

## STM32 Blue Pill Drivers

Generated on Fri Jun 30 2023 13:09:10 for STM32 Blue Pill Drivers by Doxygen 1.9.5

Fri Jun 30 2023 13:09:10



<b>1 STM32 Blue Pill Drivers</b>	<b>1</b>
<b>2 Testing applications</b>	<b>3</b>
2.1 ## Applications list . . . . .	3
2.2 ### List . . . . .	3
<b>3 Module Index</b>	<b>5</b>
3.1 Modules . . . . .	5
<b>4 Data Structure Index</b>	<b>7</b>
4.1 Data Structures . . . . .	7
<b>5 File Index</b>	<b>9</b>
5.1 File List . . . . .	9
<b>6 Module Documentation</b>	<b>11</b>
6.1 Bit Manipulation Math Macros . . . . .	11
6.1.1 Detailed Description . . . . .	11
6.1.2 Macro Definition Documentation . . . . .	11
6.1.2.1 SET_BIT . . . . .	11
6.1.2.2 CLEAR_BIT . . . . .	12
6.1.2.3 TOGGLE_BIT . . . . .	12
6.1.2.4 GET_BIT . . . . .	12
6.2 Compiler standard macros . . . . .	13
6.2.1 Detailed Description . . . . .	13
6.2.2 Macro Definition Documentation . . . . .	13
6.2.2.1 CONST . . . . .	13
6.2.2.2 STATIC . . . . .	14
6.2.2.3 VOLATILE . . . . .	14
6.2.2.4 P2VAR . . . . .	14
6.2.2.5 P2CONST . . . . .	14
6.2.2.6 CONSTP2VAR . . . . .	15
6.2.2.7 CONSTP2CONST . . . . .	15
6.2.2.8 P2FUNC . . . . .	15
6.3 Hardware registers macro-functions . . . . .	15
6.3.1 Detailed Description . . . . .	16
6.3.2 Macro Definition Documentation . . . . .	16
6.3.2.1 REGISTER_ADDRESS . . . . .	16
6.3.2.2 REGISTER . . . . .	17
6.3.2.3 REGISTER_U8 . . . . .	17
6.3.2.4 REGISTER_U16 . . . . .	18
6.3.2.5 REGISTER_U32 . . . . .	18
6.4 Standard Library . . . . .	18
6.4.1 Detailed Description . . . . .	19

6.5 Standard data types	19
6.5.1 Detailed Description	20
6.5.2 Typedef Documentation	20
6.5.2.1 t_bool	20
6.5.2.2 t_u8	20
6.5.2.3 t_s8	20
6.5.2.4 t_c8	20
6.5.2.5 t_u16	21
6.5.2.6 t_s16	21
6.5.2.7 t_u32	21
6.5.2.8 t_s32	21
6.5.2.9 t_u64	21
6.5.2.10 t_s64	22
6.5.2.11 t_fl32	22
6.5.2.12 t_fl64	22
6.6 Standard values	22
6.6.1 Detailed Description	22
6.6.2 Macro Definition Documentation	23
6.6.2.1 TRUE	23
6.6.2.2 FALSE	23
6.6.2.3 NULL	23
6.7 GPIO Module	23
6.7.1 Detailed Description	24
6.7.2 Macro Definition Documentation	25
6.7.2.1 DEFAULT_PIN_INPUT_TYPE	25
6.7.2.2 DEFAULT_PIN_OUTPUT_TYPE	25
6.7.3 Enumeration Type Documentation	25
6.7.3.1 t_GPIO_Ports	25
6.7.3.2 t_GPIO_Pins	26
6.7.3.3 t_GPIO_Direction	27
6.7.3.4 t_GPIO_Output_Type	27
6.7.3.5 t_GPIO_Input_Type	27
6.7.3.6 t_GPIO_Value	28
6.7.4 Function Documentation	28
6.7.4.1 GPIO_vSetPinDirection()	28
6.7.4.2 GPIO_vSetPinInputType()	29
6.7.4.3 GPIO_vSetPinOutputType()	30
6.7.4.4 GPIO_vSetPinValue()	31
6.7.4.5 GPIO_tGetPinValue()	32
6.8 GPIO Registers	32
6.8.1 Detailed Description	33
6.9 GPIO Addresses	33

6.9.1 Detailed Description	34
6.9.2 Macro Definition Documentation	34
6.9.2.1 BASE_ADDRESS_PORT_A	34
6.9.2.2 GPIO_A	34
6.9.2.3 BASE_ADDRESS_PORT_B	34
6.9.2.4 GPIO_B	34
6.9.2.5 BASE_ADDRESS_PORT_C	35
6.9.2.6 GPIO_C	35
6.9.2.7 BASE_ADDRESS_PORT_D	35
6.9.2.8 GPIO_D	35
6.9.2.9 BASE_ADDRESS_PORT_E	35
6.9.2.10 GPIO_E	36
6.9.2.11 BASE_ADDRESS_PORT_F	36
6.9.2.12 GPIO_F	36
6.9.2.13 BASE_ADDRESS_PORT_G	36
6.9.2.14 GPIO_G	36
6.10 GPIO Pins Constants	37
6.10.1 Detailed Description	37
6.10.2 Macro Definition Documentation	37
6.10.2.1 PIN_SHIFT_VALUE	37
6.10.2.2 PIN_RESET_MASK_VALUE	38
6.10.2.3 PIN_RESET_CONFIGURATION_MASK_VALUE	38
6.10.2.4 PIN_RESET_ODR_MASK_VALUE	38
6.10.2.5 PIN_CONFIGURATION_BITS_SHIFT_VALUE	38
6.10.2.6 PIN_RESET_SHIFT_VALUE	38
6.10.2.7 PIN_RESET_MASK	38
6.10.2.8 PIN_RESET_CONFIGURATIONS_MASK	39
6.10.2.9 PIN_RESET_ODR_MASK	39
6.11 MCAL Peripherals	39
6.11.1 Detailed Description	40
6.12 RCC Configuration	40
6.12.1 Detailed Description	40
6.12.2 Macro Definition Documentation	41
6.12.2.1 RCC_PLL_SRC	41
6.12.2.2 RCC_PLL_MUL	41
6.12.2.3 RCC_PLL_HSE_DIVIDE_BY_2	41
6.12.2.4 RCC_SYSTEM_CLOCK_SOURCE	42
6.12.2.5 RCC_MCO_SOURCE	42
6.12.2.6 RCC_ADC_PRESCALER	42
6.12.2.7 RCC_AHB_PRESCALER	43
6.12.2.8 RCC_APB1_PRESCALER	43
6.12.2.9 RCC_APB2_PRESCALER	43

6.12.2.10 RCC_USB_PRESCALER . . . . .	44
6.12.2.11 RCC_ENABLE_CSS . . . . .	44
6.13 RCC Interface Options . . . . .	44
6.13.1 Detailed Description . . . . .	46
6.13.2 Enumeration Type Documentation . . . . .	46
6.13.2.1 RCC_PLLMulFactors . . . . .	46
6.13.2.2 RCC_AHBPrescaler . . . . .	47
6.13.2.3 RCC_APBPrescaler . . . . .	47
6.13.2.4 RCC_ADCPrescaler . . . . .	48
6.13.2.5 RCC_SystemClock . . . . .	48
6.13.2.6 RCC_PLLSource . . . . .	49
6.13.2.7 RCC_MCOSources . . . . .	49
6.13.2.8 RCC_USBPrescaler . . . . .	50
6.13.2.9 t_RCC_APB2Peripherals . . . . .	50
6.13.2.10 t_RCC_APB1Peripherals . . . . .	51
6.13.2.11 t_RCC_AHBPeripherals . . . . .	52
6.14 RCC exported functions . . . . .	52
6.14.1 Detailed Description . . . . .	53
6.14.2 Function Documentation . . . . .	53
6.14.2.1 RCC_vInit() . . . . .	53
6.14.2.2 RCC_vEnablePeripheralABP2() . . . . .	53
6.14.2.3 RCC_vDisablePeripheralABP2() . . . . .	54
6.14.2.4 RCC_vEnablePeripheralABP1() . . . . .	54
6.14.2.5 RCC_vDisablePeripheralABP1() . . . . .	55
6.14.2.6 RCC_vEnablePeripheralAHB() . . . . .	55
6.14.2.7 RCC_vDisablePeripheralAHB() . . . . .	55
6.15 RCC Module . . . . .	56
6.15.1 Detailed Description . . . . .	56
6.16 RCC Registers . . . . .	56
6.16.1 Detailed Description . . . . .	57
6.17 RCC Addresses . . . . .	57
6.17.1 Detailed Description . . . . .	57
6.17.2 Macro Definition Documentation . . . . .	58
6.17.2.1 RCC_BASE_ADDRESS . . . . .	58
6.17.2.2 RCC . . . . .	58
6.17.2.3 RCC_SET_REGISTER_BIT_STATUS . . . . .	58
6.18 Testing Applications . . . . .	58
6.18.1 Detailed Description . . . . .	59
6.18.2 Function Documentation . . . . .	59
6.18.2.1 vAPPS_main() . . . . .	59
6.18.2.2 vTestApp_TestingGPIO_main() . . . . .	60
6.19 Shared library . . . . .	60

6.19.1 Detailed Description	60
6.19.2 Macro Definition Documentation	61
6.19.2.1 SET_REGISTER_BIT_STATUS	61
<b>7 Data Structure Documentation</b>	<b>63</b>
7.1 t_GPIOx_RegisterMap Struct Reference	63
7.1.1 Detailed Description	63
7.1.2 Field Documentation	63
7.1.2.1 CRL	64
7.1.2.2 CRH	64
7.1.2.3 IDR	64
7.1.2.4 ODR	64
7.1.2.5 BSRR	64
7.1.2.6 BRR	65
7.2 t_RCC_AHBENR Struct Reference	65
7.2.1 Detailed Description	65
7.2.2 Field Documentation	65
7.2.2.1 DMA1EN	65
7.2.2.2 DMA2EN	66
7.2.2.3 __pad0__	66
7.2.2.4 CRCEN	66
7.2.2.5 __pad1__	66
7.3 t_RCC_APB1ENR Struct Reference	66
7.3.1 Detailed Description	68
7.3.2 Field Documentation	68
7.3.2.1 TIM2EN	68
7.3.2.2 TIM3EN	68
7.3.2.3 TIM4EN	68
7.3.2.4 TIM5EN	68
7.3.2.5 TIM6EN	68
7.3.2.6 TIM7EN	69
7.3.2.7 __pad0__	69
7.3.2.8 WWDGEN	69
7.3.2.9 __pad1__	69
7.3.2.10 SPI2EN	69
7.3.2.11 SPI3EN	70
7.3.2.12 __pad2__	70
7.3.2.13 USART2EN	70
7.3.2.14 USART3EN	70
7.3.2.15 UART4EN	70
7.3.2.16 UART5EN	70
7.3.2.17 I2C1EN	71

7.3.2.18 I2C2EN . . . . .	71
7.3.2.19 USBEN . . . . .	71
7.3.2.20 __pad3__ . . . . .	71
7.3.2.21 CANEN . . . . .	71
7.3.2.22 __pad4__ . . . . .	72
7.3.2.23 BKPEN . . . . .	72
7.3.2.24 PWREN . . . . .	72
7.3.2.25 DACEN . . . . .	72
7.3.2.26 __pad5__ . . . . .	72
7.4 t_RCC_APB1RSTR Struct Reference . . . . .	73
7.4.1 Detailed Description . . . . .	74
7.4.2 Field Documentation . . . . .	74
7.4.2.1 TIM2RST . . . . .	74
7.4.2.2 TIM3RST . . . . .	74
7.4.2.3 TIM4RST . . . . .	74
7.4.2.4 TIM5RST . . . . .	75
7.4.2.5 TIM6RST . . . . .	75
7.4.2.6 TIM7RST . . . . .	75
7.4.2.7 __pad0__ . . . . .	75
7.4.2.8 WWDGRST . . . . .	75
7.4.2.9 __pad1__ . . . . .	76
7.4.2.10 SPI2RST . . . . .	76
7.4.2.11 SPI3RST . . . . .	76
7.4.2.12 __pad2__ . . . . .	76
7.4.2.13 USART2RST . . . . .	76
7.4.2.14 USART3RST . . . . .	77
7.4.2.15 UART4RST . . . . .	77
7.4.2.16 UART5RST . . . . .	77
7.4.2.17 I2C1RST . . . . .	77
7.4.2.18 I2C2RST . . . . .	77
7.4.2.19 USBRST . . . . .	77
7.4.2.20 __pad3__ . . . . .	78
7.4.2.21 CANRST . . . . .	78
7.4.2.22 __pad4__ . . . . .	78
7.4.2.23 BKPRST . . . . .	78
7.4.2.24 PWRRST . . . . .	78
7.4.2.25 DACRST . . . . .	79
7.4.2.26 __pad5__ . . . . .	79
7.5 t_RCC_APB2ENR Struct Reference . . . . .	79
7.5.1 Detailed Description . . . . .	80
7.5.2 Field Documentation . . . . .	80
7.5.2.1 AFIOEN . . . . .	80



7.5.2.2 __pad0__ . . . . .	80
7.5.2.3 IOPAEN . . . . .	81
7.5.2.4 IOPBEN . . . . .	81
7.5.2.5 IOPCEN . . . . .	81
7.5.2.6 IOPDEN . . . . .	81
7.5.2.7 IOPEEN . . . . .	81
7.5.2.8 IOPFEN . . . . .	81
7.5.2.9 IOPGEN . . . . .	82
7.5.2.10 ADC1EN . . . . .	82
7.5.2.11 ADC2EN . . . . .	82
7.5.2.12 TIM1EN . . . . .	82
7.5.2.13 SPI1EN . . . . .	82
7.5.2.14 __pad1__ . . . . .	82
7.5.2.15 USART1EN . . . . .	83
7.5.2.16 ADC3EN . . . . .	83
7.5.2.17 __pad2__ . . . . .	83
7.6 t_RCC_APB2RSTR Struct Reference . . . . .	83
7.6.1 Detailed Description . . . . .	84
7.6.2 Field Documentation . . . . .	84
7.6.2.1 AFIORST . . . . .	85
7.6.2.2 __pad0__ . . . . .	85
7.6.2.3 IOPARST . . . . .	85
7.6.2.4 IOPBRST . . . . .	85
7.6.2.5 IOPCRST . . . . .	85
7.6.2.6 IOPDRST . . . . .	85
7.6.2.7 IOPERST . . . . .	86
7.6.2.8 IOPFRST . . . . .	86
7.6.2.9 IOPGRST . . . . .	86
7.6.2.10 ADC1RST . . . . .	86
7.6.2.11 ADC2RST . . . . .	86
7.6.2.12 TIM1RST . . . . .	86
7.6.2.13 SPI1RST . . . . .	87
7.6.2.14 __pad1__ . . . . .	87
7.6.2.15 USART1RST . . . . .	87
7.6.2.16 ADC3RST . . . . .	87
7.6.2.17 __pad2__ . . . . .	87
7.7 t_RCC_BDCR Struct Reference . . . . .	88
7.7.1 Detailed Description . . . . .	88
7.7.2 Field Documentation . . . . .	88
7.7.2.1 LSEON . . . . .	88
7.7.2.2 LSERDY . . . . .	89
7.7.2.3 LSEBYP . . . . .	89

7.7.2.4 __pad0__ . . . . .	89
7.7.2.5 RTCSEL . . . . .	89
7.7.2.6 __pad1__ . . . . .	89
7.7.2.7 RTCEN . . . . .	90
7.7.2.8 BDRST . . . . .	90
7.7.2.9 __pad2__ . . . . .	90
7.8 t_RCC_CFGR Struct Reference . . . . .	90
7.8.1 Detailed Description . . . . .	91
7.8.2 Field Documentation . . . . .	91
7.8.2.1 SW . . . . .	91
7.8.2.2 SWS . . . . .	92
7.8.2.3 HPRE . . . . .	92
7.8.2.4 PPRE1 . . . . .	92
7.8.2.5 PPRE2 . . . . .	92
7.8.2.6 ADCPRE . . . . .	93
7.8.2.7 PLLSRC . . . . .	93
7.8.2.8 PLLXTPRE . . . . .	93
7.8.2.9 PLLMUL . . . . .	93
7.8.2.10 USBPRE . . . . .	93
7.8.2.11 __pad0__ . . . . .	94
7.8.2.12 MCO . . . . .	94
7.8.2.13 __pad1__ . . . . .	94
7.9 t_RCC_CIR Struct Reference . . . . .	94
7.9.1 Detailed Description . . . . .	95
7.9.2 Field Documentation . . . . .	96
7.9.2.1 LSIRDYF . . . . .	96
7.9.2.2 LSERDYF . . . . .	96
7.9.2.3 HSIRDYF . . . . .	96
7.9.2.4 HSERDYF . . . . .	97
7.9.2.5 PLLRDYF . . . . .	97
7.9.2.6 __pad0__ . . . . .	97
7.9.2.7 CSSF . . . . .	97
7.9.2.8 LSIRDYIE . . . . .	98
7.9.2.9 LSERDYIE . . . . .	98
7.9.2.10 HSIRDYIE . . . . .	98
7.9.2.11 HSERDYIE . . . . .	98
7.9.2.12 PLLRDYIE . . . . .	98
7.9.2.13 __pad1__ . . . . .	98
7.9.2.14 LSIRDYC . . . . .	99
7.9.2.15 LSERDYC . . . . .	99
7.9.2.16 HSIRDYC . . . . .	99
7.9.2.17 HSERDYC . . . . .	99

7.9.2.18 PLLRDYC	99
7.9.2.19 __pad2__	99
7.9.2.20 CSSC	100
7.9.2.21 __pad3__	100
7.10 t_RCC_CR Struct Reference	100
7.10.1 Detailed Description	101
7.10.2 Field Documentation	101
7.10.2.1 HSION	101
7.10.2.2 HSIRDY	101
7.10.2.3 __pad0__	101
7.10.2.4 HSITRIM	102
7.10.2.5 HSICAL	102
7.10.2.6 HSEON	102
7.10.2.7 HSERDY	102
7.10.2.8 HSEBYP	103
7.10.2.9 CSSON	103
7.10.2.10 __pad1__	103
7.10.2.11 PLLON	103
7.10.2.12 PLLRDY	103
7.10.2.13 __pad2__	104
7.11 t_RCC_CSR Struct Reference	104
7.11.1 Detailed Description	105
7.11.2 Field Documentation	105
7.11.2.1 LSION	105
7.11.2.2 LSIRDY	105
7.11.2.3 __pad0__	105
7.11.2.4 RMVF	105
7.11.2.5 __pad1__	106
7.11.2.6 PINRSTF	106
7.11.2.7 PORRSTF	106
7.11.2.8 SFTRSTF	106
7.11.2.9 IWDGRSTF	107
7.11.2.10 WWDGRSTF	107
7.11.2.11 LPWRRSTF	107
7.12 t_RCC_RegisterMap Struct Reference	107
7.12.1 Detailed Description	108
7.12.2 Field Documentation	108
7.12.2.1 CR	108
7.12.2.2 CFGR	108
7.12.2.3 CIR	109
7.12.2.4 APB2RSTR	109
7.12.2.5 APB1RSTR	109

7.12.2.6 AHBENR . . . . .	109
7.12.2.7 APB2ENR . . . . .	109
7.12.2.8 APB1ENR . . . . .	109
7.12.2.9 BDCR . . . . .	109
7.12.2.10 CSR . . . . .	109
<b>8 File Documentation</b>	<b>111</b>
8.1 APPS/APPS_main.c File Reference . . . . .	111
8.1.1 Detailed Description . . . . .	111
8.1.2 Macro Definition Documentation . . . . .	112
8.1.2.1 TESTING_APPLICATION_MAIN_FUNC . . . . .	112
8.1.2.2 TestingGPIO . . . . .	112
8.2 APPS/APPS_main.h File Reference . . . . .	112
8.2.1 Detailed Description . . . . .	112
8.3 /github/workspace/README.md File Reference . . . . .	113
8.4 APPS/README.md File Reference . . . . .	113
8.5 APPS/TestingGPIO/TestApp_TestingGPIO_main.c File Reference . . . . .	113
8.5.1 Detailed Description . . . . .	113
8.6 APPS/TestingGPIO/TestApp_TestingGPIO_main.h File Reference . . . . .	113
8.6.1 Detailed Description . . . . .	114
8.7 LIB/LSTD_BITMATH.h File Reference . . . . .	114
8.7.1 Detailed Description . . . . .	114
8.8 LIB/LSTD_COMPILER.h File Reference . . . . .	115
8.8.1 Detailed Description . . . . .	115
8.9 LIB/LSTD_HW_REGS.h File Reference . . . . .	115
8.9.1 Detailed Description . . . . .	116
8.10 LIB/LSTD_SHARED.h File Reference . . . . .	116
8.10.1 Detailed Description . . . . .	117
8.11 LIB/LSTD_TYPES.h File Reference . . . . .	117
8.11.1 Detailed Description . . . . .	118
8.12 LIB/LSTD_VALUES.h File Reference . . . . .	118
8.12.1 Detailed Description . . . . .	118
8.13 MCAL/GPIO/GPIO_config.h File Reference . . . . .	119
8.13.1 Detailed Description . . . . .	119
8.14 MCAL/GPIO/GPIO_interface.h File Reference . . . . .	119
8.14.1 Detailed Description . . . . .	120
8.15 MCAL/GPIO/GPIO_private.h File Reference . . . . .	121
8.15.1 Detailed Description . . . . .	122
8.16 MCAL/GPIO/GPIO_program.c File Reference . . . . .	122
8.16.1 Detailed Description . . . . .	123
8.16.2 Macro Definition Documentation . . . . .	123
8.16.2.1 IS_PIN_IN_LOW_REGISTER . . . . .	123

8.16.3 Function Documentation . . . . .	124
8.16.3.1 GPIO_vGetPortAddress() . . . . .	124
8.16.3.2 GPIO_vGetPinSpan() . . . . .	125
8.16.3.3 GPIO_vSetPinInputTypePullUpDown() . . . . .	125
8.17 MCAL/index.h File Reference . . . . .	126
8.18 MCAL/RCC/RCC_config.h File Reference . . . . .	126
8.18.1 Detailed Description . . . . .	127
8.19 MCAL/RCC/RCC_interface.h File Reference . . . . .	127
8.19.1 Detailed Description . . . . .	129
8.20 MCAL/RCC/RCC_private.h File Reference . . . . .	129
8.21 MCAL/RCC/RCC_program.c File Reference . . . . .	129
8.21.1 Detailed Description . . . . .	130
8.21.2 Function Documentation . . . . .	131
8.21.2.1 RCC_vInitHSEClock() . . . . .	131
8.21.2.2 RCC_vInitHSIClock() . . . . .	131
8.21.2.3 RCC_vInitPLL() . . . . .	132
8.21.2.4 RCC_vInitSystemClock() . . . . .	132
8.21.2.5 RCC_vInitMCO() . . . . .	133
8.21.2.6 RCC_vSetAPB2PeripheralClockStatus() . . . . .	133
8.21.2.7 RCC_vSetAPB1PeripheralClockStatus() . . . . .	134
8.21.2.8 RCC_vSetAHBPeripheralClockStatus() . . . . .	135
8.21.2.9 RCC_vSetADCPreScaler() . . . . .	136
8.21.2.10 RCC_vSetUSBPrescaler() . . . . .	136
8.21.2.11 RCC_vSetAHBPrescaler() . . . . .	137
8.21.2.12 RCC_vSetAPB1Prescaler() . . . . .	137
8.21.2.13 RCC_vSetAPB2Prescaler() . . . . .	137
8.22 main.c File Reference . . . . .	138
8.22.1 Function Documentation . . . . .	138
8.22.1.1 main() . . . . .	138



## Chapter 1

# STM32 Blue Pill Drivers

Drivers that could be used to interface and interact with STM32F103C8T6 Microcontroller

View the PDF documentation [here](#)





## Chapter 2

# Testing applications

These testing applications to test most of the drivers' functionalities and make sure they do what they intend to do.

