSetSingleInterval, 565
 MSysTick_vStopInterval, 566
 SysTick_Handler, 568
SysTick_Type, 142
 CALIB, 143
 CTRL, 143
 LOAD, 143
VAL, 143
T1_CC1
 ADC_private.h, 301
T1_CC2
 ADC_private.h, 301
T1_CC3
 ADC_private.h, 301
T1_CC4
 ADC_private.h, 304
T1_TRGO
 ADC_private.h, 304
T2_CC1
 ADC_private.h, 304
T2_CC2
 ADC_private.h, 301
T2_CC3
 ADC_private.h, 301
T2_CC4
 ADC_private.h, 302
T3_CC1
 ADC_private.h, 302
T3_CC2
 ADC_private.h, 304
T3_CC4
 ADC_private.h, 304
T3_TRGO
 ADC_private.h, 302
T4_CC1
 ADC_private.h, 305
T4_CC2
 ADC_private.h, 305
T4_CC3
 ADC_private.h, 305
T4_CC4
 ADC_private.h, 302
T4_TRGO
 ADC_private.h, 305
T5_CC1
 ADC_private.h, 302
T5_CC2
 ADC_private.h, 303
T5_CC3
 ADC_private.h, 303
T5_CC4
 ADC_private.h, 305
T5_TRGO
 ADC_private.h, 305
TEMP_SENSOR_AND_VREF
 ADC_config.h, 237
TEN
 ADC_private.h, 299
THERTEEN
 ADC_private.h, 300
THREE
 ADC_private.h, 298
THRESHOLD_VALUE
 UART_config.h, 575
TI

SPI_private.h, 527
TICKINT
 Systick Register Bits Positions, 99
TIM10EN
 MRCC_config.h, 417
TIM11EN
 MRCC_config.h, 417
TIM1_BRK_TIM9
 NVIC Vector Table, 72
TIM1_CC
 NVIC Vector Table, 73
TIM1_TRG_COM_TIM11
 NVIC Vector Table, 72
TIM1_UP_TIM10
 NVIC Vector Table, 72
TIM1EN
 MRCC_config.h, 419
TIM2
 NVIC Vector Table, 73
TIM2EN
 MRCC_config.h, 417
TIM3
 NVIC Vector Table, 73
TIM3EN
 MRCC_config.h, 417
TIM4
 NVIC Vector Table, 73
TIM4EN
 MRCC_config.h, 416
TIM5
 NVIC Vector Table, 77
TIM5EN
 MRCC_config.h, 416
TIM9EN
 MRCC_config.h, 417
TOGGLE_BIT
 Bit Manipulation Math Macros, 11
TOGGLE_BItS
 Bit Group Manipulation Math Macros, 13
TransferDirection
 USART_InitType, 147
TriggerStatus
 EXTI_ConfigType, 111
TRUE
 Standard values, 21
TSVREFE
 ADC_private.h, 295
TWELVE
 ADC_private.h, 300
TwoLines
 SPI_private.h, 527
TX1_PIN
 UART_config.h, 575
TX2_PIN
 UART_config.h, 575
TX6_PIN
 UART_config.h, 576
TX_ONLY
 UART_config.h, 576
 UART_interface.h, 580
TX_RX
 UART_interface.h, 581
TXCRCR
 SPI_MemoryMapType, 142
TXDMAEN
 SPI_private.h, 520
TXDMAEN_MODE
 SPI_config.h, 498
TXE
 SPI_private.h, 521
TXEIE
 SPI_private.h, 521
TXNEIE_MODE
 SPI_config.h, 499, 500
u16_t
 Standard types, 18
u32_t
 Standard types, 18
u64_t
 Standard types, 18
u8_t
 Standard types, 17
u8Direction
 DCM_MotorConfiguration, 110
u8Pin1
 DCM_MotorConfiguration, 109
u8Pin2
 DCM_MotorConfiguration, 109
u8Port
 DCM_MotorConfiguration, 109
UART_BRR_SAMPLING16
 UART_private.h, 594
UART_BRR_SAMPLING8
 UART_private.h, 595
UART_config.h
 __PCLK__, 574
 RX1_PIN, 575
 RX2_PIN, 576
 RX6_PIN, 576
 THRESHOLD_VALUE, 575
 TX1_PIN, 575
 TX2_PIN, 575
 TX6_PIN, 576
 USART1_PORT, 575
 USART2_PORT, 575
 USART6_PORT, 576
UART_DIV_SAMPLING16
 UART_private.h, 594
UART_DIV_SAMPLING8
 UART_private.h, 595
UART_DIVFRAQ_SAMPLING16
 UART_private.h, 594
UART_DIVFRAQ_SAMPLING8
 UART_private.h, 595
UART_DIVMANT_SAMPLING16
 UART_private.h, 594
UART_DIVMANT_SAMPLING8