### 2.1 ## Applications list

Below is the applications list and will follow this template of describing each application.

Name: <Application name>  
Description: <Application description>  
Activision Macro: <Application macro name>

Testing application's directory has the following structure:

- APPS\_main.h
- APPS\_main.c
- <Application name>/
  - TestApp\_<Application name>\_main.h
  - TestApp\_<Application name>\_main.c

### 2.2 ### List

Name: TestingGPIO  
Description: This application tests the GPIO driver.  
Activision Macro: TestingGPIO



## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

Standard Library . . . . .	18
Bit Manipulation Math Macros . . . . .	11
Compiler standard macros . . . . .	13
Hardware registers macro-functions . . . . .	15
Standard data types . . . . .	19
Standard values . . . . .	22
Shared library . . . . .	60
MCAL Peripherals . . . . .	39
GPIO Module . . . . .	23
GPIO Registers . . . . .	32
GPIO Addresses . . . . .	33
GPIO Pins Constants . . . . .	37
RCC Module . . . . .	56
RCC Configuration . . . . .	40
RCC Interface Options . . . . .	44
RCC exported functions . . . . .	52
RCC Registers . . . . .	56
RCC Addresses . . . . .	57
Testing Applications . . . . .	58



## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">t_GPIOx_RegisterMap</a>	
GPIO Register Map . . . . .	63
<a href="#">t_RCC_AHBENR</a>	
RCC AHB peripheral clock register . . . . .	65
<a href="#">t_RCC_APB1ENR</a>	
RCC APB1 peripheral clock enable register . . . . .	66
<a href="#">t_RCC_APB1RSTR</a>	
RCC APB1 peripheral reset register . . . . .	73
<a href="#">t_RCC_APB2ENR</a>	
RCC APB2 peripheral clock enable register . . . . .	79
<a href="#">t_RCC_APB2RSTR</a>	
RCC APB2 peripheral reset register . . . . .	83
<a href="#">t_RCC_BDCR</a>	
RCC Backup domain control register . . . . .	88
<a href="#">t_RCC_CFGR</a>	
RCC clock configuration register . . . . .	90
<a href="#">t_RCC_CIR</a>	
RCC clock interrupt register . . . . .	94
<a href="#">t_RCC_CR</a>	
RCC clock control register . . . . .	100
<a href="#">t_RCC_CSR</a>	
RCC Control/status register . . . . .	104
<a href="#">t_RCC_RegisterMap</a>	
RCC Register Map . . . . .	107



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">main.c</a> . . . . .	138
<a href="#">APPS/APPS_main.c</a> This file contains the implementation of the main function that is responsible for running the applications . . . . .	111
<a href="#">APPS/APPS_main.h</a> This file contains the prototypes of the main function of the testing applications . . . . .	112
<a href="#">APPS/TestingGPIO/TestApp_TestingGPIO_main.c</a> This file contains the main implementation for the TestApp_TestingGPIO application . . . . .	113
<a href="#">APPS/TestingGPIO/TestApp_TestingGPIO_main.h</a> This file contains the main header file for the TestApp_TestingGPIO application . . . . .	113
<a href="#">LIB/LSTD_BITMATH.h</a> This file contains the bit math manipulation macro-functions . . . . .	114
<a href="#">LIB/LSTD_COMPILER.h</a> This file contains the compiler standard macros . . . . .	115
<a href="#">LIB/LSTD_HW_REGS.h</a> This file contains the hardware registers macro-functions for memory addresses mapping and accessing . . . . .	115
<a href="#">LIB/LSTD_SHARED.h</a> Shared library for all the standard types . . . . .	116
<a href="#">LIB/LSTD_TYPES.h</a> This file contains the standard data types . . . . .	117
<a href="#">LIB/LSTD_VALUES.h</a> This file contains the standard values . . . . .	118
<a href="#">MCAL/index.h</a> . . . . .	126
<a href="#">MCAL/GPIO/GPIO_config.h</a> This file contains the configuration parameters for the GPIO module . . . . .	119
<a href="#">MCAL/GPIO/GPIO_interface.h</a> This file is the interface file for the GPIO module . . . . .	119
<a href="#">MCAL/GPIO/GPIO_private.h</a> This file contains the private macros and registers for the GPIO module . . . . .	121
<a href="#">MCAL/GPIO/GPIO_program.c</a> This file contains the implementation for the GPIO module . . . . .	122
<a href="#">MCAL/RCC/RCC_config.h</a> This file contains the configuration parameters for the RCC module . . . . .	126
<a href="#">MCAL/RCC/RCC_interface.h</a> This file is the interface file for the RCC module . . . . .	127

<a href="#">MCAL/RCC/RCC_private.h</a>	
This file contains the private macros and registers for the RCC module . . . . .	129
<a href="#">MCAL/RCC/RCC_program.c</a>	
This file contains the implementation for the RCC module . . . . .	129



## Chapter 6

# Module Documentation

### 6.1 Bit Manipulation Math Macros

This module contains the bit math manipulation macro-functions.

Collaboration diagram for Bit Manipulation Math Macros:

#### Macros

- `#define SET_BIT(REG, BITNUM) (REG) |= (1 << (BITNUM))`  
*Set a certain bit's value.*
- `#define CLEAR_BIT(REG, BITNUM) (REG) &= ~(1 << (BITNUM))`  
*Clear a certain bit's value to.*
- `#define TOGGLE_BIT(REG, BITNUM) (REG) ^= (1 << (BITNUM))`  
*Toggle a bit to 0 if it's 1, 1 otherwise.*
- `#define GET_BIT(REG, BITNUM) (((REG) >> (BITNUM)) & 1)`  
*Return the value of the bit whether it's 1 or 0*

#### 6.1.1 Detailed Description

This module contains the bit math manipulation macro-functions.

#### 6.1.2 Macro Definition Documentation

##### 6.1.2.1 SET\_BIT

```
#define SET_BIT(  
    REG,  
    BITNUM ) (REG) |= (1 << (BITNUM))
```

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

Set a certain bit's value.

**Parameters**

in	<i>REG</i>	The register to set its bit
in	<i>BITNUM</i>	The bit number to set

**6.1.2.2 CLEAR\_BIT**

```
#define CLEAR_BIT(  
    REG,  
    BITNUM ) (REG) &= ~(1 << (BITNUM))  
  
#include <LIB/LSTD_BITMATH.h>
```

Clear a certain bit's value to.

**Parameters**

in	<i>REG</i>	The register to clear its bit
in	<i>BITNUM</i>	The bit number to clear

**6.1.2.3 TOGGLE\_BIT**

```
#define TOGGLE_BIT(  
    REG,  
    BITNUM ) (REG) ^= (1 << (BITNUM))  
  
#include <LIB/LSTD_BITMATH.h>
```

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

**Parameters**

in	<i>REG</i>	The register to toggle its bit
in	<i>BITNUM</i>	The bit number to toggle

**6.1.2.4 GET\_BIT**

```
#define GET_BIT(  
    REG,  
    BITNUM ) (((REG) >> (BITNUM)) & 1)  
  
#include <LIB/LSTD_BITMATH.h>
```

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

## Parameters

in	<i>REG</i>	The register to get its bit
in	<i>BITNUM</i>	The bit number to get

## 6.2 Compiler standard macros

This file contains the compiler standard macros.

Collaboration diagram for Compiler standard macros:

### Macros

- `#define CONST const`  
*Declare a standard constant variable with the specified type.*
- `#define STATIC static`  
*Declare a standard static variable/function.*
- `#define VOLATILE volatile`  
*Declare a standard volatile variable.*
- `#define P2VAR(ptrtype) ptrtype *`  
*Declare a pointer-to-variable with the specified type.*
- `#define P2CONST(ptrtype) CONST ptrtype *`  
*Declare a constant pointer-to-variable with the specified type.*
- `#define CONSTP2VAR(ptrtype) ptrtype *CONST`  
*Declare a pointer-to-variable constant with the specified type.*
- `#define CONSTP2CONST(ptrtype) CONST ptrtype *CONST`  
*Declare a constant pointer-to-variable constant with the specified type.*
- `#define P2FUNC(rettype, fctname) rettype(*fctname)`  
*Declare a pointer-to-function with the specified return type.*

### 6.2.1 Detailed Description

This file contains the compiler standard macros.

### 6.2.2 Macro Definition Documentation

#### 6.2.2.1 **CONST**

```
#define CONST const
```

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

Declare a standard constant variable with the specified type.

### 6.2.2.2 STATIC

```
#define STATIC static
```

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

Declare a standard static variable/function.

### 6.2.2.3 VOLATILE

```
#define VOLATILE volatile
```

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

Declare a standard volatile variable.

### 6.2.2.4 P2VAR

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

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

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

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 6.2.2.5 P2CONST

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

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

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

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 6.2.2.6 CONSTP2VAR

```
#define CONSTP2VAR(  
    ptrtype ) ptrtype *CONST  
  
#include <LIB/LSTD_COMPILER.h>
```

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

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 6.2.2.7 CONSTP2CONST

```
#define CONSTP2CONST(  
    ptrtype ) CONST ptrtype *CONST  
  
#include <LIB/LSTD_COMPILER.h>
```

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

#### Parameters

in	<i>ptrtype</i>	The type of the pointer
----	----------------	-------------------------

### 6.2.2.8 P2FUNC

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

Declare a pointer-to-function with the specified return type.

#### Parameters

in	<i>rettype</i>	The return type of the function
in	<i>fctname</i>	The name of the function

## 6.3 Hardware registers macro-functions

These macro-functions help in mapping and accessing the hardware registers.

Collaboration diagram for Hardware registers macro-functions:

## Macros

- `#define REGISTER_ADDRESS(ADDRESS, OFFSET) ((ADDRESS) + (OFFSET))`  
*Placeholder for declaring a register address.*
- `#define REGISTER(REG_TYPE, ADDRESS) (*(VOLATILE P2VAR(REG_TYPE))(ADDRESS))`  
*Map to a certain register by its address in the memory.*
- `#define REGISTER_U8(ADDRESS) REGISTER(t_u8, ADDRESS)`  
*Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)*
- `#define REGISTER_U16(ADDRESS) REGISTER(t_u16, ADDRESS)`  
*Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)*
- `#define REGISTER_U32(ADDRESS) REGISTER(t_u32, ADDRESS)`  
*Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)*

### 6.3.1 Detailed Description

These macro-functions help in mapping and accessing the hardware registers.

### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 REGISTER\_ADDRESS

```
#define REGISTER_ADDRESS(  
    ADDRESS,  
    OFFSET ) ((ADDRESS) + (OFFSET))
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Placeholder for declaring a register address.

#### Parameters

in	<i>ADDRESS</i>	The address of the register
in	<i>OFFSET</i>	The offset of the register

#### Returns

The final address of the register

### 6.3.2.2 REGISTER

```
#define REGISTER(  
    REG_TYPE,  
    ADDRESS ) (*(VOLATILE P2VAR(REG_TYPE)) (ADDRESS))
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its address in the memory.

#### Parameters

in	<i>ADDRESS</i>	The address of the register
in	<i>REG_TYPE</i>	The type of the register

#### Note

REG\_TYPE can be a standard type (e.g. t\_u8, t\_u16, t\_u32, ...) or a user-defined type

#### Returns

The value of the register

#### See also

[Standard data types](#)

### 6.3.2.3 REGISTER\_U8

```
#define REGISTER_U8(  
    ADDRESS ) REGISTER(t_u8, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)

#### Parameters

in	<i>ADDRESS</i>	The address of the register
----	----------------	-----------------------------

#### Returns

The value of the register

#### 6.3.2.4 REGISTER\_U16

```
#define REGISTER_U16(  
    ADDRESS ) REGISTER(t_u16, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)

##### Parameters

in	ADDRESS	The address of the register
----	---------	-----------------------------

##### Returns

The value of the register

#### 6.3.2.5 REGISTER\_U32

```
#define REGISTER_U32(  
    ADDRESS ) REGISTER(t_u32, ADDRESS)
```

```
#include <LIB/LSTD_HW_REGS.h>
```

Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)

##### Parameters

in	ADDRESS	The address of the register
----	---------	-----------------------------

##### Returns

The value of the register

## 6.4 Standard Library

This module contains the standard library-related macros, types, and functions.

Collaboration diagram for Standard Library:

### Modules

- [Bit Manipulation Math Macros](#)

*This module contains the bit math manipulation macro-functions.*

- [Compiler standard macros](#)



*This file contains the compiler standard macros.*

- [Hardware registers macro-functions](#)

*These macro-functions help in mapping and accessing the hardware registers.*

- [Standard data types](#)

*This module contains the standard data types.*

- [Standard values](#)

*This module contains the standard values.*

- [Shared library](#)

*This library contains the shared macros and functions for all the standard types.*

### 6.4.1 Detailed Description

This module contains the standard library-related macros, types, and functions.

## 6.5 Standard data types

This module contains the standard data types.

Collaboration diagram for Standard data types:

### Typedefs

- typedef unsigned char [t\\_bool](#)

*Type definition for boolean.*

- typedef unsigned char [t\\_u8](#)

*Type definition for 8-bit unsigned INT.*

- typedef signed char [t\\_s8](#)

*Type definition for 8-bit signed INT.*

- typedef unsigned char [t\\_c8](#)

*Type definition for 8-bit char.*

- typedef unsigned short int [t\\_u16](#)

*Type definition for 16-bit unsigned int.*

- typedef signed short int [t\\_s16](#)

*Type definition for 16-bit signed INT.*

- typedef unsigned int [t\\_u32](#)

*Type definition for 32-bit unsigned int.*

- typedef signed int [t\\_s32](#)

*Type definition for 32-bit signed INT.*

- typedef unsigned long int [t\\_u64](#)

*Type definition for 64-bit unsigned int.*

- typedef signed long int [t\\_s64](#)

*Type definition for 64-bit signed int.*

- typedef float [t\\_f32](#)

*Type definition for 32-bit float.*

- typedef double [t\\_f64](#)

*Type definition for 64-bit float.*

## 6.5.1 Detailed Description

This module contains the standard data types.

## 6.5.2 Typedef Documentation

### 6.5.2.1 t\_bool

t\_bool

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

Type definition for boolean.

### 6.5.2.2 t\_u8

t\_u8

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

Type definition for 8-bit unsigned INT.

### 6.5.2.3 t\_s8

t\_s8

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

Type definition for 8-bit signed INT.

### 6.5.2.4 t\_c8

t\_c8

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

Type definition for 8-bit char.

#### 6.5.2.5 t\_u16

t\_u16

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

Type definition for 16-bit unsigned int.

#### 6.5.2.6 t\_s16

t\_s16

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

Type definition for 16-bit signed INT.

#### 6.5.2.7 t\_u32

t\_u32

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

Type definition for 32-bit unsigned int.

#### 6.5.2.8 t\_s32

t\_s32

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

Type definition for 32-bit signed INT.

#### 6.5.2.9 t\_u64

t\_u64

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

Type definition for 64-bit unsigned int.

#### 6.5.2.10 t\_s64

t\_s64

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

Type definition for 64-bit signed int.

#### 6.5.2.11 t\_fl32

t\_fl32

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

Type definition for 32-bit float.

#### 6.5.2.12 t\_fl64

t\_fl64

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

Type definition for 64-bit float.

## 6.6 Standard values

This module contains the standard values.

Collaboration diagram for Standard values:

### Macros

- #define TRUE ((t\_bool)1)  
*Type definition for TRUE.*
- #define FALSE ((t\_bool)0)  
*Type definition for FALSE.*
- #define NULL ((P2VAR(void))0)  
*Type definition for NULL.*

### 6.6.1 Detailed Description

This module contains the standard values.

## 6.6.2 Macro Definition Documentation

### 6.6.2.1 TRUE

```
#define TRUE ((t_bool)1)
```

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

Type definition for TRUE.

### 6.6.2.2 FALSE

```
#define FALSE ((t_bool)0)
```

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

Type definition for FALSE.

### 6.6.2.3 NULL

```
#define NULL ((P2VAR(void))0)
```

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

Type definition for NULL.

## 6.7 GPIO Module

GPIO Module.

Collaboration diagram for GPIO Module:

### Modules

- [GPIO Registers](#)  
*GPIO Registers.*
- [GPIO Addresses](#)  
*GPIO Addresses.*
- [GPIO Pins Constants](#)  
*GPIO Pins Constants.*

## Macros

- #define `DEFAULT_PIN_INPUT_TYPE` (`GPIO_Input_Type_Pull_Down`)  
*The default input type for the GPIO pins.*
- #define `DEFAULT_PIN_OUTPUT_TYPE` (`GPIO_Output_Type_Push_Pull`)  
*The default output type for the GPIO pins.*

## Enumerations

- enum `t_GPIO_Ports` {  
  `GPIO_Ports_A` = 0 , `GPIO_Ports_B` , `GPIO_Ports_C` , `GPIO_Ports_D` ,  
  `GPIO_Ports_E` , `GPIO_Ports_F` , `GPIO_Ports_G` }  
*GPIO Ports.*
- enum `t_GPIO_Pins` {  
  `GPIO_Pins_0` = 0 , `GPIO_Pins_1` , `GPIO_Pins_2` , `GPIO_Pins_3` ,  
  `GPIO_Pins_4` , `GPIO_Pins_5` , `GPIO_Pins_6` , `GPIO_Pins_7` ,  
  `GPIO_Pins_8` , `GPIO_Pins_9` , `GPIO_Pins_10` , `GPIO_Pins_11` ,  
  `GPIO_Pins_12` , `GPIO_Pins_13` , `GPIO_Pins_14` , `GPIO_Pins_15` }  
*GPIO Pins.*
- enum `t_GPIO_Direction` { `GPIO_Direction_Input` = 0 , `GPIO_Direction_Output_10MHz` , `GPIO_Direction_Output_2MHz` ,  
  `GPIO_Direction_Output_50MHz` }  
*GPIO Direction.*
- enum `t_GPIO_Output_Type` { `GPIO_Output_Type_Push_Pull` = 0 , `GPIO_Output_Type_Open_Drain` ,  
  `GPIO_Output_Type_Alternate_Push_Pull` , `GPIO_Output_Type_Alternate_Open_Drain` }  
*GPIO Output Type.*
- enum `t_GPIO_Input_Type` { `GPIO_Input_Type_Analog` = 0 , `GPIO_Input_Type_Floating` , `GPIO_Input_Type_Pull_Down` ,  
  `GPIO_Input_Type_Pull_Up` }  
*GPIO Input Type.*
- enum `t_GPIO_Value` { `GPIO_Value_Low` = 0 , `GPIO_Value_High` }  
*GPIO Pin Value.*

## Functions

- void `GPIO_vSetPinDirection` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Direction` tDirection)  
*This function is used to set the direction of a GPIO pin.*
- void `GPIO_vSetPinInputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Input_Type` tInputType)  
*This function is used to set the input type of a GPIO pin.*
- void `GPIO_vSetPinOutputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Output_Type` tOutputType)  
*This function is used to set the output type of a GPIO pin.*
- void `GPIO_vSetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Value` tValue)  
*This function is used to set the value of a GPIO pin.*
- `t_GPIO_Value` `GPIO_tGetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin)  
*This function is used to get the value of a GPIO pin.*

### 6.7.1 Detailed Description

GPIO Module.

This module contains all the APIs related to the GPIO module

## 6.7.2 Macro Definition Documentation

### 6.7.2.1 DEFAULT\_PIN\_INPUT\_TYPE

```
#define DEFAULT_PIN_INPUT_TYPE (GPIO_Input_Type_Pull_Down)
```

```
#include <MCAL/GPIO/GPIO_config.h>
```

The default input type for the GPIO pins.

This macro defines the default input type for the GPIO pins

### 6.7.2.2 DEFAULT\_PIN\_OUTPUT\_TYPE

```
#define DEFAULT_PIN_OUTPUT_TYPE (GPIO_Output_Type_Push_Pull)
```

```
#include <MCAL/GPIO/GPIO_config.h>
```

The default output type for the GPIO pins.

This macro defines the default output type for the GPIO pins

## 6.7.3 Enumeration Type Documentation

### 6.7.3.1 t\_GPIO\_Ports

```
enum t_GPIO_Ports
```

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

GPIO Ports.

This type is used to select a GPIO port

Enumerator

GPIO_Ports_A	GPIO Port A.
GPIO_Ports_B	GPIO Port B.
GPIO_Ports_C	GPIO Port C.
GPIO_Ports_D	GPIO Port D.
GPIO_Ports_E	GPIO Port E.
GPIO_Ports_F	GPIO Port F.
GPIO_Ports_G	GPIO Port G.

```

29 {
33     GPIO_Ports_A = 0,
37     GPIO_Ports_B,
41     GPIO_Ports_C,
45     GPIO_Ports_D,
49     GPIO_Ports_E,
53     GPIO_Ports_F,
57     GPIO_Ports_G
58 } t_GPIO_Ports;

```

### 6.7.3.2 t\_GPIO\_Pins

```
enum t_GPIO_Pins
```

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

GPIO Pins.

This type is used to select a GPIO pin

Enumerator

GPIO_Pins_0	GPIO Pin 0.
GPIO_Pins_1	GPIO Pin 1.
GPIO_Pins_2	GPIO Pin 2.
GPIO_Pins_3	GPIO Pin 3.
GPIO_Pins_4	GPIO Pin 4.
GPIO_Pins_5	GPIO Pin 5.
GPIO_Pins_6	GPIO Pin 6.
GPIO_Pins_7	GPIO Pin 7.
GPIO_Pins_8	GPIO Pin 8.
GPIO_Pins_9	GPIO Pin 9.
GPIO_Pins_10	GPIO Pin 10.
GPIO_Pins_11	GPIO Pin 11.
GPIO_Pins_12	GPIO Pin 12.
GPIO_Pins_13	GPIO Pin 13.
GPIO_Pins_14	GPIO Pin 14.
GPIO_Pins_15	GPIO Pin 15.

```

66 {
70     GPIO_Pins_0 = 0,
74     GPIO_Pins_1,
78     GPIO_Pins_2,
82     GPIO_Pins_3,
86     GPIO_Pins_4,
90     GPIO_Pins_5,
94     GPIO_Pins_6,
98     GPIO_Pins_7,
102    GPIO_Pins_8,
106    GPIO_Pins_9,
110    GPIO_Pins_10,
114    GPIO_Pins_11,
118    GPIO_Pins_12,
122    GPIO_Pins_13,
126    GPIO_Pins_14,
130    GPIO_Pins_15
131 } t_GPIO_Pins;

```



### 6.7.3.3 t\_GPIO\_Direction

```
enum t_GPIO_Direction
```

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

GPIO Direction.

This type is used to select a GPIO pin direction

#### Enumerator

GPIO_Direction_Input	GPIO Pin Input.
GPIO_Direction_Output_10MHz	GPIO Pin Output with max speed of 10 MHz.
GPIO_Direction_Output_2MHz	GPIO Pin Output with max speed of 2 MHz.
GPIO_Direction_Output_50MHz	GPIO Pin Output with max speed of 50 MHz.

```
139 {
143     GPIO_Direction_Input = 0,
147     GPIO_Direction_Output_10MHz,
151     GPIO_Direction_Output_2MHz,
155     GPIO_Direction_Output_50MHz
156 } t_GPIO_Direction;
```

### 6.7.3.4 t\_GPIO\_Output\_Type

```
enum t_GPIO_Output_Type
```

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

GPIO Output Type.

This type is used to select a GPIO pin output type

#### Enumerator

GPIO_Output_Type_Push_Pull	GPIO Pin Output Type: push-pull.
GPIO_Output_Type_Open_Drain	GPIO Pin Output Type: open-drain.
GPIO_Output_Type_Alternate_Push_Pull	GPIO Pin Output Type: alternate push-pull.
GPIO_Output_Type_Alternate_Open_Drain	GPIO Pin Output Type: alternate open-drain.

```
164 {
168     GPIO_Output_Type_Push_Pull = 0,
172     GPIO_Output_Type_Open_Drain,
176     GPIO_Output_Type_Alternate_Push_Pull,
180     GPIO_Output_Type_Alternate_Open_Drain
181 } t_GPIO_Output_Type;
```

### 6.7.3.5 t\_GPIO\_Input\_Type

```
enum t_GPIO_Input_Type
```

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

GPIO Input Type.

This type is used to select a GPIO pin input type

#### Enumerator

GPIO_Input_Type_Analog	GPIO Pin Input Type: Analog.
GPIO_Input_Type_Floating	GPIO Pin Input Type: Floating.
GPIO_Input_Type_Pull_Down	GPIO Pin Input Type: Pull-Down.
GPIO_Input_Type_Pull_Up	GPIO Pin Input Type: Pull-Up.

```
189 {
193     GPIO_Input_Type_Analog = 0,
197     GPIO_Input_Type_Floating,
201     GPIO_Input_Type_Pull_Down,
205     GPIO_Input_Type_Pull_Up
206 } t_GPIO_Input_Type;
```

#### 6.7.3.6 t\_GPIO\_Value

```
enum t_GPIO_Value
```

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

GPIO Pin Value.

This type is used to select a GPIO pin value

#### Enumerator

GPIO_Value_Low	GPIO Pin Value: LOW (0)
GPIO_Value_High	GPIO Pin Value: HIGH (1)

```
214 {
218     GPIO_Value_Low = 0,
222     GPIO_Value_High
223 } t_GPIO_Value;
```

### 6.7.4 Function Documentation

#### 6.7.4.1 GPIO\_vSetPinDirection()

```
void GPIO_vSetPinDirection (
    t_GPIO_Ports tPort,
    t_GPIO_Pins tPin,
    t_GPIO_Direction tDirection )

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

This function is used to set the direction of a GPIO pin.

This function is used to set the direction of a GPIO pin

## Parameters

in	<i>tPort</i>	The GPIO port
in	<i>tPin</i>	The GPIO pin
in	<i>tDirection</i>	The GPIO pin direction

## See also

[t\\_GPIO\\_Ports](#) [t\\_GPIO\\_Pins](#) [t\\_GPIO\\_Direction](#)

```

104 {
105     /* Get the pin span (the number of bits to shift to reach the target pin mode and configuration
106        bits) */
107     t_u8 u8PinSpan = GPIO_vGetPinSpan(tPin);
108     /* Store the base address of the GPIO port */
109     P2VAR(t_GPIOx_RegisterMap)
110     pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))NULL;
111     /* Store the pin mode and configuration bits of the pin */
112     P2VAR(t_u32)
113     pu32TargetPinModeConfig = (P2VAR(t_u32))NULL;
114     /* Get the base address of the GPIO port */
115     GPIO_vGetPortAddress(tPort, &pu32PortBaseAddress);
116     /* Get the pin mode and configuration bits of the pin */
117     pu32TargetPinModeConfig = IS_PIN_IN_LOW_REGISTER(tPin) ? &pu32PortBaseAddress->CRL :
118     &pu32PortBaseAddress->CRH;
119     /* Set the mode and configuration bits of the pin */
120     *pu32TargetPinModeConfig = (*pu32TargetPinModeConfig & PIN_RESET_MASK(u8PinSpan)) | (tDirection <
121     u8PinSpan);
122     /* Set the pull-up/pull-down resistor */
123     if (tDirection == GPIO_Direction_Input)
124     {
125         GPIO_vSetPinInputType(tPort, tPin, DEFAULT_PIN_INPUT_TYPE);
126     }
127     else
128     {
129         GPIO_vSetPinOutputType(tPort, tPin, DEFAULT_PIN_OUTPUT_TYPE);
130     }
131 }

```

References [DEFAULT\\_PIN\\_INPUT\\_TYPE](#), [DEFAULT\\_PIN\\_OUTPUT\\_TYPE](#), [GPIO\\_Direction\\_Input](#), [GPIO\\_vGetPinSpan\(\)](#), [GPIO\\_vGetPortAddress\(\)](#), [GPIO\\_vSetPinInputType\(\)](#), [GPIO\\_vSetPinOutputType\(\)](#), [IS\\_PIN\\_IN\\_LOW\\_REGISTER](#), [NULL](#), [P2VAR](#), and [PIN\\_RESET\\_MASK](#).

Referenced by [vTestApp\\_TestingGPIO\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

## 6.7.4.2 GPIO\_vSetPinInputType()

```

void GPIO_vSetPinInputType (
    t_GPIO_Ports tPort,
    t_GPIO_Pins tPin,
    t_GPIO_Input_Type tInputType )

```

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

This function is used to set the input type of a GPIO pin.

This function is used to set the input type of a GPIO pin

## Parameters

in	<i>tPort</i>	The GPIO port
in	<i>tPin</i>	The GPIO pin
in	<i>tInputType</i>	The GPIO pin input type

See also

[t\\_GPIO\\_Ports](#) [t\\_GPIO\\_Pins](#) [t\\_GPIO\\_Input\\_Type](#)

```

133 {
134     /* Get the pin span (the number of bits to shift to reach the target pin mode and configuration
        bits) */
135     t_u8 u8PinSpan = GPIO_vGetPinSpan(tPin);
136     /* Check if the input type is pull-up or pull-down */
137     t_GPIO_Input_Type u8PinInputTypeModeConfigurations = (tInputType == GPIO_Input_Type_Pull_Up) ?
        GPIO_Input_Type_Pull_Down : tInputType;
138     /* Store the base address of the GPIO port */
139     P2VAR(t_GPIOx_RegisterMap)
140     pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))NULL;
141     /* Store the pin mode and configuration bits of the pin */
142     P2VAR(t_u32)
143     pu32TargetPinModeConfig = (P2VAR(t_u32))NULL;
144
145     /* Get the base address of the GPIO port */
146     GPIO_vGetPortAddress(tPort, &pu32PortBaseAddress);
147     /* Get the pin mode and configuration bits of the pin */
148     pu32TargetPinModeConfig = IS_PIN_IN_LOW_REGISTER(tPin) ? &pu32PortBaseAddress->CRL :
        &pu32PortBaseAddress->CRH;
149     /* Set the mode and configuration bits of the pin */
150     *pu32TargetPinModeConfig = (*pu32TargetPinModeConfig & PIN_RESET_CONFIGURATIONS_MASK(u8PinSpan)) |
        (u8PinInputTypeModeConfigurations << (u8PinSpan + PIN_CONFIGURATION_BITS_SHIFT_VALUE));
151     /* Set the pull-up/pull-down resistor */
152     GPIO_vSetPinInputTypePullUpDown(pu32PortBaseAddress, tPin, tInputType);
153 }

```

References [GPIO\\_Input\\_Type\\_Pull\\_Down](#), [GPIO\\_Input\\_Type\\_Pull\\_Up](#), [GPIO\\_vGetPinSpan\(\)](#), [GPIO\\_vGetPortAddress\(\)](#), [GPIO\\_vSetPinInputTypePullUpDown\(\)](#), [IS\\_PIN\\_IN\\_LOW\\_REGISTER](#), [NULL](#), [P2VAR](#), [PIN\\_CONFIGURATION\\_BITS\\_SHIFT\\_VALUE](#) and [PIN\\_RESET\\_CONFIGURATIONS\\_MASK](#).

Referenced by [GPIO\\_vSetPinDirection\(\)](#), and [vTestApp\\_TestingGPIO\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

#### 6.7.4.3 GPIO\_vSetPinOutputType()

```

void GPIO_vSetPinOutputType (
    t_GPIO_Ports tPort,
    t_GPIO_Pins tPin,
    t_GPIO_Output_Type tOutputType )

```

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

This function is used to set the output type of a GPIO pin.

This function is used to set the output type of a GPIO pin

##### Parameters

in	<i>tPort</i>	The GPIO port
in	<i>tPin</i>	The GPIO pin
in	<i>tOutputType</i>	The GPIO pin output type

See also

[t\\_GPIO\\_Ports](#) [t\\_GPIO\\_Pins](#) [t\\_GPIO\\_Output\\_Type](#)

```

156 {
157     /* Get the pin span (the number of bits to shift to reach the target pin mode and configuration
        bits) */
158     t_u8 u8PinSpan = GPIO_vGetPinSpan(tPin);

```

```

159  /* Store the base address of the GPIO port */
160  P2VAR(t_GPIOx_RegisterMap)
161  pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))NULL;
162  /* Store the pin mode and configuration bits of the pin */
163  P2VAR(t_u32)
164  pu32TargetPinModeConfig = (P2VAR(t_u32))NULL;
165
166  /* Get the base address of the GPIO port */
167  GPIO_vGetPortAddress(tPort, &pu32PortBaseAddress);
168  /* Get the pin mode and configuration bits of the pin */
169  pu32TargetPinModeConfig = IS_PIN_IN_LOW_REGISTER(tPin) ? &pu32PortBaseAddress->CRL :
    &pu32PortBaseAddress->CRH;
170  /* Set the mode and configuration bits of the pin */
171  *pu32TargetPinModeConfig = (*pu32TargetPinModeConfig & PIN_RESET_CONFIGURATIONS_MASK(u8PinSpan)) |
    (t_u32)(tOutputType << (u8PinSpan + PIN_CONFIGURATION_BITS_SHIFT_VALUE));
172 }

```

References [GPIO\\_vGetPinSpan\(\)](#), [GPIO\\_vGetPortAddress\(\)](#), [IS\\_PIN\\_IN\\_LOW\\_REGISTER](#), [NULL](#), [P2VAR](#), [PIN\\_CONFIGURATION\\_BITS\\_SHIFT\\_VALUE](#), and [PIN\\_RESET\\_CONFIGURATIONS\\_MASK](#).

Referenced by [GPIO\\_vSetPinDirection\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

#### 6.7.4.4 GPIO\_vSetPinValue()

```

void GPIO_vSetPinValue (
    t_GPIO_Ports tPort,
    t_GPIO_Pins tPin,
    t_GPIO_Value tValue )

```

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

This function is used to set the value of a GPIO pin.

This function is used to set the value of a GPIO pin

##### Parameters

in	<i>tPort</i>	The GPIO port
in	<i>tPin</i>	The GPIO pin
in	<i>tValue</i>	The GPIO pin value

##### See also

[t\\_GPIO\\_Ports](#) [t\\_GPIO\\_Pins](#) [t\\_GPIO\\_Value](#)

```

175 {
176  /* Get the pin location (the number of bits to shift to reach the target pin) */
177  t_u32 u32PinLocation = tPin;
178  /* Store the base address of the GPIO port */
179  P2VAR(t_GPIOx_RegisterMap)
180  pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))NULL;
181
182  /* Get the base address of the GPIO port */
183  GPIO_vGetPortAddress(tPort, &pu32PortBaseAddress);
184
185  if (tValue == GPIO_Value_Low)
186  {
187      u32PinLocation += PIN_RESET_SHIFT_VALUE;
188  }
189  else
190  {
191      /* Do nothing */
192  }
193 }

```

```

194     /* Set the value of the pin */
195     pu32PortBaseAddress->BSRR = (t_u32) (TRUE << u32PinLocation);
196 }

```

References [GPIO\\_Value\\_Low](#), [GPIO\\_vGetPortAddress\(\)](#), [NULL](#), [P2VAR](#), [PIN\\_RESET\\_SHIFT\\_VALUE](#), and [TRUE](#).

Referenced by [vTestApp\\_TestingGPIO\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

#### 6.7.4.5 GPIO\_tGetPinValue()

```

t_GPIO_Value GPIO_tGetPinValue (
    t_GPIO_Ports tPort,
    t_GPIO_Pins tPin )

```

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

This function is used to get the value of a GPIO pin.

This function is used to get the value of a GPIO pin

##### Parameters

in	<i>tPort</i>	The GPIO port
in	<i>tPin</i>	The GPIO pin

##### Returns

The GPIO pin value

##### See also

[t\\_GPIO\\_Ports](#) [t\\_GPIO\\_Pins](#) [t\\_GPIO\\_Value](#)

```

199 {
200     /* Store the base address of the GPIO port */
201     P2VAR(t_GPIOx_RegisterMap)
202     pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap)) NULL;
203
204     /* Get the base address of the GPIO port */
205     GPIO_vGetPortAddress(tPort, &pu32PortBaseAddress);
206
207     /* Get the value of the pin */
208     return (t_GPIO_Value) ((pu32PortBaseAddress->IDR >> tPin) & TRUE);
209 }

```

References [GPIO\\_vGetPortAddress\(\)](#), [NULL](#), [P2VAR](#), and [TRUE](#).

Referenced by [vTestApp\\_TestingGPIO\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

## 6.8 GPIO Registers

GPIO Registers.

Collaboration diagram for GPIO Registers:

## Data Structures

- struct [t\\_GPIOx\\_RegisterMap](#)  
*GPIO Register Map.*

### 6.8.1 Detailed Description

GPIO Registers.

## 6.9 GPIO Addresses

GPIO Addresses.

Collaboration diagram for GPIO Addresses:

## Macros

- #define [BASE\\_ADDRESS\\_PORT\\_A REGISTER\\_ADDRESS](#)(0x40010800, 0)  
*Base Address of Port A.*
- #define [GPIO\\_A REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_A](#))  
*GPIO Port A.*
- #define [BASE\\_ADDRESS\\_PORT\\_B REGISTER\\_ADDRESS](#)(0x40010C00, 0)  
*Base Address of Port B.*
- #define [GPIO\\_B REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_B](#))  
*GPIO Port B.*
- #define [BASE\\_ADDRESS\\_PORT\\_C REGISTER\\_ADDRESS](#)(0x40011000, 0)  
*Base Address of Port C.*
- #define [GPIO\\_C REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_C](#))  
*GPIO Port C.*
- #define [BASE\\_ADDRESS\\_PORT\\_D REGISTER\\_ADDRESS](#)(0x40011400, 0)  
*Base Address of Port D.*
- #define [GPIO\\_D REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_D](#))  
*GPIO Port D.*
- #define [BASE\\_ADDRESS\\_PORT\\_E REGISTER\\_ADDRESS](#)(0x40011800, 0)  
*Base Address of Port E.*
- #define [GPIO\\_E REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_E](#))  
*GPIO Port E.*
- #define [BASE\\_ADDRESS\\_PORT\\_F REGISTER\\_ADDRESS](#)(0x40011C00, 0)  
*Base Address of Port F.*
- #define [GPIO\\_F REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_F](#))  
*GPIO Port F.*
- #define [BASE\\_ADDRESS\\_PORT\\_G REGISTER\\_ADDRESS](#)(0x40012000, 0)  
*Base Address of Port G.*
- #define [GPIO\\_G REGISTER](#)(t\_GPIOx\_RegisterMap, [BASE\\_ADDRESS\\_PORT\\_G](#))  
*GPIO Port G.*

## 6.9.1 Detailed Description

GPIO Addresses.

This module contains the addresses of the GPIO registers

## 6.9.2 Macro Definition Documentation

### 6.9.2.1 BASE\_ADDRESS\_PORT\_A

```
#define BASE_ADDRESS_PORT_A REGISTER_ADDRESS(0x40010800, 0)
```

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

Base Address of Port A.

### 6.9.2.2 GPIO\_A

```
#define GPIO_A REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_A)
```

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

GPIO Port A.

### 6.9.2.3 BASE\_ADDRESS\_PORT\_B

```
#define BASE_ADDRESS_PORT_B REGISTER_ADDRESS(0x40010C00, 0)
```

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

Base Address of Port B.

### 6.9.2.4 GPIO\_B

```
#define GPIO_B REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_B)
```

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

GPIO Port B.



#### 6.9.2.5 BASE\_ADDRESS\_PORT\_C

```
#define BASE_ADDRESS_PORT_C REGISTER_ADDRESS(0x40011000, 0)
```

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

Base Address of Port C.

#### 6.9.2.6 GPIO\_C

```
#define GPIO_C REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_C)
```

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

GPIO Port C.

#### 6.9.2.7 BASE\_ADDRESS\_PORT\_D

```
#define BASE_ADDRESS_PORT_D REGISTER_ADDRESS(0x40011400, 0)
```

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

Base Address of Port D.

#### 6.9.2.8 GPIO\_D

```
#define GPIO_D REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_D)
```

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

GPIO Port D.

#### 6.9.2.9 BASE\_ADDRESS\_PORT\_E

```
#define BASE_ADDRESS_PORT_E REGISTER_ADDRESS(0x40011800, 0)
```

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

Base Address of Port E.

#### 6.9.2.10 GPIO\_E

```
#define GPIO_E REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_E)

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

GPIO Port E.

#### 6.9.2.11 BASE\_ADDRESS\_PORT\_F

```
#define BASE_ADDRESS_PORT_F REGISTER_ADDRESS(0x40011C00, 0)

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

Base Address of Port F.

#### 6.9.2.12 GPIO\_F

```
#define GPIO_F REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_F)

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

GPIO Port F.