UART_private.h, 595
UART_interface.h
 __BAUDRATE__, 582
 DISABLE, 582
 ENABLE, 582
 EVEN_PARITY, 581
 MODE_8BIT, 581
 MODE_9BIT, 581
 MUSART1_vSetCallBack, 590
 MUSART2_vSetCallBack, 590
 MUSART6_vSetCallBack, 590
 MUSART_ptrReceiveStringSynchNonBlocking,
 587
 MUSART_u8CompareString, 588
 MUSART_u8ReadDataRegister, 589
 MUSART_u8ReceiveByteSynchBlocking, 587
 MUSART_u8ReceiveByteSynchNonBlocking, 586
 MUSART_vClearFlags, 589
 MUSART_vDisable, 585
 MUSART_vEnable, 585
 MUSART_vInit, 583
 MUSART_vRecieveString, 588
 MUSART_vRxIntSetStatus, 589
 MUSART_vTransmitByte, 585
 MUSART_vTransmitString, 586
 ODD_PARITY, 581
 OVER_SAMPLING_16, 580
 OVER_SAMPLING_8, 580
 RX_ONLY, 580
 STOP_BIT_0_5, 582
 STOP_BIT_1, 581
 STOP_BIT_1_5, 582
 STOP_BIT_2, 582
 TX_ONLY, 580
 TX_RX, 581
 USART1_BASE_ADDRESS, 579
 USART1_REG, 579
 USART2_BASE_ADDRESS, 579
 USART2_REG, 580
 USART6_BASE_ADDRESS, 579
 USART6_REG, 580
UART_private.h
 MUSART_CR1_IDLEIE_BIT, 598
 MUSART_CR1_M_BIT, 599
 MUSART_CR1_OVER8_BIT, 600
 MUSART_CR1_PCE_BIT, 599
 MUSART_CR1_PEIE_BIT, 599
 MUSART_CR1_PS_BIT, 599
 MUSART_CR1_RE_BIT, 598
 MUSART_CR1_RWU_BIT, 598
 MUSART_CR1_RXNEIE_BIT, 598
 MUSART_CR1_SBK_BIT, 597
 MUSART_CR1_TCIE_BIT, 598
 MUSART_CR1_TE_BIT, 598
 MUSART_CR1_TXEIE_BIT, 599
 MUSART_CR1_UE_BIT, 600
 MUSART_CR1_WAKE_BIT, 599
 MUSART_CR2_ADD0_BIT, 600
MUSART_CR2_ADD1_BIT, 600
MUSART_CR2_ADD2_BIT, 600
MUSART_CR2_ADD3_BIT, 600
MUSART_CR2_CLKEN_BIT, 601
MUSART_CR2_CPHA_BIT, 601
MUSART_CR2_CPOL_BIT, 601
MUSART_CR2_LBCL_BIT, 601
MUSART_CR2_LBDIE_BIT, 601
MUSART_CR2_LBDL_BIT, 601
MUSART_CR2_LINEN_BIT, 602
MUSART_CR2_STOP0_BIT, 602
MUSART_CR2_STOP1_BIT, 602
MUSART_CR2_STOP_BIT, 602
MUSART_CR3_CTSE_BIT, 602
MUSART_CR3_CTSIE_BIT, 602
MUSART_CR3_DMAR_BIT, 603
MUSART_CR3_DMAT_BIT, 603
MUSART_CR3_EIE_BIT, 604
MUSART_CR3_HDSEL_BIT, 603
MUSART_CR3_IREN_BIT, 604
MUSART_CR3_IRLP_BIT, 604
MUSART_CR3_NACK_BIT, 603
MUSART_CR3_RTSE_BIT, 603
MUSART_CR3_SCEN_BIT, 603
MUSART_SR_CTS_BIT, 597
MUSART_SR_FE_BIT, 596
MUSART_SR_IDLE_BIT, 596
MUSART_SR_LBD_BIT, 597
MUSART_SR_NE_BIT, 596
MUSART_SR_ORE_BIT, 596
MUSART_SR_PE_BIT, 596
MUSART_SR_RXNE_BIT, 597
MUSART_SR_TC_BIT, 597
MUSART_SR_TXE_BIT, 597
UART_BRR_SAMPLING16, 594
UART_BRR_SAMPLING8, 595
UART_DIV_SAMPLING16, 594
UART_DIV_SAMPLING8, 595
UART_DIVFRAQ_SAMPLING16, 594
UART_DIVFRAQ_SAMPLING8, 595
UART_DIVMANT_SAMPLING16, 594
UART_DIVMANT_SAMPLING8, 595
UART_program.c
 G_u8String, 615
 MUSART1_CallBack, 616
 MUSART1_vSetCallBack, 614
 MUSART2_CallBack, 616
 MUSART2_vSetCallBack, 614
 MUSART6_CallBack, 616
 MUSART6_vSetCallBack, 614
 MUSART_ptrReceiveStringSynchNonBlocking,
 611
 MUSART_u8CompareString, 612
 MUSART_u8ReadDataRegister, 613
 MUSART_u8ReceiveByteSynchBlocking, 611
 MUSART_u8ReceiveByteSynchNonBlocking, 610
 MUSART_vClearFlags, 613
 MUSART_vDisable, 609