#### 6.9.2.13 BASE\_ADDRESS\_PORT\_G

```
#define BASE_ADDRESS_PORT_G REGISTER_ADDRESS(0x40012000, 0)

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

Base Address of Port G.

#### 6.9.2.14 GPIO\_G

```
#define GPIO_G REGISTER(t_GPIOx_RegisterMap, BASE_ADDRESS_PORT_G)

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

GPIO Port G.

## 6.10 GPIO Pins Constants

GPIO Pins Constants.

Collaboration diagram for GPIO Pins Constants:

### Macros

- `#define PIN_SHIFT_VALUE (4)`  
*Pin Shift Value.*
- `#define PIN_RESET_MASK_VALUE ((t_u32)0x0000000FU)`  
*Pin Reset Mask Value.*
- `#define PIN_RESET_CONFIGURATION_MASK_VALUE ((t_u32)0x0000000CU)`  
*Pin Reset Configuration Mask Value.*
- `#define PIN_RESET_ODR_MASK_VALUE ((t_u32)0x00000001U)`  
*Pin Reset ODR Mask Value.*
- `#define PIN_CONFIGURATION_BITS_SHIFT_VALUE (2)`  
*Pin Mode Bits Shift Value.*
- `#define PIN_RESET_SHIFT_VALUE (16)`  
*Pin Reset Shift Value.*
- `#define PIN_RESET_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset Mask.*
- `#define PIN_RESET_CONFIGURATIONS_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_CONFIGURATION_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset Configurations Mask.*
- `#define PIN_RESET_ODR_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_ODR_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset ODR Mask.*

### 6.10.1 Detailed Description

GPIO Pins Constants.

This module contains a set of constants used in configuring the GPIO pins in the configuration register (CRL or CRH)

### 6.10.2 Macro Definition Documentation

#### 6.10.2.1 PIN\_SHIFT\_VALUE

```
#define PIN_SHIFT_VALUE (4)
```

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

Pin Shift Value.

This value is used to shift to the start of a specific pin in the configuration register (CRL or CRH)

### 6.10.2.2 PIN\_RESET\_MASK\_VALUE

```
#define PIN_RESET_MASK_VALUE ((t_u32)0x0000000FU)

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

Pin Reset Mask Value.

This value is used to reset the mode and configuration bits of a certain pin

### 6.10.2.3 PIN\_RESET\_CONFIGURATION\_MASK\_VALUE

```
#define PIN_RESET_CONFIGURATION_MASK_VALUE ((t_u32)0x0000000CU)

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

Pin Reset Configuration Mask Value.

This value is used to reset the configuration bits of a certain pin

### 6.10.2.4 PIN\_RESET\_ODR\_MASK\_VALUE

```
#define PIN_RESET_ODR_MASK_VALUE ((t_u32)0x00000001U)

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

Pin Reset ODR Mask Value.

This value is used to reset the ODR bit of a certain pin (in the BSRR register)

### 6.10.2.5 PIN\_CONFIGURATION\_BITS\_SHIFT\_VALUE

```
#define PIN_CONFIGURATION_BITS_SHIFT_VALUE (2)

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

Pin Mode Bits Shift Value.

This value is used to shift to the configuration bits of a certain pin

### 6.10.2.6 PIN\_RESET\_SHIFT\_VALUE

```
#define PIN_RESET_SHIFT_VALUE (16)

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

Pin Reset Shift Value.

This value is used to shift to the start of the position of the first pin bit in the BSRR register

### 6.10.2.7 PIN\_RESET\_MASK

```
#define PIN_RESET_MASK(
    GPIO_PIN_SPAN ) ~(PIN_RESET_MASK_VALUE << GPIO_PIN_SPAN)

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

Pin Reset Mask.

This mask is used to reset the mode and configuration bits of a certain pin

## Parameters

in	<i>GPIO_PIN_SPAN</i>	The span of the pin in the configuration register (CRL or CRH)
----	----------------------	--

**6.10.2.8 PIN\_RESET\_CONFIGURATIONS\_MASK**

```
#define PIN_RESET_CONFIGURATIONS_MASK(  
    GPIO_PIN_SPAN ) ~ (PIN_RESET_CONFIGURATION_MASK_VALUE << GPIO_PIN_SPAN)
```

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

Pin Reset Configurations Mask.

This mask is used to reset the configuration bits of a certain pin

## Parameters

in	<i>GPIO_PIN_SPAN</i>	The span of the pin in the configuration register (CRL or CRH)
----	----------------------	--

**6.10.2.9 PIN\_RESET\_ODR\_MASK**

```
#define PIN_RESET_ODR_MASK(  
    GPIO_PIN_SPAN ) ~ (PIN_RESET_ODR_MASK_VALUE << GPIO_PIN_SPAN)
```

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

Pin Reset ODR Mask.

This mask is used to reset the ODR bit of a certain pin (in the BSRR register)

## Parameters

in	<i>GPIO_PIN_SPAN</i>	The span of the pin in the configuration register (CRL or CRH)
----	----------------------	--

**6.11 MCAL Peripherals**

MCAL Peripheral Drivers.

Collaboration diagram for MCAL Peripherals:

## Modules

- [GPIO Module](#)  
*GPIO Module.*
- [RCC Module](#)  
*RCC Module.*

### 6.11.1 Detailed Description

MCAL Peripheral Drivers.

## 6.12 RCC Configuration

This group contains the configuration parameters of the RCC module.

Collaboration diagram for RCC Configuration:

## Macros

- `#define RCC_PLL_SRC (RCC_PLLSource_HSE)`  
*This macro defines the PLL source.*
- `#define RCC_PLL_MUL (RCC_PLLMulFactors_9)`  
*This macro defines the PLL multiplication factor.*
- `#define RCC_PLL_HSE_DIVIDE_BY_2 (TRUE)`  
*This macro defines the PLL entry HSE divider.*
- `#define RCC_SYSTEM_CLOCK_SOURCE (RCC_SystemClock_PLL)`  
*This macro defines the system clock source.*
- `#define RCC_MCO_SOURCE (RCC_MCOSources_NoClock)`  
*This macro defines the MCO source.*
- `#define RCC_ADC_PRESCALER (RCC_ADCPrescaler_DividedBy4)`  
*This macro defines the ADC clock prescaler.*
- `#define RCC_AHB_PRESCALER (RCC_AHBPrescaler_NotDivided)`  
*This macro defines the AHB clock prescaler.*
- `#define RCC_APB1_PRESCALER (RCC_APBPrescaler_NotDivided)`  
*This macro defines the APB1 clock prescaler.*
- `#define RCC_APB2_PRESCALER (RCC_APBPrescaler_NotDivided)`  
*This macro defines the APB2 clock prescaler.*
- `#define RCC_USB_PRESCALER (RCC_USBPrescaler_1)`  
*This macro defines the USB clock prescaler.*
- `#define RCC_ENABLE_CSS (FALSE)`  
*This macro defines the clock security system state.*

### 6.12.1 Detailed Description

This group contains the configuration parameters of the RCC module.

## 6.12.2 Macro Definition Documentation

### 6.12.2.1 RCC\_PLL\_SRC

```
#define RCC_PLL_SRC (RCC_PLLSource_HSE)

#include <MCAL/RCC/RCC_config.h>
```

This macro defines the PLL source.

This macro defines the PLL source

See also

[RCC\\_PLLSource](#)

### 6.12.2.2 RCC\_PLL\_MUL

```
#define RCC_PLL_MUL (RCC_PLLMulFactors_9)

#include <MCAL/RCC/RCC_config.h>
```

This macro defines the PLL multiplication factor.

This macro defines the PLL multiplication factor

See also

[RCC\\_PLLMulFactors](#)

### 6.12.2.3 RCC\_PLL\_HSE\_DIVIDE\_BY\_2

```
#define RCC_PLL_HSE_DIVIDE_BY_2 (TRUE)

#include <MCAL/RCC/RCC_config.h>
```

This macro defines the PLL entry HSE divider.

This macro defines the PLL entry HSE divider Options:

- FALSE: HSE clock is not divided
- TRUE: HSE clock divided by 2

#### 6.12.2.4 RCC\_SYSTEM\_CLOCK\_SOURCE

```
#define RCC_SYSTEM_CLOCK_SOURCE (RCC_SystemClock_PLL)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the system clock source.

This macro defines the system clock source

See also

[RCC\\_SystemClock](#)

#### 6.12.2.5 RCC\_MCO\_SOURCE

```
#define RCC_MCO_SOURCE (RCC_MCOSources_NoClock)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the MCO source.

This macro defines the MCO source

See also

[RCC\\_MCOSources](#)

#### 6.12.2.6 RCC\_ADC\_PRESCALER

```
#define RCC_ADC_PRESCALER (RCC_ADCPrescaler_DividedBy4)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the ADC clock prescaler.

This macro defines the ADC clock prescaler

See also

[RCC\\_ADCPrescaler](#)



### 6.12.2.7 RCC\_AHB\_PRESCALER

```
#define RCC_AHB_PRESCALER (RCC_AHBPrescaler_NotDivided)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the AHB clock prescaler.

This macro defines the AHB clock prescaler

See also

[RCC\\_AHBPrescaler](#)

### 6.12.2.8 RCC\_APB1\_PRESCALER

```
#define RCC_APB1_PRESCALER (RCC_APBPrescaler_NotDivided)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the APB1 clock prescaler.

This macro defines the APB1 clock prescaler

See also

[RCC\\_APBPrescaler](#)

### 6.12.2.9 RCC\_APB2\_PRESCALER

```
#define RCC_APB2_PRESCALER (RCC_APBPrescaler_NotDivided)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the APB2 clock prescaler.

This macro defines the APB2 clock prescaler

See also

[RCC\\_APBPrescaler](#)

#### 6.12.2.10 RCC\_USB\_PRESCALER

```
#define RCC_USB_PRESCALER (RCC_USBPrescaler_1)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the USB clock prescaler.

This macro defines the USB clock prescaler

See also

[RCC\\_USBPrescaler](#)

#### 6.12.2.11 RCC\_ENABLE\_CSS

```
#define RCC_ENABLE_CSS (FALSE)
```

```
#include <MCAL/RCC/RCC_config.h>
```

This macro defines the clock security system state.

This macro defines the clock security system state Options:

- FALSE: CSS disabled
- TRUE: CSS enabled

### 6.13 RCC Interface Options

This group contains the options of the RCC module interface.

Collaboration diagram for RCC Interface Options:

## Enumerations

- enum `RCC_PLLMulFactors` {  
`RCC_PLLMulFactors_2` = 0 , `RCC_PLLMulFactors_3` , `RCC_PLLMulFactors_4` , `RCC_PLLMulFactors_5` ,  
`RCC_PLLMulFactors_6` , `RCC_PLLMulFactors_7` , `RCC_PLLMulFactors_8` , `RCC_PLLMulFactors_9` ,  
`RCC_PLLMulFactors_10` , `RCC_PLLMulFactors_11` , `RCC_PLLMulFactors_12` , `RCC_PLLMulFactors_13` ,  
`RCC_PLLMulFactors_14` , `RCC_PLLMulFactors_15` , `RCC_PLLMulFactors_16` }  
*This enum contains the multiplication factors of the PLL.*
- enum `RCC_AHBPrescaler` {  
`RCC_AHBPrescaler_NotDivided` = 0 , `RCC_AHBPrescaler_DividedBy2` = 0b1000 , `RCC_AHBPrescaler_DividedBy4` ,  
`RCC_AHBPrescaler_DividedBy8` ,  
`RCC_AHBPrescaler_DividedBy16` , `RCC_AHBPrescaler_DividedBy64` , `RCC_AHBPrescaler_DividedBy128` ,  
`RCC_AHBPrescaler_DividedBy256` ,  
`RCC_AHBPrescaler_DividedBy512` }  
*This enum contains the prescaler factors of the AHB bus.*
- enum `RCC_APBPrescaler` {  
`RCC_APBPrescaler_NotDivided` = 0 , `RCC_APBPrescaler_DividedBy2` = 0b100 , `RCC_APBPrescaler_DividedBy4` ,  
`RCC_APBPrescaler_DividedBy8` ,  
`RCC_APBPrescaler_DividedBy16` }  
*This enum contains the prescaler factors of the APB (1 & 2) bus.*
- enum `RCC_ADCPrescaler` { `RCC_ADCPrescaler_DividedBy2` = 0 , `RCC_ADCPrescaler_DividedBy4` ,  
`RCC_ADCPrescaler_DividedBy6` , `RCC_ADCPrescaler_DividedBy8` }  
*This enum contains the prescaler factors of the ADC bus.*
- enum `RCC_SystemClock` { `RCC_SystemClock_HSI` = 0 , `RCC_SystemClock_HSE` , `RCC_SystemClock_PLL` }  
*This enum contains the system clock sources.*
- enum `RCC_PLLSource` { `RCC_PLLSource_HSI_DividedBy2` = 0 , `RCC_PLLSource_HSE` }  
*This enum contains the PLL clock sources.*
- enum `RCC_MCOSources` {  
`RCC_MCOSources_NoClock` = 0 , `RCC_MCOSources_SystemClock` = 0b100 , `RCC_MCOSources_HSI` ,  
`RCC_MCOSources_HSE` ,  
`RCC_MCOSources_PLL_DividedBy2` }  
*This enum contains the MCO sources.*
- enum `RCC_USBPrescaler` { `RCC_USBPrescaler_1_5` = 0 , `RCC_USBPrescaler_1` }  
*This enum contains the USB prescaler factors.*
- enum `t_RCC_APB2Peripherals` {  
`RCC_APB2Peripherals_AFIO` = 0 , `RCC_APB2Peripherals_PORTA` = 2 , `RCC_APB2Peripherals_PORTB` ,  
`RCC_APB2Peripherals_PORTC` ,  
`RCC_APB2Peripherals_PORTD` , `RCC_APB2Peripherals_PORTE` , `RCC_APB2Peripherals_PORTF` ,  
`RCC_APB2Peripherals_PORTG` ,  
`RCC_APB2Peripherals_ADC1` , `RCC_APB2Peripherals_ADC2` , `RCC_APB2Peripherals_TIM1` , `RCC_APB2Peripherals_SPI1` ,  
`RCC_APB2Peripherals_USART1` = 14 , `RCC_APB2Peripherals_ADC3` }  
*This enum contains the APB2 peripherals that are connected to the APB2 bus.*
- enum `t_RCC_APB1Peripherals` {  
`RCC_APB1Peripherals_TIM2` = 0 , `RCC_APB1Peripherals_TIM3` , `RCC_APB1Peripherals_TIM4` ,  
`RCC_APB1Peripherals_TIM5` ,  
`RCC_APB1Peripherals_TIM6` , `RCC_APB1Peripherals_TIM7` , `RCC_APB1Peripherals_WWDG` = 11 ,  
`RCC_APB1Peripherals_SPI2` = 14 ,  
`RCC_APB1Peripherals_SPI3` , `RCC_APB1Peripherals_USART2` = 17 , `RCC_APB1Peripherals_USART3` ,  
`RCC_APB1Peripherals_UART4` ,  
`RCC_APB1Peripherals_UART5` , `RCC_APB1Peripherals_I2C1` , `RCC_APB1Peripherals_I2C2` , `RCC_APB1Peripherals_USB` ,  
`RCC_APB1Peripherals_CAN` = 25 , `RCC_APB1Peripherals_BKP` = 27 , `RCC_APB1Peripherals_PWR` ,  
`RCC_APB1Peripherals_DAC` }

*This enum contains the APB1 peripherals that are connected to the APB1 bus.*

- enum `t_RCC_AHBPeripherals` { `RCC_AHBPeripherals_DMA1` = 0 , `RCC_AHBPeripherals_DMA2` , `RCC_AHBPeripherals_CRC` = 5 }

*This enum contains the AHB peripherals that are connected to the AHB bus.*

### 6.13.1 Detailed Description

This group contains the options of the RCC module interface.

### 6.13.2 Enumeration Type Documentation

#### 6.13.2.1 RCC\_PLLMulFactors

```
enum RCC_PLLMulFactors
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the multiplication factors of the PLL.

This enum contains the multiplication factors of the PLL

Enumerator

<code>RCC_PLLMulFactors_2</code>	Multiply the PLL1 input clock by 2.
<code>RCC_PLLMulFactors_3</code>	Multiply the PLL1 input clock by 3.
<code>RCC_PLLMulFactors_4</code>	Multiply the PLL1 input clock by 4.
<code>RCC_PLLMulFactors_5</code>	Multiply the PLL1 input clock by 5.
<code>RCC_PLLMulFactors_6</code>	Multiply the PLL1 input clock by 6.
<code>RCC_PLLMulFactors_7</code>	Multiply the PLL1 input clock by 7.
<code>RCC_PLLMulFactors_8</code>	Multiply the PLL1 input clock by 8.
<code>RCC_PLLMulFactors_9</code>	Multiply the PLL1 input clock by 9.
<code>RCC_PLLMulFactors_10</code>	Multiply the PLL1 input clock by 10.
<code>RCC_PLLMulFactors_11</code>	Multiply the PLL1 input clock by 11.
<code>RCC_PLLMulFactors_12</code>	Multiply the PLL1 input clock by 12.
<code>RCC_PLLMulFactors_13</code>	Multiply the PLL1 input clock by 13.
<code>RCC_PLLMulFactors_14</code>	Multiply the PLL1 input clock by 14.
<code>RCC_PLLMulFactors_15</code>	Multiply the PLL1 input clock by 15.
<code>RCC_PLLMulFactors_16</code>	Multiply the PLL1 input clock by 16.

```
35 {
39     RCC_PLLMulFactors_2 = 0,
43     RCC_PLLMulFactors_3,
47     RCC_PLLMulFactors_4,
51     RCC_PLLMulFactors_5,
55     RCC_PLLMulFactors_6,
59     RCC_PLLMulFactors_7,
63     RCC_PLLMulFactors_8,
67     RCC_PLLMulFactors_9,
71     RCC_PLLMulFactors_10,
75     RCC_PLLMulFactors_11,
```

```

79     RCC_PLLMulFactors_12,
83     RCC_PLLMulFactors_13,
87     RCC_PLLMulFactors_14,
91     RCC_PLLMulFactors_15,
95     RCC_PLLMulFactors_16
96 };

```

### 6.13.2.2 RCC\_AHBPrescaler

```
enum RCC_AHBPrescaler
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the prescaler factors of the AHB bus.

This enum contains the prescaler factors of the AHB bus

#### Enumerator

RCC_AHBPrescaler_NotDivided	AHB bus not divided.
RCC_AHBPrescaler_DividedBy2	AHB bus divided by 2.
RCC_AHBPrescaler_DividedBy4	AHB bus divided by 4.
RCC_AHBPrescaler_DividedBy8	AHB bus divided by 8.
RCC_AHBPrescaler_DividedBy16	AHB bus divided by 16.
RCC_AHBPrescaler_DividedBy64	AHB bus divided by 64.
RCC_AHBPrescaler_DividedBy128	AHB bus divided by 128.
RCC_AHBPrescaler_DividedBy256	AHB bus divided by 256.
RCC_AHBPrescaler_DividedBy512	AHB bus divided by 512.

```

104 {
108     RCC_AHBPrescaler_NotDivided = 0,
112     RCC_AHBPrescaler_DividedBy2 = 0b1000,
116     RCC_AHBPrescaler_DividedBy4,
120     RCC_AHBPrescaler_DividedBy8,
124     RCC_AHBPrescaler_DividedBy16,
128     RCC_AHBPrescaler_DividedBy64,
132     RCC_AHBPrescaler_DividedBy128,
136     RCC_AHBPrescaler_DividedBy256,
140     RCC_AHBPrescaler_DividedBy512
141 };

```

### 6.13.2.3 RCC\_APBPrescaler

```
enum RCC_APBPrescaler
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the prescaler factors of the APB (1 & 2) bus.

This enum contains the prescaler factors of the APB (1 & 2) bus

#### Enumerator

RCC_APBPrescaler_NotDivided	APB1 bus not divided.
-----------------------------	-----------------------

**Enumerator**

RCC_APBPrescaler_DividedBy2	APB1 bus divided by 2.
RCC_APBPrescaler_DividedBy4	APB1 bus divided by 4.
RCC_APBPrescaler_DividedBy8	APB1 bus divided by 8.
RCC_APBPrescaler_DividedBy16	APB1 bus divided by 16.

```

149 {
153     RCC_APBPrescaler_NotDivided = 0,
157     RCC_APBPrescaler_DividedBy2 = 0b100,
161     RCC_APBPrescaler_DividedBy4,
165     RCC_APBPrescaler_DividedBy8,
169     RCC_APBPrescaler_DividedBy16
170 };

```

**6.13.2.4 RCC\_ADCPrescaler**

```
enum RCC_ADCPrescaler
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the prescaler factors of the ADC bus.

This enum contains the prescaler factors of the ADC bus

**Enumerator**

RCC_ADCPrescaler_DividedBy2	ADC bus divided by 2.
RCC_ADCPrescaler_DividedBy4	ADC bus divided by 4.
RCC_ADCPrescaler_DividedBy6	ADC bus divided by 6.
RCC_ADCPrescaler_DividedBy8	ADC bus divided by 8.

```

178 {
182     RCC_ADCPrescaler_DividedBy2 = 0,
186     RCC_ADCPrescaler_DividedBy4,
190     RCC_ADCPrescaler_DividedBy6,
194     RCC_ADCPrescaler_DividedBy8
195 };

```

**6.13.2.5 RCC\_SystemClock**

```
enum RCC_SystemClock
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the system clock sources.

This enum contains the system clock sources

**Enumerator**

RCC_SystemClock_HSI	HSI clock selected as system clock.
RCC_SystemClock_HSE	HSE clock selected as system clock.
RCC_SystemClock_PLL	PLL clock selected as system clock.

```

203 {
207     RCC_SystemClock_HSI = 0,
211     RCC_SystemClock_HSE,
215     RCC_SystemClock_PLL
216 };

```

### 6.13.2.6 RCC\_PLLSource

```
enum RCC_PLLSource
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the PLL clock sources.

This enum contains the PLL clock sources

#### Enumerator

RCC_PLLSource_HSI_DividedBy2	HSI clock divided by 2 selected as PLL clock.
RCC_PLLSource_HSE	HSE clock selected as PLL clock.

```

224 {
228     RCC_PLLSource_HSI_DividedBy2 = 0,
232     RCC_PLLSource_HSE
233 };

```

### 6.13.2.7 RCC\_MCOSources

```
enum RCC_MCOSources
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the MCO sources.

This enum contains the MCO sources

#### Enumerator

RCC_MCOSources_NoClock	No clock selected as MCO.
RCC_MCOSources_SystemClock	System clock selected as MCO.
RCC_MCOSources_HSI	HSI clock selected as MCO.
RCC_MCOSources_HSE	HSE clock selected as MCO.
RCC_MCOSources_PLL_DividedBy2	PLL clock divided by 2 selected as MCO.

```

241 {
245     RCC_MCOSources_NoClock = 0,
249     RCC_MCOSources_SystemClock = 0b100,
253     RCC_MCOSources_HSI,
257     RCC_MCOSources_HSE,
261     RCC_MCOSources_PLL_DividedBy2
262 };

```

### 6.13.2.8 RCC\_USBPrescaler

```
enum RCC_USBPrescaler
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the USB prescaler factors.

This enum contains the USB prescaler factors

#### Enumerator

RCC_USBPrescaler_1↔ _5	USB clock divided by 1.5.
RCC_USBPrescaler_1	USB clock divided by 1.

```
270 {
274     RCC_USBPrescaler_1_5 = 0,
278     RCC_USBPrescaler_1
279 };
```

### 6.13.2.9 t\_RCC\_APB2Peripherals

```
enum t_RCC_APB2Peripherals
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the APB2 peripherals that are connected to the APB2 bus.

This enum contains the APB2 peripherals that are connected to the APB2 bus

#### Enumerator

RCC_APB2Peripherals_AFIO	Alternate function I/O clock.
RCC_APB2Peripherals_PORTA	I/O port A clock.
RCC_APB2Peripherals_PORTB	I/O port B clock.
RCC_APB2Peripherals_PORTC	I/O port C clock.
RCC_APB2Peripherals_PORTD	I/O port D clock.
RCC_APB2Peripherals_PORTE	I/O port E clock.
RCC_APB2Peripherals_PORTF	I/O port F clock.
RCC_APB2Peripherals_PORTG	I/O port G clock.
RCC_APB2Peripherals_ADC1	ADC 1 interface clock.
RCC_APB2Peripherals_ADC2	ADC 2 interface clock.
RCC_APB2Peripherals_TIM1	Timer 1 (TIM1) clock.
RCC_APB2Peripherals_SPI1	SPI 1 clock.
RCC_APB2Peripherals_USART1	USART1 clock.
RCC_APB2Peripherals_ADC3	ADC 3 interface clock.

```
287 {
291     RCC_APB2Peripherals_AFIO = 0,
295     RCC_APB2Peripherals_PORTA = 2,
299     RCC_APB2Peripherals_PORTB,
303     RCC_APB2Peripherals_PORTC,
```



```

307     RCC_APB2Peripherals_PORTD,
311     RCC_APB2Peripherals_PORTE,
315     RCC_APB2Peripherals_PORTF,
319     RCC_APB2Peripherals_PORTG,
323     RCC_APB2Peripherals_ADC1,
327     RCC_APB2Peripherals_ADC2,
331     RCC_APB2Peripherals_TIM1,
335     RCC_APB2Peripherals_SPI1,
339     RCC_APB2Peripherals_USART1 = 14,
343     RCC_APB2Peripherals_ADC3
344 } t_RCC_APB2Peripherals;

```

### 6.13.2.10 t\_RCC\_APB1Peripherals

```
enum t_RCC_APB1Peripherals
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the APB1 peripherals that are connected to the APB1 bus.

This enum contains the APB1 peripherals that are connected to the APB1 bus

#### Enumerator

RCC_APB1Peripherals_TIM2	Timer 2 (TIM2) clock.
RCC_APB1Peripherals_TIM3	Timer 3 (TIM3) clock.
RCC_APB1Peripherals_TIM4	Timer 4 (TIM4) clock.
RCC_APB1Peripherals_TIM5	Timer 5 (TIM5) clock.
RCC_APB1Peripherals_TIM6	Timer 6 (TIM6) clock.
RCC_APB1Peripherals_TIM7	Timer 7 (TIM7) clock.
RCC_APB1Peripherals_WWDG	Window watchdog (WWDG) clock.
RCC_APB1Peripherals_SPI2	SPI 2 clock.
RCC_APB1Peripherals_SPI3	SPI 3 clock.
RCC_APB1Peripherals_USART2	USART 2 clock.
RCC_APB1Peripherals_USART3	USART 3 clock.
RCC_APB1Peripherals_UART4	UART 4 clock.
RCC_APB1Peripherals_UART5	UART 5 clock.
RCC_APB1Peripherals_I2C1	I2C 1 clock.
RCC_APB1Peripherals_I2C2	I2C 2 clock.
RCC_APB1Peripherals_USB	USB clock.
RCC_APB1Peripherals_CAN	CAN clock.
RCC_APB1Peripherals_BKP	Backup interface clock.
RCC_APB1Peripherals_PWR	Power interface clock.
RCC_APB1Peripherals_DAC	DAC interface clock.

```

352 {
356     RCC_APB1Peripherals_TIM2 = 0,
360     RCC_APB1Peripherals_TIM3,
364     RCC_APB1Peripherals_TIM4,
368     RCC_APB1Peripherals_TIM5,
372     RCC_APB1Peripherals_TIM6,
376     RCC_APB1Peripherals_TIM7,
380     RCC_APB1Peripherals_WWDG = 11,
384     RCC_APB1Peripherals_SPI2 = 14,
388     RCC_APB1Peripherals_SPI3,
392     RCC_APB1Peripherals_USART2 = 17,
396     RCC_APB1Peripherals_USART3,

```

```

400     RCC_APB1Peripherals_UART4,
404     RCC_APB1Peripherals_UART5,
408     RCC_APB1Peripherals_I2C1,
412     RCC_APB1Peripherals_I2C2,
416     RCC_APB1Peripherals_USB,
420     RCC_APB1Peripherals_CAN = 25,
424     RCC_APB1Peripherals_BKP = 27,
428     RCC_APB1Peripherals_PWR,
432     RCC_APB1Peripherals_DAC
433 } t_RCC_APB1Peripherals;

```

### 6.13.2.11 t\_RCC\_AHBPeripherals

```
enum t_RCC_AHBPeripherals
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This enum contains the AHB peripherals that are connected to the AHB bus.

This enum contains the AHB peripherals that are connected to the AHB bus

#### Enumerator

RCC_AHBPeripherals_DMA1	DMA 1 clock.
RCC_AHBPeripherals_DMA2	DMA 2 clock.
RCC_AHBPeripherals_CRC	CRC clock.

```

441 {
445     RCC_AHBPeripherals_DMA1 = 0,
449     RCC_AHBPeripherals_DMA2,
453     RCC_AHBPeripherals_CRC = 5
454 } t_RCC_AHBPeripherals;

```

## 6.14 RCC exported functions

RCC exported functions.

Collaboration diagram for RCC exported functions:

### Functions

- void [RCC\\_vInit](#) (void)  
*This function initializes the RCC peripheral.*
- void [RCC\\_vEnablePeripheralABP2](#) (t\_RCC\_APB2Peripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the APB2 bus.*
- void [RCC\\_vDisablePeripheralABP2](#) (t\_RCC\_APB2Peripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the APB2 bus.*
- void [RCC\\_vEnablePeripheralABP1](#) (t\_RCC\_APB1Peripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the APB1 bus.*
- void [RCC\\_vDisablePeripheralABP1](#) (t\_RCC\_APB1Peripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the APB1 bus.*
- void [RCC\\_vEnablePeripheralAHB](#) (t\_RCC\_AHBPeripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the AHB bus.*
- void [RCC\\_vDisablePeripheralAHB](#) (t\_RCC\_AHBPeripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the AHB bus.*

### 6.14.1 Detailed Description

RCC exported functions.

### 6.14.2 Function Documentation

#### 6.14.2.1 RCC\_vInit()

```
void RCC_vInit (
    void )

#include <MCAL/RCC/RCC_interface.h>
```

This function initializes the RCC peripheral.

This function initializes the RCC peripheral

```
325 {
326     if (RCC_SYSTEM_CLOCK_SOURCE == RCC_SystemClock_HSE)
327     {
328         RCC_vInitHSEClock();
329     }
330     else if (RCC_SYSTEM_CLOCK_SOURCE == RCC_SystemClock_HSI)
331     {
332         RCC_vInitHSIClock();
333     }
334     else if (RCC_SYSTEM_CLOCK_SOURCE == RCC_SystemClock_PLL)
335     {
336         RCC_vInitPLL();
337     }
338     else
339     {
340         /* Do nothing */
341     }
342
343     RCC_vInitSystemClock();
344     RCC_vSetAHBPrescaler();
345     RCC_vSetAPB1Prescaler();
346     RCC_vSetAPB2Prescaler();
347     RCC_vSetADCPreScaler();
348     RCC_vSetUSBPrescaler();
349     RCC_vInitMCO();
350 }
```

References [RCC\\_SYSTEM\\_CLOCK\\_SOURCE](#), [RCC\\_SystemClock\\_HSE](#), [RCC\\_SystemClock\\_HSI](#), [RCC\\_SystemClock\\_PLL](#), [RCC\\_vInitHSEClock\(\)](#), [RCC\\_vInitHSIClock\(\)](#), [RCC\\_vInitMCO\(\)](#), [RCC\\_vInitPLL\(\)](#), [RCC\\_vInitSystemClock\(\)](#), [RCC\\_vSetADCPreScaler\(\)](#), [RCC\\_vSetAHBPrescaler\(\)](#), [RCC\\_vSetAPB1Prescaler\(\)](#), [RCC\\_vSetAPB2Prescaler\(\)](#), and [RCC\\_vSetUSBPrescaler\(\)](#).

Referenced by [vAPPS\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

#### 6.14.2.2 RCC\_vEnablePeripheralABP2()

```
void RCC_vEnablePeripheralABP2 (
    t_RCC_APB2Peripherals enuPeripheral )

#include <MCAL/RCC/RCC_interface.h>
```

This function enables the clock of a peripheral connected to the APB2 bus.

This function enables the clock of a peripheral connected to the APB2 bus

**Parameters**

in	<i>enuPeripheral</i>	Peripheral to be enabled
----	----------------------	--------------------------

```

353 {
354     RCC_vSetAPB2PeripheralClockStatus(enuPeripheral, TRUE);
355 }

```

References [RCC\\_vSetAPB2PeripheralClockStatus\(\)](#), and [TRUE](#).

Referenced by [vTestApp\\_TestingGPIO\\_main\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

**6.14.2.3 RCC\_vDisablePeripheralABP2()**

```

void RCC_vDisablePeripheralABP2 (
    t_RCC_APB2Peripherals enuPeripheral )

```

```
#include <MCAL/RCC/RCC_interface.h>
```

This function disables the clock of a peripheral connected to the APB2 bus.

This function disables the clock of a peripheral connected to the APB2 bus

**Parameters**

in	<i>enuPeripheral</i>	Peripheral to be disabled
----	----------------------	---------------------------

```

358 {
359     RCC_vSetAPB2PeripheralClockStatus(enuPeripheral, FALSE);
360 }

```

References [FALSE](#), and [RCC\\_vSetAPB2PeripheralClockStatus\(\)](#).

Here is the call graph for this function:

**6.14.2.4 RCC\_vEnablePeripheralABP1()**

```

void RCC_vEnablePeripheralABP1 (
    t_RCC_APB1Peripherals enuPeripheral )

```

```
#include <MCAL/RCC/RCC_interface.h>
```

This function enables the clock of a peripheral connected to the APB1 bus.

This function enables the clock of a peripheral connected to the APB1 bus

**Parameters**

in	<i>enuPeripheral</i>	Peripheral to be enabled
----	----------------------	--------------------------

```

363 {
364     RCC_vSetAPB1PeripheralClockStatus(enuPeripheral, TRUE);
365 }

```

References [RCC\\_vSetAPB1PeripheralClockStatus\(\)](#), and [TRUE](#).

Here is the call graph for this function:

#### 6.14.2.5 RCC\_vDisablePeripheralABP1()

```
void RCC_vDisablePeripheralABP1 (
    t_RCC_APB1Peripherals enuPeripheral )
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This function disables the clock of a peripheral connected to the APB1 bus.

This function disables the clock of a peripheral connected to the APB1 bus

##### Parameters

in	<i>enuPeripheral</i>	Peripheral to be disabled
----	----------------------	---------------------------

```
368 {
369     RCC_vSetAPB1PeripheralClockStatus(enuPeripheral, FALSE);
370 }
```

References [FALSE](#), and [RCC\\_vSetAPB1PeripheralClockStatus\(\)](#).

Here is the call graph for this function:

#### 6.14.2.6 RCC\_vEnablePeripheralAHB()

```
void RCC_vEnablePeripheralAHB (
    t_RCC_AHBPeripherals enuPeripheral )
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This function enables the clock of a peripheral connected to the AHB bus.

This function enables the clock of a peripheral connected to the AHB bus

##### Parameters

in	<i>enuPeripheral</i>	Peripheral to be enabled
----	----------------------	--------------------------

```
373 {
374     RCC_vSetAHBPeripheralClockStatus(enuPeripheral, TRUE);
375 }
```

References [RCC\\_vSetAHBPeripheralClockStatus\(\)](#), and [TRUE](#).

Here is the call graph for this function:

#### 6.14.2.7 RCC\_vDisablePeripheralIAHB()

```
void RCC_vDisablePeripheralIAHB (
    t_RCC_AHBPeripherals enuPeripheral )
```

```
#include <MCAL/RCC/RCC_interface.h>
```

This function disables the clock of a peripheral connected to the AHB bus.

This function disables the clock of a peripheral connected to the AHB bus

#### Parameters

in	<i>enuPeripheral</i>	Peripheral to be disabled
----	----------------------	---------------------------

```
378 {
379     RCC_vSetAHBPeripheralClockStatus(enuPeripheral, FALSE);
380 }
```

References [FALSE](#), and [RCC\\_vSetAHBPeripheralClockStatus\(\)](#).

Here is the call graph for this function:

## 6.15 RCC Module

RCC Module.

Collaboration diagram for RCC Module:

### Modules

- [RCC Configuration](#)  
*This group contains the configuration parameters of the RCC module.*
- [RCC Interface Options](#)  
*This group contains the options of the RCC module interface.*
- [RCC exported functions](#)  
*RCC exported functions.*
- [RCC Registers](#)  
*RCC Registers.*
- [RCC Addresses](#)  
*RCC Addresses.*

### 6.15.1 Detailed Description

RCC Module.

RCC Module contains the functions for controlling the clock system of the microcontroller.

## 6.16 RCC Registers

RCC Registers.

Collaboration diagram for RCC Registers:

## Data Structures

- struct [t\\_RCC\\_CR](#)  
*RCC clock control register.*
- struct [t\\_RCC\\_CFGR](#)  
*RCC clock configuration register.*
- struct [t\\_RCC\\_CIR](#)  
*RCC clock interrupt register.*
- struct [t\\_RCC\\_APB2RSTR](#)  
*RCC APB2 peripheral reset register.*
- struct [t\\_RCC\\_APB1RSTR](#)  
*RCC APB1 peripheral reset register.*
- struct [t\\_RCC\\_AHBENR](#)  
*RCC AHB peripheral clock register.*
- struct [t\\_RCC\\_APB2ENR](#)  
*RCC APB2 peripheral clock enable register.*
- struct [t\\_RCC\\_APB1ENR](#)  
*RCC APB1 peripheral clock enable register.*
- struct [t\\_RCC\\_BDCR](#)  
*RCC Backup domain control register.*
- struct [t\\_RCC\\_CSR](#)  
*RCC Control/status register.*
- struct [t\\_RCC\\_RegisterMap](#)  
*RCC Register Map.*

### 6.16.1 Detailed Description

RCC Registers.

## 6.17 RCC Addresses

RCC Addresses.

Collaboration diagram for RCC Addresses:

## Macros

- #define [RCC\\_BASE\\_ADDRESS\\_REGISTER\\_ADDRESS](#)(0x40021000, 0)  
*RCC Base Address in the memory.*
- #define [RCC\\_REGISTER](#)(t\_RCC\_RegisterMap, RCC\_BASE\_ADDRESS)  
*RCC Register Map.*
- #define [RCC\\_SET\\_REGISTER\\_BIT\\_STATUS](#)(REG, BIT, BSTATUS) [SET\\_REGISTER\\_BIT\\_STATUS](#)(RCC, REG, BIT, BSTATUS)  
*Set RCC register bit status (TRUE or FALSE)*

### 6.17.1 Detailed Description

RCC Addresses.

## 6.17.2 Macro Definition Documentation

### 6.17.2.1 RCC\_BASE\_ADDRESS

```
#define RCC_BASE_ADDRESS REGISTER_ADDRESS(0x40021000, 0)
```

```
#include <MCAL/RCC/RCC_private.h>
```

RCC Base Address in the memory.

### 6.17.2.2 RCC

```
#define RCC REGISTER(t_RCC_RegisterMap, RCC_BASE_ADDRESS)
```

```
#include <MCAL/RCC/RCC_private.h>
```

RCC Register Map.

### 6.17.2.3 RCC\_SET\_REGISTER\_BIT\_STATUS

```
#define RCC_SET_REGISTER_BIT_STATUS(  
    REG,  
    BIT,  
    BSTATUS ) SET_REGISTER_BIT_STATUS(RCC, REG, BIT, BSTATUS)
```

```
#include <MCAL/RCC/RCC_private.h>
```

Set RCC register bit status (TRUE or FALSE)

This macro sets the RCC register bit status (TRUE or FALSE)

#### Parameters

in	<i>REG</i>	Register name
in	<i>BIT</i>	Bit name
in	<i>BSTATUS</i>	Bit status value (TRUE or FALSE)

## 6.18 Testing Applications

Testing Applications.



## Functions

- void [vAPPS\\_main](#) (void)  
*This function is responsible for running the applications.*
- void [vTestApp\\_TestingGPIO\\_main](#) (void)  
*This function is the main function for the TestApp\_TestingGPIO application.*

### 6.18.1 Detailed Description

Testing Applications.

This module contains all the testing applications

### 6.18.2 Function Documentation

#### 6.18.2.1 vAPPS\_main()

```
void vAPPS_main (  
    void )
```

```
#include <APPS/APPS_main.h>
```

This function is responsible for running the applications.

This function is responsible for running the applications

```
26 {  
27     RCC\_vInit();  
28     TESTING\_APPLICATION\_MAIN\_FUNC(TestingGPIO);  
29  
30     for (;;)   
31     {  
32         /* Do nothing */  
33     }  
34 }
```

References [RCC\\_vInit\(\)](#), [TESTING\\_APPLICATION\\_MAIN\\_FUNC](#), and [TestingGPIO](#).

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

Here is the call graph for this function: Here is the caller graph for this function:

### 6.18.2.2 vTestApp\_TestingGPIO\_main()

```
void vTestApp_TestingGPIO_main (
    void )
```

```
#include <APPS/TestingGPIO/TestApp_TestingGPIO_main.h>
```

This function is the main function for the TestApp\_TestingGPIO application.

This function is the main function for the TestApp\_TestingGPIO application

```
16 {
17     // t_u32 u32Counter = 0;
18     RCC_vEnablePeripheralABP2(RCC_APB2Peripherals_PORTA);
19     RCC_vEnablePeripheralABP2(RCC_APB2Peripherals_PORTB);
20     GPIO_vSetPinDirection(GPIO_Ports_A, GPIO_Pins_1, GPIO_Direction_Output_50MHz);
21     GPIO_vSetPinDirection(GPIO_Ports_B, GPIO_Pins_7, GPIO_Direction_Input);
22     // GPIO_vSetPinInputType(GPIO_Ports_B, GPIO_Pins_7, GPIO_Input_Type_Pull_Down);
23     GPIO_vSetPinInputType(GPIO_Ports_B, GPIO_Pins_7, GPIO_Input_Type_Pull_Up);
24
25     for (;;)
26     {
27         GPIO_vSetPinValue(GPIO_Ports_A, GPIO_Pins_1, GPIO_tGetPinValue(GPIO_Ports_B, GPIO_Pins_7));
28         // if (GPIO_tGetPinValue(GPIO_Ports_B, GPIO_Pins_7) == GPIO_Value_High)
29         // if (GPIO_tGetPinValue(GPIO_Ports_B, GPIO_Pins_7) == GPIO_Value_Low)
30         // {
31             // GPIO_vSetPinValue(GPIO_Ports_A, GPIO_Pins_1, GPIO_Value_High);
32         // }
33         // else
34         // {
35             // GPIO_vSetPinValue(GPIO_Ports_A, GPIO_Pins_1, GPIO_Value_Low);
36         // }
37
38         // GPIO_vSetPinValue(GPIO_Ports_A, GPIO_Pins_1, GPIO_Value_High);
39         // for (u32Counter = 0; u32Counter < 10000000; u32Counter++)
40         // ;
41         // GPIO_vSetPinValue(GPIO_Ports_A, GPIO_Pins_1, GPIO_Value_Low);
42         // for (u32Counter = 0; u32Counter < 10000000; u32Counter++)
43         // ;
44     }
45 }
```

References [GPIO\\_Direction\\_Input](#), [GPIO\\_Direction\\_Output\\_50MHz](#), [GPIO\\_Input\\_Type\\_Pull\\_Up](#), [GPIO\\_Pins\\_1](#), [GPIO\\_Pins\\_7](#), [GPIO\\_Ports\\_A](#), [GPIO\\_Ports\\_B](#), [GPIO\\_tGetPinValue\(\)](#), [GPIO\\_vSetPinDirection\(\)](#), [GPIO\\_vSetPinInputType\(\)](#), [GPIO\\_vSetPinValue\(\)](#), [RCC\\_APB2Peripherals\\_PORTA](#), [RCC\\_APB2Peripherals\\_PORTB](#), and [RCC\\_vEnablePeripheralABP2\(\)](#).