MUSART_vEnable, 609
 USART_vInit, 607
 USART_vReceiveString, 612
 USART_vRxIntSetStatus, 613
 USART_vTransmitByte, 609
 USART_vTransmitString, 610
 USART1_IRQHandler, 614
 USART2_IRQHandler, 615
 USART6_IRQHandler, 615

UDR
 SPI_private.h, 521

UDR_MODE
 SPI_config.h, 500

USAGE_FAULT
 NVIC Settable Priorities, 84

USART1
 NVIC Vector Table, 75

USART1_BASE_ADDRESS
 UART_interface.h, 579

USART1_IRQHandler
 UART_program.c, 614

USART1_PORT
 UART_config.h, 575

USART1_REG
 UART_interface.h, 579

USART1EN
 MRCC_config.h, 418

USART2
 NVIC Vector Table, 75

USART2_BASE_ADDRESS
 UART_interface.h, 579

USART2_IRQHandler
 UART_program.c, 615

USART2_PORT
 UART_config.h, 575

USART2_REG
 UART_interface.h, 580

USART2EN
 MRCC_config.h, 416

USART6
 NVIC Vector Table, 80

USART6_BASE_ADDRESS
 UART_interface.h, 579

USART6_IRQHandler
 UART_program.c, 615

USART6_PORT
 UART_config.h, 576

USART6_REG
 UART_interface.h, 580

USART6EN
 MRCC_config.h, 418

USART_ClockInitTypeDef, 144
 ClockOutput, 144
 ClockPhase, 145
 ClockPolarity, 144
 LastBitClockPulse, 145

USART_InitType, 145
 BaudRate, 146

DataWidth, 146
 HardwareFlowControl, 147
 Oversampling, 147
 Parity_Enable, 146
 Parity_Selection, 146
 StopBits, 146
 TransferDirection, 147

USART_MemoryMapType, 147

BRR_REG, 148
 CR1_REG, 148
 CR2_REG, 149
 CR3_REG, 149
 DR_REG, 148
 GTPR_REG, 149
 SR_REG, 148

UserInfo_t, 149
 strPassword, 150
 strUsername, 150

UsersLogin_interface.h
 UsersLogin_vUserLoginProcess, 622

UsersLogin_private.h
 MAX_INFO_LENGTH, 623

UsersLogin_program.c
 UsersLogin_vUserLoginProcess, 624

UsersLogin_vUserLoginProcess
 UsersLogin_interface.h, 622
 UsersLogin_program.c, 624

Utilities macros, 16
 MY_MS_DELAY, 16

V_BATTERY
 ADC_config.h, 237

VAL
 SysTick_Type, 143

VAR
 Compiler standard macros, 14

VBATE
 ADC_private.h, 295

VECTKEY_PASSWORD
 NVIC_private.h, 401

VERY_BRIGHT
 LDR_interface.h, 218

VERY_DIM
 LDR_interface.h, 218

VTOR
 SCB_MemoryMapType, 136

WWDG
 NVIC Vector Table, 67

WWDGGEN
 MRCC_config.h, 416

ZERO
 ADC_private.h, 298