Here is the call graph for this function:

## 6.19 Shared library

This library contains the shared macros and functions for all the standard types.

Collaboration diagram for Shared library:

### Macros

- `#define SET_REGISTER_BIT_STATUS(PERPH, REG, BIT, BSTATUS) (PERPH.REG.BIT = (BSTATUS == TRUE) ? 1U : 0U)`  
Set register bit status (1 or 0)

### 6.19.1 Detailed Description

This library contains the shared macros and functions for all the standard types.

## 6.19.2 Macro Definition Documentation

### 6.19.2.1 SET\_REGISTER\_BIT\_STATUS

```
#define SET_REGISTER_BIT_STATUS(  
    PERPH,  
    REG,  
    BIT,  
    BSTATUS ) (PERPH.REG.BIT = (BSTATUS == TRUE) ? 1U : 0U)
```

```
#include <LIB/LSTD_SHARED.h>
```

Set register bit status (1 or 0)

This macro is used to set a specific bit in a specific register

#### Parameters

in	<i>PERPH</i>	Peripheral address
in	<i>REG</i>	Register name
in	<i>BIT</i>	Bit name
in	<i>BSTATUS</i>	Bit status (TRUE or FALSE)



## Chapter 7

# Data Structure Documentation

### 7.1 t\_GPIOx\_RegisterMap Struct Reference

GPIO Register Map.

```
#include "MCAL/GPIO/GPIO_private.h"
```

Collaboration diagram for t\_GPIOx\_RegisterMap:

#### Data Fields

- [t\\_u32 CRL](#)  
*Port Configuration Register Low.*
- [t\\_u32 CRH](#)  
*Port Configuration Register High.*
- [t\\_u32 IDR](#)  
*Port Input Data Register.*
- [t\\_u32 ODR](#)  
*Port Output Data Register.*
- [t\\_u32 BSRR](#)  
*Port Bit Set/Reset Register.*
- [t\\_u32 BRR](#)  
*Port Bit Reset Register.*

#### 7.1.1 Detailed Description

GPIO Register Map.

This type is used to access the GPIO registers

#### 7.1.2 Field Documentation

#### 7.1.2.1 CRL

`t_u32` CRL

Port Configuration Register Low.

#### 7.1.2.2 CRH

`t_u32` CRH

Port Configuration Register High.

#### 7.1.2.3 IDR

`t_u32` IDR

Port Input Data Register.

##### Warning

This register is read-only

#### 7.1.2.4 ODR

`t_u32` ODR

Port Output Data Register.

Referenced by [GPIO\\_vSetPinInputTypePullUpDown\(\)](#).

#### 7.1.2.5 BSRR

`t_u32` BSRR

Port Bit Set/Reset Register.

### 7.1.2.6 BRR

[t\\_u32](#) BRR

Port Bit Reset Register.

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

- [MCAL/GPIO/GPIO\\_private.h](#)

## 7.2 t\_RCC\_AHBENR Struct Reference

RCC AHB peripheral clock register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_AHBENR:

### Data Fields

- [t\\_u32 DMA1EN](#): 1  
*DMA1 clock enable.*
- [t\\_u32 DMA2EN](#): 1  
*DMA2 clock enable.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 4  
*Reserved bit(s)*
- [t\\_u32 CRCEN](#): 1  
*CRC clock enable.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 25  
*Reserved bit(s)*

### 7.2.1 Detailed Description

RCC AHB peripheral clock register.

### 7.2.2 Field Documentation

#### 7.2.2.1 DMA1EN

[t\\_u32](#) DMA1EN

DMA1 clock enable.

This field enables the DMA1 clock

### 7.2.2.2 DMA2EN

`t_u32 DMA2EN`

DMA2 clock enable.

This field enables the DMA2 clock

### 7.2.2.3 \_\_pad0\_\_

`t_u32 __pad0__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.2.2.4 CRCEN

`t_u32 CRCEN`

CRC clock enable.

This field enables the CRC clock

### 7.2.2.5 \_\_pad1\_\_

`t_u32 __pad1__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- [MCAL/RCC/RCC\\_private.h](#)

## 7.3 t\_RCC\_APB1ENR Struct Reference

RCC APB1 peripheral clock enable register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_APB1ENR:



## Data Fields

- [t\\_u32 TIM2EN](#): 1  
*Timer 2 clock enable.*
- [t\\_u32 TIM3EN](#): 1  
*Timer 3 clock enable.*
- [t\\_u32 TIM4EN](#): 1  
*Timer 4 clock enable.*
- [t\\_u32 TIM5EN](#): 1  
*Timer 5 clock enable.*
- [t\\_u32 TIM6EN](#): 1  
*Timer 6 clock enable.*
- [t\\_u32 TIM7EN](#): 1  
*Timer 7 clock enable.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 5  
*Reserved bit(s)*
- [t\\_u32 WWDGEN](#): 1  
*Window watchdog clock enable.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 2  
*Reserved bit(s)*
- [t\\_u32 SPI2EN](#): 1  
*SPI 2 clock enable.*
- [t\\_u32 SPI3EN](#): 1  
*SPI 3 clock enable.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 USART2EN](#): 1  
*USART 2 clock enable.*
- [t\\_u32 USART3EN](#): 1  
*USART 3 clock enable.*
- [t\\_u32 UART4EN](#): 1  
*UART 4 clock enable.*
- [t\\_u32 UART5EN](#): 1  
*UART 5 clock enable.*
- [t\\_u32 I2C1EN](#): 1  
*I2C 1 clock enable.*
- [t\\_u32 I2C2EN](#): 1  
*I2C 2 clock enable.*
- [t\\_u32 USBEN](#): 1  
*USB clock enable.*
- [t\\_u32 \\_\\_pad3\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 CANEN](#): 1  
*CAN clock enable.*
- [t\\_u32 \\_\\_pad4\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 BKPEN](#): 1  
*Backup interface clock enable.*
- [t\\_u32 PWREN](#): 1  
*Power interface clock enable.*
- [t\\_u32 DACEN](#): 1  
*DAC interface clock enable.*
- [t\\_u32 \\_\\_pad5\\_\\_](#): 2  
*Reserved bit(s)*

### 7.3.1 Detailed Description

RCC APB1 peripheral clock enable register.

### 7.3.2 Field Documentation

#### 7.3.2.1 TIM2EN

`t_u32 TIM2EN`

Timer 2 clock enable.

This field enables the Timer 2 clock

#### 7.3.2.2 TIM3EN

`t_u32 TIM3EN`

Timer 3 clock enable.

This field enables the Timer 3 clock

#### 7.3.2.3 TIM4EN

`t_u32 TIM4EN`

Timer 4 clock enable.

This field enables the Timer 4 clock

#### 7.3.2.4 TIM5EN

`t_u32 TIM5EN`

Timer 5 clock enable.

This field enables the Timer 5 clock

#### 7.3.2.5 TIM6EN

`t_u32 TIM6EN`

Timer 6 clock enable.

This field enables the Timer 6 clock

### 7.3.2.6 TIM7EN

t\_u32 TIM7EN

Timer 7 clock enable.

This field enables the Timer 7 clock

### 7.3.2.7 \_\_pad0\_\_

t\_u32 \_\_pad0\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.3.2.8 WWDGEN

t\_u32 WWDGEN

Window watchdog clock enable.

This field enables the Window watchdog clock

### 7.3.2.9 \_\_pad1\_\_

t\_u32 \_\_pad1\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.3.2.10 SPI2EN

t\_u32 SPI2EN

SPI 2 clock enable.

This field enables the SPI 2 clock

#### 7.3.2.11 SPI3EN

`t_u32` SPI3EN

SPI 3 clock enable.

This field enables the SPI 3 clock

#### 7.3.2.12 \_\_pad2\_\_

`t_u32` \_\_pad2\_\_

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.3.2.13 USART2EN

`t_u32` USART2EN

USART 2 clock enable.

This field enables the USART 2 clock

#### 7.3.2.14 USART3EN

`t_u32` USART3EN

USART 3 clock enable.

This field enables the USART 3 clock

#### 7.3.2.15 UART4EN

`t_u32` UART4EN

UART 4 clock enable.

This field enables the UART 4 clock

#### 7.3.2.16 UART5EN

`t_u32` UART5EN

UART 5 clock enable.

This field enables the UART 5 clock

#### 7.3.2.17 I2C1EN

t\_u32 I2C1EN

I2C 1 clock enable.

This field enables the I2C 1 clock

#### 7.3.2.18 I2C2EN

t\_u32 I2C2EN

I2C 2 clock enable.

This field enables the I2C 2 clock

#### 7.3.2.19 USBEN

t\_u32 USBEN

USB clock enable.

This field enables the USB clock

#### 7.3.2.20 \_\_pad3\_\_

t\_u32 \_\_pad3\_\_

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.3.2.21 CANEN

t\_u32 CANEN

CAN clock enable.

This field enables the CAN clock

**7.3.2.22 \_\_pad4\_\_**

t\_u32 \_\_pad4\_\_

Reserved bit(s)

**Attention**

This field is reserved and must be kept at reset value.

**7.3.2.23 BKPEN**

t\_u32 BKPEN

Backup interface clock enable.

This field enables the Backup interface clock

**7.3.2.24 PWREN**

t\_u32 PWREN

Power interface clock enable.

This field enables the Power interface clock

**7.3.2.25 DACEN**

t\_u32 DACEN

DAC interface clock enable.

This field enables the DAC interface clock

**7.3.2.26 \_\_pad5\_\_**

t\_u32 \_\_pad5\_\_

Reserved bit(s)

**Attention**

This field is reserved and must be kept at reset value.

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

- MCAL/RCC/[RCC\\_private.h](#)

## 7.4 t\_RCC\_APB1RSTR Struct Reference

RCC APB1 peripheral reset register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_APB1RSTR:

### Data Fields

- [t\\_u32 TIM2RST](#): 1  
*TIM2 timer reset.*
- [t\\_u32 TIM3RST](#): 1  
*TIM3 timer reset.*
- [t\\_u32 TIM4RST](#): 1  
*TIM4 timer reset.*
- [t\\_u32 TIM5RST](#): 1  
*TIM5 timer reset.*
- [t\\_u32 TIM6RST](#): 1  
*TIM6 timer reset.*
- [t\\_u32 TIM7RST](#): 1  
*TIM7 timer reset.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 5  
*Reserved bit(s)*
- [t\\_u32 WWDGRST](#): 1  
*Window watchdog reset.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 2  
*Reserved bit(s)*
- [t\\_u32 SPI2RST](#): 1  
*SPI 2 reset.*
- [t\\_u32 SPI3RST](#): 1  
*SPI 3 reset.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 USART2RST](#): 1  
*USART 2 reset.*
- [t\\_u32 USART3RST](#): 1  
*USART 3 reset.*
- [t\\_u32 UART4RST](#): 1  
*UART 4 reset.*
- [t\\_u32 UART5RST](#): 1  
*UART 5 reset.*
- [t\\_u32 I2C1RST](#): 1  
*I2C 1 reset.*
- [t\\_u32 I2C2RST](#): 1  
*I2C 2 reset.*
- [t\\_u32 USBRST](#): 1  
*USB interface reset.*
- [t\\_u32 \\_\\_pad3\\_\\_](#): 1  
*Reserved bit(s)*

- [t\\_u32 CANRST](#): 1  
*CAN reset.*
- [t\\_u32 \\_\\_pad4\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 BKPRST](#): 1  
*Backup interface reset.*
- [t\\_u32 PWRRST](#): 1  
*Power interface reset.*
- [t\\_u32 DACRST](#): 1  
*DAC interface reset.*
- [t\\_u32 \\_\\_pad5\\_\\_](#): 2  
*Reserved bit(s)*

### 7.4.1 Detailed Description

RCC APB1 peripheral reset register.

### 7.4.2 Field Documentation

#### 7.4.2.1 TIM2RST

[t\\_u32](#) TIM2RST

TIM2 timer reset.

This field resets the TIM2 configuration registers

#### 7.4.2.2 TIM3RST

[t\\_u32](#) TIM3RST

TIM3 timer reset.

This field resets the TIM3 configuration registers

#### 7.4.2.3 TIM4RST

[t\\_u32](#) TIM4RST

TIM4 timer reset.

This field resets the TIM4 configuration registers



#### 7.4.2.4 TIM5RST

`t_u32` TIM5RST

TIM5 timer reset.

This field resets the TIM5 configuration registers

#### 7.4.2.5 TIM6RST

`t_u32` TIM6RST

TIM6 timer reset.

This field resets the TIM6 configuration registers

#### 7.4.2.6 TIM7RST

`t_u32` TIM7RST

TIM7 timer reset.

This field resets the TIM7 configuration registers

#### 7.4.2.7 \_\_pad0\_\_

`t_u32` \_\_pad0\_\_

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.4.2.8 WWDGRST

`t_u32` WWDGRST

Window watchdog reset.

This field resets the window watchdog configuration registers

#### 7.4.2.9 \_\_pad1\_\_

`t_u32 __pad1__`

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.4.2.10 SPI2RST

`t_u32 SPI2RST`

SPI 2 reset.

This field resets the SPI 2 configuration registers

#### 7.4.2.11 SPI3RST

`t_u32 SPI3RST`

SPI 3 reset.

This field resets the SPI 3 configuration registers

#### 7.4.2.12 \_\_pad2\_\_

`t_u32 __pad2__`

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.4.2.13 USART2RST

`t_u32 USART2RST`

USART 2 reset.

This field resets the USART 2 configuration registers

#### 7.4.2.14 USART3RST

`t_u32` USART3RST

USART 3 reset.

This field resets the USART 3 configuration registers

#### 7.4.2.15 UART4RST

`t_u32` UART4RST

UART 4 reset.

This field resets the UART 4 configuration registers

#### 7.4.2.16 UART5RST

`t_u32` UART5RST

UART 5 reset.

This field resets the UART 5 configuration registers

#### 7.4.2.17 I2C1RST

`t_u32` I2C1RST

I2C 1 reset.

This field resets the I2C 1 configuration registers

#### 7.4.2.18 I2C2RST

`t_u32` I2C2RST

I2C 2 reset.

This field resets the I2C 2 configuration registers

#### 7.4.2.19 USBRST

`t_u32` USBRST

USB interface reset.

This field resets the USB interface configuration registers

**7.4.2.20 \_\_pad3\_\_**

`t_u32 __pad3__`

Reserved bit(s)

**Attention**

This field is reserved and must be kept at reset value.

**7.4.2.21 CANRST**

`t_u32 CANRST`

CAN reset.

This field resets the CAN configuration registers

**7.4.2.22 \_\_pad4\_\_**

`t_u32 __pad4__`

Reserved bit(s)

**Attention**

This field is reserved and must be kept at reset value.

**7.4.2.23 BKPRST**

`t_u32 BKPRST`

Backup interface reset.

This field resets the backup interface configuration registers

**7.4.2.24 PWRRST**

`t_u32 PWRRST`

Power interface reset.

This field resets the power interface configuration registers

#### 7.4.2.25 DACRST

[t\\_u32](#) DACRST

DAC interface reset.

This field resets the DAC interface configuration registers

#### 7.4.2.26 \_\_pad5\_\_

[t\\_u32](#) \_\_pad5\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- MCAL/RCC/[RCC\\_private.h](#)

## 7.5 t\_RCC\_APB2ENR Struct Reference

RCC APB2 peripheral clock enable register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_APB2ENR:

### Data Fields

- [t\\_u32 AFIOEN](#): 1  
*Alternate function I/O clock enable.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 IOPAEN](#): 1  
*I/O port A clock enable.*
- [t\\_u32 IOPBEN](#): 1  
*I/O port B clock enable.*
- [t\\_u32 IOPCEN](#): 1  
*I/O port C clock enable.*
- [t\\_u32 IOPDEN](#): 1  
*I/O port D clock enable.*
- [t\\_u32 IOPEEN](#): 1  
*I/O port E clock enable.*
- [t\\_u32 IOPFEN](#): 1  
*I/O port F clock enable.*

- [t\\_u32 IOPGEN](#): 1  
*I/O port G clock enable.*
- [t\\_u32 ADC1EN](#): 1  
*ADC 1 interface clock enable.*
- [t\\_u32 ADC2EN](#): 1  
*ADC 2 interface clock enable.*
- [t\\_u32 TIM1EN](#): 1  
*TIM1 Timer clock enable.*
- [t\\_u32 SPI1EN](#): 1  
*SPI 1 clock enable.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 USART1EN](#): 1  
*USART1 clock enable.*
- [t\\_u32 ADC3EN](#): 1  
*ADC 3 interface clock enable.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 16  
*Reserved bit(s)*

### 7.5.1 Detailed Description

RCC APB2 peripheral clock enable register.

### 7.5.2 Field Documentation

#### 7.5.2.1 AFIOEN

[t\\_u32 AFIOEN](#)

Alternate function I/O clock enable.

This field enables the alternate function I/O clock

#### 7.5.2.2 \_\_pad0\_\_

[t\\_u32 \\_\\_pad0\\_\\_](#)

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.5.2.3 IOPAEN

`t_u32` IOPAEN

I/O port A clock enable.

This field enables the I/O port A clock

### 7.5.2.4 IOPBEN

`t_u32` IOPBEN

I/O port B clock enable.

This field enables the I/O port B clock

### 7.5.2.5 IOPCEN

`t_u32` IOPCEN

I/O port C clock enable.

This field enables the I/O port C clock

### 7.5.2.6 IOPDEN

`t_u32` IOPDEN

I/O port D clock enable.

This field enables the I/O port D clock

### 7.5.2.7 IOPEEN

`t_u32` IOPEEN

I/O port E clock enable.

This field enables the I/O port E clock

### 7.5.2.8 IOPFEN

`t_u32` IOPFEN

I/O port F clock enable.

This field enables the I/O port F clock

### 7.5.2.9 IOPGEN

`t_u32 IOPGEN`

I/O port G clock enable.

This field enables the I/O port G clock

### 7.5.2.10 ADC1EN

`t_u32 ADC1EN`

ADC 1 interface clock enable.

This field enables the ADC 1 interface clock

### 7.5.2.11 ADC2EN

`t_u32 ADC2EN`

ADC 2 interface clock enable.

This field enables the ADC 2 interface clock

### 7.5.2.12 TIM1EN

`t_u32 TIM1EN`

TIM1 Timer clock enable.

This field enables the TIM1 Timer clock

### 7.5.2.13 SPI1EN

`t_u32 SPI1EN`

SPI 1 clock enable.

This field enables the SPI 1 clock

### 7.5.2.14 \_\_pad1\_\_

`t_u32 __pad1__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.



### 7.5.2.15 USART1EN

t\_u32 USART1EN

USART1 clock enable.

This field enables the USART1 clock

### 7.5.2.16 ADC3EN

t\_u32 ADC3EN

ADC 3 interface clock enable.

This field enables the ADC 3 interface clock

### 7.5.2.17 \_\_pad2\_\_

t\_u32 \_\_pad2\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- MCAL/RCC/RCC\_private.h

## 7.6 t\_RCC\_APB2RSTR Struct Reference

RCC APB2 peripheral reset register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_APB2RSTR:

## Data Fields

- [t\\_u32 AFIORST](#): 1  
*Alternate function I/O reset.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 IOPARST](#): 1  
*IO port A reset.*
- [t\\_u32 IOPBRST](#): 1  
*IO port B reset.*
- [t\\_u32 IOPCRST](#): 1  
*IO port C reset.*
- [t\\_u32 IOPDRST](#): 1  
*IO port D reset.*
- [t\\_u32 IOPERST](#): 1  
*IO port E reset.*
- [t\\_u32 IOPFRST](#): 1  
*IO port F reset.*
- [t\\_u32 IOPGRST](#): 1  
*IO port G reset.*
- [t\\_u32 ADC1RST](#): 1  
*ADC 1 interface reset.*
- [t\\_u32 ADC2RST](#): 1  
*ADC 2 interface reset.*
- [t\\_u32 TIM1RST](#): 1  
*TIM1 timer reset.*
- [t\\_u32 SPI1RST](#): 1  
*SPI 1 reset.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 USART1RST](#): 1  
*USART1 reset.*
- [t\\_u32 ADC3RST](#): 1  
*ADC 3 interface reset.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 16  
*Reserved bit(s)*

### 7.6.1 Detailed Description

RCC APB2 peripheral reset register.

### 7.6.2 Field Documentation

### 7.6.2.1 AFIORST

`t_u32` AFIORST

Alternate function I/O reset.

This field resets the IO port configuration registers

### 7.6.2.2 \_\_pad0\_\_

`t_u32` \_\_pad0\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.6.2.3 IOPARST

`t_u32` IOPARST

IO port A reset.

This field resets the IO port A configuration registers

### 7.6.2.4 IOPBRST

`t_u32` IOPBRST

IO port B reset.

This field resets the IO port B configuration registers

### 7.6.2.5 IOPCRST

`t_u32` IOPCRST

IO port C reset.

This field resets the IO port C configuration registers

### 7.6.2.6 IOPDRST

`t_u32` IOPDRST

IO port D reset.

This field resets the IO port D configuration registers

#### 7.6.2.7 IOPERST

`t_u32 IOPERST`

IO port E reset.

This field resets the IO port E configuration registers

#### 7.6.2.8 IOPFRST

`t_u32 IOPFRST`

IO port F reset.

This field resets the IO port F configuration registers

#### 7.6.2.9 IOPGRST

`t_u32 IOPGRST`

IO port G reset.

This field resets the IO port G configuration registers

#### 7.6.2.10 ADC1RST

`t_u32 ADC1RST`

ADC 1 interface reset.

This field resets the ADC 1 configuration registers

#### 7.6.2.11 ADC2RST

`t_u32 ADC2RST`

ADC 2 interface reset.

This field resets the ADC 2 configuration registers

#### 7.6.2.12 TIM1RST

`t_u32 TIM1RST`

TIM1 timer reset.

This field resets the TIM1 configuration registers

### 7.6.2.13 SPI1RST

`t_u32 SPI1RST`

SPI 1 reset.

This field resets the SPI 1 configuration registers

### 7.6.2.14 \_\_pad1\_\_

`t_u32 __pad1__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.6.2.15 USART1RST

`t_u32 USART1RST`

USART1 reset.

This field resets the USART1 configuration registers

### 7.6.2.16 ADC3RST

`t_u32 ADC3RST`

ADC 3 interface reset.

This field resets the ADC 3 configuration registers

### 7.6.2.17 \_\_pad2\_\_

`t_u32 __pad2__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- [MCAL/RCC/RCC\\_private.h](#)

## 7.7 t\_RCC\_BDCR Struct Reference

RCC Backup domain control register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_BDCR:

### Data Fields

- [t\\_u32 LSEON](#): 1  
*External Low Speed oscillator enable.*
- [t\\_u32 LSERDY](#): 1  
*External Low Speed oscillator ready.*
- [t\\_u32 LSEBYP](#): 1  
*External Low Speed oscillator bypass.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 5  
*Reserved bit(s)*
- [t\\_u32 RTCSEL](#): 2  
*RTC clock source selection.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 5  
*Reserved bit(s)*
- [t\\_u32 RTCEN](#): 1  
*RTC clock enable.*
- [t\\_u32 BDRST](#): 1  
*Backup domain software reset.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 15  
*Reserved bit(s)*

### 7.7.1 Detailed Description

RCC Backup domain control register.

### 7.7.2 Field Documentation

#### 7.7.2.1 LSEON

[t\\_u32 LSEON](#)

External Low Speed oscillator enable.

This field enables the external Low Speed oscillator (LSE)

### 7.7.2.2 LSERDY

t\_u32 LSERDY

External Low Speed oscillator ready.

This field is set and cleared by hardware to indicate when the external Low Speed oscillator (LSE) is stable.

#### Warning

This field is read-only

### 7.7.2.3 LSEBYP

t\_u32 LSEBYP

External Low Speed oscillator bypass.

This field is used to bypass the oscillator with an external clock.

### 7.7.2.4 \_\_pad0\_\_

t\_u32 \_\_pad0\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.7.2.5 RTCSEL

t\_u32 RTCSEL

RTC clock source selection.

This field selects the clock source to be used for the RTC.

### 7.7.2.6 \_\_pad1\_\_

t\_u32 \_\_pad1\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.7.2.7 RTCEN

`t_u32` RTCEN

RTC clock enable.

This field enables the RTC clock

### 7.7.2.8 BDRST

`t_u32` BDRST

Backup domain software reset.

This field forces the Backup domain to reset.

### 7.7.2.9 \_\_pad2\_\_

`t_u32` \_\_pad2\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- [MCAL/RCC/RCC\\_private.h](#)

## 7.8 t\_RCC\_CFGR Struct Reference

RCC clock configuration register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_CFGR:



## Data Fields

- [t\\_u32 SW](#): 2  
*System clock switch.*
- [t\\_u32 SWS](#): 2  
*System clock switch status.*
- [t\\_u32 HPRE](#): 4  
*AHB prescaler.*
- [t\\_u32 PPRE1](#): 3  
*APB Low speed prescaler (APB1)*
- [t\\_u32 PPRE2](#): 3  
*APB high-speed prescaler (APB2)*
- [t\\_u32 ADCPRE](#): 2  
*ADC prescaler.*
- [t\\_u32 PLLSRC](#): 1  
*PLL entry/input clock source.*
- [t\\_u32 PLLXTPRE](#): 1  
*LSB of PLL division factor PREDIV1.*
- [t\\_u32 PLLMUL](#): 4  
*PLL multiplication factor.*
- [t\\_u32 USBPRE](#): 1  
*USB prescaler.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 1  
*Reserved bit(s)*
- [t\\_u32 MCO](#): 3  
*Microcontroller clock output.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 5  
*Reserved bit(s)*

### 7.8.1 Detailed Description

RCC clock configuration register.

### 7.8.2 Field Documentation

#### 7.8.2.1 SW

[t\\_u32 SW](#)

System clock switch.

Select the system clock source

### 7.8.2.2 SWS

`t_u32` SWS

System clock switch status.

These bits are set and cleared by hardware to inform about which clock source is used as the system clock

#### Warning

This field is read-only

### 7.8.2.3 HPRE

`t_u32` HPRE

AHB prescaler.

Configure the AHB clock (HCLK) prescaler

### 7.8.2.4 PPRE1

`t_u32` PPRE1

APB Low speed prescaler (APB1)

Configure the APB1 clock (PCLK1) prescaler

#### Warning

Max frequency for APB1 is 36 MHz

### 7.8.2.5 PPRE2

`t_u32` PPRE2

APB high-speed prescaler (APB2)

Configure the APB2 clock (PCLK2) prescaler

#### Warning

Max frequency for APB2 is 72 MHz

### 7.8.2.6 ADCPRE

`t_u32` ADCPRE

ADC prescaler.

Configure the ADC clock (ADCCLK) prescaler

#### Warning

Max frequency for ADC is 14 MHz

### 7.8.2.7 PLLSRC

`t_u32` PLLSRC

PLL entry/input clock source.

Select the PLL input clock source

### 7.8.2.8 PLLXTPRE

`t_u32` PLLXTPRE

LSB of PLL division factor PREDIV1.

Select the LSB of the division factor for PLL input clock

### 7.8.2.9 PLLMUL

`t_u32` PLLMUL

PLL multiplication factor.

Configure the PLL multiplication factor

### 7.8.2.10 USBPRE

`t_u32` USBPRE

USB prescaler.

Configure the USB clock (USBCLK) prescaler

### 7.8.2.11 \_\_pad0\_\_

`t_u32 __pad0__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.8.2.12 MCO

`t_u32 MCO`

Microcontroller clock output.

Configure the MCO clock

### 7.8.2.13 \_\_pad1\_\_

`t_u32 __pad1__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- [MCAL/RCC/RCC\\_private.h](#)

## 7.9 t\_RCC\_CIR Struct Reference

RCC clock interrupt register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for `t_RCC_CIR`:

## Data Fields

- [t\\_u32 LSIRDYF](#): 1  
*LSI ready interrupt flag.*
- [t\\_u32 LSERDYF](#): 1  
*LSE ready interrupt flag.*
- [t\\_u32 HSIRDYF](#): 1  
*HSI ready interrupt flag.*
- [t\\_u32 HSERDYF](#): 1  
*HSE ready interrupt flag.*
- [t\\_u32 PLLRDYF](#): 1  
*PLL ready interrupt flag.*
- [t\\_u32 \\_\\_pad0\\_\\_](#): 2  
*Reserved bit(s)*
- [t\\_u32 CSSF](#): 1  
*Clock security system interrupt flag.*
- [t\\_u32 LSIRDYIE](#): 1  
*LSI ready interrupt enable.*
- [t\\_u32 LSERDYIE](#): 1  
*LSE ready interrupt enable.*
- [t\\_u32 HSIRDYIE](#): 1  
*HSI ready interrupt enable.*
- [t\\_u32 HSERDYIE](#): 1  
*HSE ready interrupt enable.*
- [t\\_u32 PLLRDYIE](#): 1  
*PLL ready interrupt enable.*
- [t\\_u32 \\_\\_pad1\\_\\_](#): 3  
*Reserved bit(s)*
- [t\\_u32 LSIRDYC](#): 1  
*LSI ready interrupt clear.*
- [t\\_u32 LSERDYC](#): 1  
*LSE ready interrupt clear.*
- [t\\_u32 HSIRDYC](#): 1  
*HSI ready interrupt clear.*
- [t\\_u32 HSERDYC](#): 1  
*HSE ready interrupt clear.*
- [t\\_u32 PLLRDYC](#): 1  
*PLL ready interrupt clear.*
- [t\\_u32 \\_\\_pad2\\_\\_](#): 2  
*Reserved bit(s)*
- [t\\_u32 CSSC](#): 1  
*Clock security system interrupt clear.*
- [t\\_u32 \\_\\_pad3\\_\\_](#): 8  
*Reserved bit(s)*

### 7.9.1 Detailed Description

RCC clock interrupt register.

## 7.9.2 Field Documentation

### 7.9.2.1 LSIRDYF

`t_u32` LSIRDYF

LSI ready interrupt flag.

This field indicates whether the LSI oscillator is stable or not

#### Warning

This field is read-only

### 7.9.2.2 LSERDYF

`t_u32` LSERDYF

LSE ready interrupt flag.

This field indicates whether the LSE oscillator is stable or not

#### Warning

This field is read-only

### 7.9.2.3 HSIRDYF

`t_u32` HSIRDYF

HSI ready interrupt flag.

This field indicates whether the HSI oscillator is stable or not

#### Warning

This field is read-only

#### 7.9.2.4 HSERDYF

t\_u32 HSERDYF

HSE ready interrupt flag.

This field indicates whether the HSE oscillator is stable or not

##### Warning

This field is read-only

#### 7.9.2.5 PLLRDYF

t\_u32 PLLRDYF

PLL ready interrupt flag.

This field indicates whether the PLL oscillator is stable or not

##### Warning

This field is read-only

#### 7.9.2.6 \_\_pad0\_\_

t\_u32 \_\_pad0\_\_

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.9.2.7 CSSF

t\_u32 CSSF

Clock security system interrupt flag.

This field indicates whether a failure is detected on the external 3.3 V oscillator

##### Warning

This field is read-only

#### 7.9.2.8 LSIRDYIE

`t_u32` LSIRDYIE

LSI ready interrupt enable.

This field enables the LSI ready interrupt

#### 7.9.2.9 LSERDYIE

`t_u32` LSERDYIE

LSE ready interrupt enable.

This field enables the LSE ready interrupt

#### 7.9.2.10 HSIRDYIE

`t_u32` HSIRDYIE

HSI ready interrupt enable.

This field enables the HSI ready interrupt

#### 7.9.2.11 HSERDYIE

`t_u32` HSERDYIE

HSE ready interrupt enable.

This field enables the HSE ready interrupt

#### 7.9.2.12 PLLRDYIE

`t_u32` PLLRDYIE

PLL ready interrupt enable.

This field enables the PLL ready interrupt

#### 7.9.2.13 \_\_pad1\_\_

`t_u32` \_\_pad1\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.



#### 7.9.2.14 LSIRDYC

`t_u32` LSIRDYC

LSI ready interrupt clear.

This field clears the LSI ready interrupt

#### 7.9.2.15 LSERDYC

`t_u32` LSERDYC

LSE ready interrupt clear.

This field clears the LSE ready interrupt

#### 7.9.2.16 HSIRDYC

`t_u32` HSIRDYC

HSI ready interrupt clear.

This field clears the HSI ready interrupt

#### 7.9.2.17 HSERDYC

`t_u32` HSERDYC

HSE ready interrupt clear.

This field clears the HSE ready interrupt

#### 7.9.2.18 PLLRDYC

`t_u32` PLLRDYC

PLL ready interrupt clear.

This field clears the PLL ready interrupt

#### 7.9.2.19 \_\_pad2\_\_

`t_u32` \_\_pad2\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.9.2.20 CSSC

`t_u32` CSSC

Clock security system interrupt clear.

This field clears the clock security system interrupt

### 7.9.2.21 \_\_pad3\_\_

`t_u32` \_\_pad3\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- MCAL/RCC/RCC\_private.h

## 7.10 t\_RCC\_CR Struct Reference

RCC clock control register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_CR:

### Data Fields

- `t_u32` HSION: 1  
*Internal high-speed clock enable.*
- `t_u32` HSIRDY: 1  
*Internal high-speed clock ready flag.*
- `t_u32` \_\_pad0\_\_: 1  
*Reserved bit(s)*
- `t_u32` HSITRIM: 5  
*Internal high-speed clock trimming.*
- `t_u32` HSICAL: 8  
*Internal high-speed clock calibration.*
- `t_u32` HSEON: 1  
*HSE clock enable.*
- `t_u32` HSERDY: 1  
*External high-speed clock ready flag.*
- `t_u32` HSEBYP: 1  
*External high-speed clock bypass.*
- `t_u32` CSSON: 1  
*Clock security system enable.*
- `t_u32` \_\_pad1\_\_: 4  
*Reserved bit(s)*
- `t_u32` PLLON: 1  
*PLL enable.*
- `t_u32` PLLRDY: 1  
*PLL clock ready flag.*
- `t_u32` \_\_pad2\_\_: 6  
*Reserved bit(s)*

### 7.10.1 Detailed Description

RCC clock control register.

### 7.10.2 Field Documentation

#### 7.10.2.1 HSION

`t_u32` HSION

Internal high-speed clock enable.

Enable or disable the internal high-speed clock (HSI)

#### 7.10.2.2 HSIRDY

`t_u32` HSIRDY

Internal high-speed clock ready flag.

Whether the internal high-speed clock is ready or not

#### Warning

This field is read-only

#### 7.10.2.3 \_\_pad0\_\_

`t_u32` \_\_pad0\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

#### 7.10.2.4 HSITRIM

`t_u32` HSITRIM

Internal high-speed clock trimming.

Adjusts the internal high-speed clock (HSI) frequency in conjunction with the HSICAL field

##### Note

Default value: 16

#### 7.10.2.5 HSICAL

`t_u32` HSICAL

Internal high-speed clock calibration.

This field holds the factory calibration value

##### Warning

This field is read-only

#### 7.10.2.6 HSEON

`t_u32` HSEON

HSE clock enable.

Enable or disable the external high-speed clock (HSE)

#### 7.10.2.7 HSERDY

`t_u32` HSERDY

External high-speed clock ready flag.

Whether the external high-speed clock is ready or not

##### Warning

This field is read-only

### 7.10.2.8 HSEBYP

t\_u32 HSEBYP

External high-speed clock bypass.

Bypass the oscillator with an external clock

### 7.10.2.9 CSSON

t\_u32 CSSON

Clock security system enable.

Enable or disable the clock security system

### 7.10.2.10 \_\_pad1\_\_

t\_u32 \_\_pad1\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

### 7.10.2.11 PLLON

t\_u32 PLLON

PLL enable.

Enable or disable the PLL

### 7.10.2.12 PLLRDY

t\_u32 PLLRDY

PLL clock ready flag.

This bit is set by hardware to indicate that the PLL clock is locked

#### Warning

This field is read-only

### 7.10.2.13 \_\_pad2\_\_

`t_u32 __pad2__`

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

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

- MCAL/RCC/RCC\_private.h

## 7.11 t\_RCC\_CSR Struct Reference

RCC Control/status register.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_CSR:

### Data Fields

- `t_u32 LSION`: 1  
*Internal low speed oscillator enable.*
- `t_u32 LSIRDY`: 1  
*Internal low speed oscillator ready.*
- `t_u32 __pad0__`: 22  
*Reserved bit(s)*
- `t_u32 RMVF`: 1  
*Remove reset flag.*
- `t_u32 __pad1__`: 1  
*Reserved bit(s)*
- `t_u32 PINRSTF`: 1  
*Pin reset flag.*
- `t_u32 PORRSTF`: 1  
*POR/PDR reset flag.*
- `t_u32 SFTRSTF`: 1  
*Software reset flag.*
- `t_u32 IWDGRSTF`: 1  
*Independent watchdog reset flag.*
- `t_u32 WWDGRSTF`: 1  
*Window watchdog reset flag.*
- `t_u32 LPWRRSTF`: 1  
*Low-power reset flag.*

### 7.11.1 Detailed Description

RCC Control/status register.

### 7.11.2 Field Documentation

#### 7.11.2.1 LSION

`t_u32` LSION

Internal low speed oscillator enable.

This field enables the internal low speed oscillator (LSI)

#### 7.11.2.2 LSIRDY

`t_u32` LSIRDY

Internal low speed oscillator ready.

This field is set and cleared by hardware to indicate when the internal low speed oscillator (LSI) is stable.

#### Warning

This field is read-only

#### 7.11.2.3 \_\_pad0\_\_

`t_u32` \_\_pad0\_\_

Reserved bit(s)

#### Attention

This field is reserved and must be kept at reset value.

#### 7.11.2.4 RMVF

`t_u32` RMVF

Remove reset flag.

This field resets the reset flags.

#### 7.11.2.5 `__pad1__`

`t_u32 __pad1__`

Reserved bit(s)

##### Attention

This field is reserved and must be kept at reset value.

#### 7.11.2.6 `PINRSTF`

`t_u32 PINRSTF`

Pin reset flag.

This field indicates that a reset has been caused by an NRST pin reset

##### Warning

This field is read-only

#### 7.11.2.7 `PORRSTF`

`t_u32 PORRSTF`

POR/PDR reset flag.

This field indicates that a reset has been caused by a POR/PDR reset

##### Warning

This field is read-only

#### 7.11.2.8 `SFTRSTF`

`t_u32 SFTRSTF`

Software reset flag.

This field indicates that a reset has been caused by a software reset

##### Warning

This field is read-only



### 7.11.2.9 IWDGRSTF

`t_u32` IWDGRSTF

Independent watchdog reset flag.

This field indicates that a reset has been caused by an independent watchdog reset

#### Warning

This field is read-only

### 7.11.2.10 WWDGRSTF

`t_u32` WWDGRSTF

Window watchdog reset flag.

This field indicates that a reset has been caused by a window watchdog reset

#### Warning

This field is read-only

### 7.11.2.11 LPWRRSTF

`t_u32` LPWRRSTF

Low-power reset flag.

This field indicates that a reset has been caused by a low-power reset

#### Warning

This field is read-only

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

- [MCAL/RCC/RCC\\_private.h](#)

## 7.12 t\_RCC\_RegisterMap Struct Reference

RCC Register Map.

```
#include "MCAL/RCC/RCC_private.h"
```

Collaboration diagram for t\_RCC\_RegisterMap:

## Data Fields

- [t\\_RCC\\_CR](#) CR  
*Clock control register.*
- [t\\_RCC\\_CFGR](#) CFGR  
*Clock configuration register.*
- [t\\_RCC\\_CIR](#) CIR  
*Clock interrupt register.*
- [t\\_RCC\\_APB2RSTR](#) APB2RSTR  
*RCC APB2 peripheral reset register.*
- [t\\_RCC\\_APB1RSTR](#) APB1RSTR  
*RCC APB1 peripheral reset register.*
- [t\\_RCC\\_AHBENR](#) AHBENR  
*RCC AHB peripheral clock register.*
- [t\\_RCC\\_APB2ENR](#) APB2ENR  
*RCC APB2 peripheral clock enable register.*
- [t\\_RCC\\_APB1ENR](#) APB1ENR  
*RCC APB1 peripheral clock enable register.*
- [t\\_RCC\\_BDCR](#) BDCR  
*RCC Backup domain control register.*
- [t\\_RCC\\_CSR](#) CSR  
*RCC Control/status register.*

### 7.12.1 Detailed Description

RCC Register Map.

### 7.12.2 Field Documentation

#### 7.12.2.1 CR

[t\\_RCC\\_CR](#) CR

Clock control register.

#### 7.12.2.2 CFGR

[t\\_RCC\\_CFGR](#) CFGR

Clock configuration register.

### 7.12.2.3 CIR

[t\\_RCC\\_CIR](#) CIR

Clock interrupt register.

### 7.12.2.4 APB2RSTR

[t\\_RCC\\_APB2RSTR](#) APB2RSTR

RCC APB2 peripheral reset register.

### 7.12.2.5 APB1RSTR

[t\\_RCC\\_APB1RSTR](#) APB1RSTR

RCC APB1 peripheral reset register.

### 7.12.2.6 AHBENR

[t\\_RCC\\_AHBENR](#) AHBENR

RCC AHB peripheral clock register.

### 7.12.2.7 APB2ENR

[t\\_RCC\\_APB2ENR](#) APB2ENR

RCC APB2 peripheral clock enable register.

### 7.12.2.8 APB1ENR

[t\\_RCC\\_APB1ENR](#) APB1ENR

RCC APB1 peripheral clock enable register.

### 7.12.2.9 BDCR

[t\\_RCC\\_BDCR](#) BDCR

RCC Backup domain control register.

### 7.12.2.10 CSR

[t\\_RCC\\_CSR](#) CSR

RCC Control/status register.

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

- [MCAL/RCC/RCC\\_private.h](#)



## Chapter 8

# File Documentation

### 8.1 APPS/APPS\_main.c File Reference

This file contains the implementation of the main function that is responsible for running the applications.

```
#include "APPS_main.h"  
#include "../COTS/MCAL/RCC/RCC_interface.h"  
#include "TestingGPIO/TestApp_TestingGPIO_main.h"  
Include dependency graph for APPS_main.c:
```

#### Macros

- #define [TESTING\\_APPLICATION\\_MAIN\\_FUNC](#)(APP\_NAME) vTestApp\_##APP\_NAME##\_main()  
*This macro is used to define the main function of the application.*
- #define [TestingGPIO](#)

#### Functions

- void [vAPPS\\_main](#) (void)  
*This function is responsible for running the applications.*

#### 8.1.1 Detailed Description

This file contains the implementation of the main function that is responsible for running the applications.

##### Author

Mohamed Alaa

##### Version

1.0.0

##### Date

2023-06-16

## 8.1.2 Macro Definition Documentation

### 8.1.2.1 TESTING\_APPLICATION\_MAIN\_FUNC

```
#define TESTING_APPLICATION_MAIN_FUNC (
    APP_NAME ) vTestApp_##APP_NAME##_main()
```

This macro is used to define the main function of the application.

This macro is used to define the main function of the application

#### Parameters

in	APP_NAME	The name of the application
----	----------	-----------------------------

### 8.1.2.2 TestingGPIO

```
#define TestingGPIO
```

## 8.2 APPS/APPS\_main.h File Reference

This file contains the prototypes of the main function of the testing applications.

This graph shows which files directly or indirectly include this file:

### Functions

- void [vAPPS\\_main](#) (void)  
*This function is responsible for running the applications.*

### 8.2.1 Detailed Description

This file contains the prototypes of the main function of the testing applications.

#### Author

Mohamed Alaa

This file contains the prototypes of the main function that is responsible for running the applications

#### Version

1.0.0

#### Date

2023-06-16

## 8.3 /github/workspace/README.md File Reference

## 8.4 APPS/README.md File Reference

## 8.5 APPS/TestingGPIO/TestApp\_TestingGPIO\_main.c File Reference

This file contains the main implementation for the TestApp\_TestingGPIO application.

```
#include "TestApp_TestingGPIO_main.h"
#include "../../COTS/LIB/LSTD_TYPES.h"
#include "../../COTS/MCAL/RCC/RCC_interface.h"
#include "../../COTS/MCAL/GPIO/GPIO_interface.h"
Include dependency graph for TestApp_TestingGPIO_main.c:
```

### Functions

- void [vTestApp\\_TestingGPIO\\_main](#) (void)  
*This function is the main function for the TestApp\_TestingGPIO application.*

### 8.5.1 Detailed Description

This file contains the main implementation for the TestApp\_TestingGPIO application.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the main implementation for the TestApp\_TestingGPIO application.

#### Date

2023-06-25

## 8.6 APPS/TestingGPIO/TestApp\_TestingGPIO\_main.h File Reference

This file contains the main header file for the TestApp\_TestingGPIO application.

This graph shows which files directly or indirectly include this file:

### Functions

- void [vTestApp\\_TestingGPIO\\_main](#) (void)  
*This function is the main function for the TestApp\_TestingGPIO application.*

### 8.6.1 Detailed Description

This file contains the main header file for the TestApp\_TestingGPIO application.

**Author**

Mohamed Alaa

**Version**

1.0.0

This file contains the main header file for the TestApp\_TestingGPIO application.

**Date**

2023-06-25

## 8.7 LIB/LSTD\_BITMATH.h File Reference

This file contains the bit math manipulation macro-functions.

This graph shows which files directly or indirectly include this file:

### Macros

- `#define SET_BIT(REG, BITNUM) (REG) |= (1 << (BITNUM))`  
*Set a certain bit's value.*
- `#define CLEAR_BIT(REG, BITNUM) (REG) &= ~(1 << (BITNUM))`  
*Clear a certain bit's value to.*
- `#define TOGGLE_BIT(REG, BITNUM) (REG) ^= (1 << (BITNUM))`  
*Toggle a bit to 0 if it's 1, 1 otherwise.*
- `#define GET_BIT(REG, BITNUM) (((REG) >> (BITNUM)) & 1)`  
*Return the value of the bit whether it's 1 or 0*

### 8.7.1 Detailed Description

This file contains the bit math manipulation macro-functions.

**Author**

Mohamed alaa

**Version**

1.0.0

**Date**

2023-06-18



## 8.8 LIB/LSTD\_COMPILER.h File Reference

This file contains the compiler standard macros.

This graph shows which files directly or indirectly include this file:

### Macros

- #define `CONST` `const`  
*Declare a standard constant variable with the specified type.*
- #define `STATIC` `static`  
*Declare a standard static variable/function.*
- #define `VOLATILE` `volatile`  
*Declare a standard volatile variable.*
- #define `P2VAR`(ptrtype) ptrtype \*  
*Declare a pointer-to-variable with the specified type.*
- #define `P2CONST`(ptrtype) `CONST` ptrtype \*  
*Declare a constant pointer-to-variable with the specified type.*
- #define `CONSTP2VAR`(ptrtype) ptrtype \*`CONST`  
*Declare a pointer-to-variable constant with the specified type.*
- #define `CONSTP2CONST`(ptrtype) `CONST` ptrtype \*`CONST`  
*Declare a constant pointer-to-variable constant with the specified type.*
- #define `P2FUNC`(rettype, fctname) rettype(\*fctname)  
*Declare a pointer-to-function with the specified return type.*

### 8.8.1 Detailed Description

This file contains the compiler standard macros.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-18

## 8.9 LIB/LSTD\_HW\_REGS.h File Reference

This file contains the hardware registers macro-functions for memory addresses mapping and accessing.

```
#include "LSTD_TYPES.h"
```

```
#include "LSTD_COMPILER.h"
```

Include dependency graph for LSTD\_HW\_REGS.h: This graph shows which files directly or indirectly include this file:

## Macros

- #define `REGISTER_ADDRESS`(ADDRESS, OFFSET) ((ADDRESS) + (OFFSET))  
*Placeholder for declaring a register address.*
- #define `REGISTER`(REG\_TYPE, ADDRESS) (\*(`VOLATILE P2VAR`(REG\_TYPE))(ADDRESS))  
*Map to a certain register by its address in the memory.*
- #define `REGISTER_U8`(ADDRESS) `REGISTER`(`t_u8`, ADDRESS)  
*Map to a certain register by its 8-bit address in the memory (used for 8-bit registers)*
- #define `REGISTER_U16`(ADDRESS) `REGISTER`(`t_u16`, ADDRESS)  
*Map to a certain register by its 16-bit address in the memory (used for 16-bit registers)*
- #define `REGISTER_U32`(ADDRESS) `REGISTER`(`t_u32`, ADDRESS)  
*Map to a certain register by its 32-bit address in the memory (used for 32-bit registers)*

### 8.9.1 Detailed Description

This file contains the hardware registers macro-functions for memory addresses mapping and accessing.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-18

## 8.10 LIB/LSTD\_SHARED.h File Reference

Shared library for all the standard types.

```
#include "LSTD_TYPES.h"
```

Include dependency graph for LSTD\_SHARED.h: This graph shows which files directly or indirectly include this file:

## Macros

- #define `SET_REGISTER_BIT_STATUS`(PERPH, REG, BIT, BSTATUS) (PERPH.REG.BIT = (BSTATUS == `TRUE`) ? 1U : 0U)  
*Set register bit status (1 or 0)*

### 8.10.1 Detailed Description

Shared library for all the standard types.

#### Author

Mohamed Alaa

#### Version

1.0.0

#### Date

2023-06-24

## 8.11 LIB/LSTD\_TYPES.h File Reference

This file contains the standard data types.

This graph shows which files directly or indirectly include this file:

### Typedefs

- typedef unsigned char [t\\_bool](#)  
*Type definition for boolean.*
- typedef unsigned char [t\\_u8](#)  
*Type definition for 8-bit unsigned INT.*
- typedef signed char [t\\_s8](#)  
*Type definition for 8-bit signed INT.*
- typedef unsigned char [t\\_c8](#)  
*Type definition for 8-bit char.*
- typedef unsigned short int [t\\_u16](#)  
*Type definition for 16-bit unsigned int.*
- typedef signed short int [t\\_s16](#)  
*Type definition for 16-bit signed INT.*
- typedef unsigned int [t\\_u32](#)  
*Type definition for 32-bit unsigned int.*
- typedef signed int [t\\_s32](#)  
*Type definition for 32-bit signed INT.*
- typedef unsigned long int [t\\_u64](#)  
*Type definition for 64-bit unsigned int.*
- typedef signed long int [t\\_s64](#)  
*Type definition for 64-bit signed int.*
- typedef float [t\\_fl32](#)  
*Type definition for 32-bit float.*
- typedef double [t\\_fl64](#)  
*Type definition for 64-bit float.*

### 8.11.1 Detailed Description

This file contains the standard data types.

**Author**

Mohamed Alaa

**Version**

1.0.0

**Date**

2023-06-18

## 8.12 LIB/LSTD\_VALUES.h File Reference

This file contains the standard values.

```
#include "LSTD_TYPES.h"  
#include "LSTD_COMPILER.h"
```

Include dependency graph for LSTD\_VALUES.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define TRUE ((t_bool)1)`  
*Type definition for TRUE.*
- `#define FALSE ((t_bool)0)`  
*Type definition for FALSE.*
- `#define NULL ((P2VAR(void))0)`  
*Type definition for NULL.*

### 8.12.1 Detailed Description

This file contains the standard values.

**Author**

Mohamed Alaa

**Version**

1.0.0

**Date**

2023-06-18

## 8.13 MCAL/GPIO/GPIO\_config.h File Reference

This file contains the configuration parameters for the GPIO module.

```
#include "GPIO_interface.h"
```

Include dependency graph for GPIO\_config.h: This graph shows which files directly or indirectly include this file:

### Macros

- #define [DEFAULT\\_PIN\\_INPUT\\_TYPE](#) ([GPIO\\_Input\\_Type\\_Pull\\_Down](#))  
*The default input type for the GPIO pins.*
- #define [DEFAULT\\_PIN\\_OUTPUT\\_TYPE](#) ([GPIO\\_Output\\_Type\\_Push\\_Pull](#))  
*The default output type for the GPIO pins.*

### 8.13.1 Detailed Description

This file contains the configuration parameters for the GPIO module.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the configuration parameters for the GPIO module.

#### Date

2023-06-25

## 8.14 MCAL/GPIO/GPIO\_interface.h File Reference

This file is the interface file for the GPIO module.

This graph shows which files directly or indirectly include this file:

## Enumerations

- enum `t_GPIO_Ports` {  
`GPIO_Ports_A` = 0 , `GPIO_Ports_B` , `GPIO_Ports_C` , `GPIO_Ports_D` ,  
`GPIO_Ports_E` , `GPIO_Ports_F` , `GPIO_Ports_G` }  
*GPIO Ports.*
- enum `t_GPIO_Pins` {  
`GPIO_Pins_0` = 0 , `GPIO_Pins_1` , `GPIO_Pins_2` , `GPIO_Pins_3` ,  
`GPIO_Pins_4` , `GPIO_Pins_5` , `GPIO_Pins_6` , `GPIO_Pins_7` ,  
`GPIO_Pins_8` , `GPIO_Pins_9` , `GPIO_Pins_10` , `GPIO_Pins_11` ,  
`GPIO_Pins_12` , `GPIO_Pins_13` , `GPIO_Pins_14` , `GPIO_Pins_15` }  
*GPIO Pins.*
- enum `t_GPIO_Direction` { `GPIO_Direction_Input` = 0 , `GPIO_Direction_Output_10MHz` , `GPIO_Direction_Output_2MHz` ,  
`GPIO_Direction_Output_50MHz` }  
*GPIO Direction.*
- enum `t_GPIO_Output_Type` { `GPIO_Output_Type_Push_Pull` = 0 , `GPIO_Output_Type_Open_Drain` ,  
`GPIO_Output_Type_Alternate_Push_Pull` , `GPIO_Output_Type_Alternate_Open_Drain` }  
*GPIO Output Type.*
- enum `t_GPIO_Input_Type` { `GPIO_Input_Type_Analog` = 0 , `GPIO_Input_Type_Floating` , `GPIO_Input_Type_Pull_Down` ,  
`GPIO_Input_Type_Pull_Up` }  
*GPIO Input Type.*
- enum `t_GPIO_Value` { `GPIO_Value_Low` = 0 , `GPIO_Value_High` }  
*GPIO Pin Value.*

## Functions

- void `GPIO_vSetPinDirection` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Direction` tDirection)  
*This function is used to set the direction of a GPIO pin.*
- void `GPIO_vSetPinInputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Input_Type` tInputType)  
*This function is used to set the input type of a GPIO pin.*
- void `GPIO_vSetPinOutputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Output_Type` tOutputType)  
*This function is used to set the output type of a GPIO pin.*
- void `GPIO_vSetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Value` tValue)  
*This function is used to set the value of a GPIO pin.*
- `t_GPIO_Value` `GPIO_tGetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin)  
*This function is used to get the value of a GPIO pin.*

### 8.14.1 Detailed Description

This file is the interface file for the GPIO module.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the prototypes of the functions of the GPIO module.

#### Date

2023-06-25

## 8.15 MCAL/GPIO/GPIO\_private.h File Reference

This file contains the private macros and registers for the GPIO module.

```
#include "../LIB/LSTD_TYPES.h"
#include "../LIB/LSTD_HW_REGS.h"
```

Include dependency graph for GPIO\_private.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [t\\_GPIOx\\_RegisterMap](#)  
*GPIO Register Map.*

### Macros

- #define [BASE\\_ADDRESS\\_PORT\\_A\\_REGISTER\\_ADDRESS](#)(0x40010800, 0)  
*Base Address of Port A.*
- #define [GPIO\\_A\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_A)  
*GPIO Port A.*
- #define [BASE\\_ADDRESS\\_PORT\\_B\\_REGISTER\\_ADDRESS](#)(0x40010C00, 0)  
*Base Address of Port B.*
- #define [GPIO\\_B\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_B)  
*GPIO Port B.*
- #define [BASE\\_ADDRESS\\_PORT\\_C\\_REGISTER\\_ADDRESS](#)(0x40011000, 0)  
*Base Address of Port C.*
- #define [GPIO\\_C\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_C)  
*GPIO Port C.*
- #define [BASE\\_ADDRESS\\_PORT\\_D\\_REGISTER\\_ADDRESS](#)(0x40011400, 0)  
*Base Address of Port D.*
- #define [GPIO\\_D\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_D)  
*GPIO Port D.*
- #define [BASE\\_ADDRESS\\_PORT\\_E\\_REGISTER\\_ADDRESS](#)(0x40011800, 0)  
*Base Address of Port E.*
- #define [GPIO\\_E\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_E)  
*GPIO Port E.*
- #define [BASE\\_ADDRESS\\_PORT\\_F\\_REGISTER\\_ADDRESS](#)(0x40011C00, 0)  
*Base Address of Port F.*
- #define [GPIO\\_F\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_F)  
*GPIO Port F.*
- #define [BASE\\_ADDRESS\\_PORT\\_G\\_REGISTER\\_ADDRESS](#)(0x40012000, 0)  
*Base Address of Port G.*
- #define [GPIO\\_G\\_REGISTER](#)(t\_GPIOx\_RegisterMap, BASE\_ADDRESS\_PORT\_G)  
*GPIO Port G.*
- #define [PIN\\_SHIFT\\_VALUE](#) (4)  
*Pin Shift Value.*
- #define [PIN\\_RESET\\_MASK\\_VALUE](#) ((t\_u32)0x0000000FU)  
*Pin Reset Mask Value.*
- #define [PIN\\_RESET\\_CONFIGURATION\\_MASK\\_VALUE](#) ((t\_u32)0x0000000CU)  
*Pin Reset Configuration Mask Value.*

- `#define PIN_RESET_ODR_MASK_VALUE ((t_u32)0x00000001U)`  
*Pin Reset ODR Mask Value.*
- `#define PIN_CONFIGURATION_BITS_SHIFT_VALUE (2)`  
*Pin Mode Bits Shift Value.*
- `#define PIN_RESET_SHIFT_VALUE (16)`  
*Pin Reset Shift Value.*
- `#define PIN_RESET_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset Mask.*
- `#define PIN_RESET_CONFIGURATIONS_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_CONFIGURATION_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset Configurations Mask.*
- `#define PIN_RESET_ODR_MASK(GPIO_PIN_SPAN) ~(PIN_RESET_ODR_MASK_VALUE << GPIO_PIN_SPAN)`  
*Pin Reset ODR Mask.*

### 8.15.1 Detailed Description

This file contains the private macros and registers for the GPIO module.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the private macros and registers for the GPIO module.

#### Date

2023-06-25

## 8.16 MCAL/GPIO/GPIO\_program.c File Reference

This file contains the implementation for the GPIO module.

```
#include "../LIB/LSTD_TYPES.h"
#include "../LIB/LSTD_COMPILER.h"
#include "../LIB/LSTD_BITMATH.h"
#include "../LIB/LSTD_VALUES.h"
#include "GPIO_private.h"
#include "GPIO_interface.h"
#include "GPIO_config.h"
```

Include dependency graph for GPIO\_program.c:

### Macros

- `#define IS_PIN_IN_LOW_REGISTER(GPIO_PIN) (GPIO_PIN <= GPIO_Pins_7)`  
*Check if the pin is in the low register.*



## Functions

- static void `GPIO_vGetPortAddress` (`t_GPIO_Ports` tPort, `t_GPIOx_RegisterMap` \*\*ppu32PortBaseAddress)  
*This function gets the base address of a GPIO port.*
- static `t_u8` `GPIO_vGetPinSpan` (`t_GPIO_Pins` tPin)  
*This function gets the pin span.*
- static void `GPIO_vSetPinInputTypePullUpDown` (`t_GPIOx_RegisterMap` \*pu32PortBaseAddress, `t_GPIO_Pins` tPin, `t_GPIO_Input_Type` tInputType)  
*This function is used to set the pin input type.*
- void `GPIO_vSetPinDirection` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Direction` tDirection)  
*This function is used to set the direction of a GPIO pin.*
- void `GPIO_vSetPinInputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Input_Type` tInputType)  
*This function is used to set the input type of a GPIO pin.*
- void `GPIO_vSetPinOutputType` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Output_Type` tOutputType)  
*This function is used to set the output type of a GPIO pin.*
- void `GPIO_vSetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin, `t_GPIO_Value` tValue)  
*This function is used to set the value of a GPIO pin.*
- `t_GPIO_Value` `GPIO_tGetPinValue` (`t_GPIO_Ports` tPort, `t_GPIO_Pins` tPin)  
*This function is used to get the value of a GPIO pin.*

### 8.16.1 Detailed Description

This file contains the implementation for the GPIO module.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the implementation for the GPIO module.

#### Date

2023-06-25

### 8.16.2 Macro Definition Documentation

#### 8.16.2.1 IS\_PIN\_IN\_LOW\_REGISTER

```
#define IS_PIN_IN_LOW_REGISTER(  
    GPIO_PIN ) (GPIO_PIN <= GPIO_Pins_7)
```

Check if the pin is in the low register.

This macro checks if the pin is in the low register (CRL)

## Parameters

in	<i>GPIO_PIN</i>	The pin to check if it is in the low register (CRL) or not
----	-----------------	--

## 8.16.3 Function Documentation

### 8.16.3.1 GPIO\_vGetPortAddress()

```
static void GPIO_vGetPortAddress (
    t_GPIO_Ports tPort,
    t_GPIOx_RegisterMap ** ppu32PortBaseAddress ) [static]
```

This function gets the base address of a GPIO port.

This function gets the base address of a GPIO port

## Parameters

in	<i>tPort</i>	The GPIO port to get its base address
out	<i>ppu32PortBaseAddress</i>	The base address of the GPIO port

```
33 {
34     P2VAR(VOLATILE t_GPIOx_RegisterMap)
35     pu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))NULL;
36
37     switch (tPort)
38     {
39     case GPIO_Ports_A:
40         pu32PortBaseAddress = &GPIO_A;
41         break;
42     case GPIO_Ports_B:
43         pu32PortBaseAddress = &GPIO_B;
44         break;
45     case GPIO_Ports_C:
46         pu32PortBaseAddress = &GPIO_C;
47         break;
48     case GPIO_Ports_D:
49         pu32PortBaseAddress = &GPIO_D;
50         break;
51     case GPIO_Ports_E:
52         pu32PortBaseAddress = &GPIO_E;
53         break;
54     case GPIO_Ports_F:
55         pu32PortBaseAddress = &GPIO_F;
56         break;
57     case GPIO_Ports_G:
58         pu32PortBaseAddress = &GPIO_G;
59         break;
60     default:
61         break;
62     }
63
64     *ppu32PortBaseAddress = (P2VAR(t_GPIOx_RegisterMap))pu32PortBaseAddress;
65 }
```

References [GPIO\\_A](#), [GPIO\\_B](#), [GPIO\\_C](#), [GPIO\\_D](#), [GPIO\\_E](#), [GPIO\\_F](#), [GPIO\\_G](#), [GPIO\\_Ports\\_A](#), [GPIO\\_Ports\\_B](#), [GPIO\\_Ports\\_C](#), [GPIO\\_Ports\\_D](#), [GPIO\\_Ports\\_E](#), [GPIO\\_Ports\\_F](#), [GPIO\\_Ports\\_G](#), [NULL](#), [P2VAR](#), and [VOLATILE](#).

Referenced by [GPIO\\_tGetPinValue\(\)](#), [GPIO\\_vSetPinDirection\(\)](#), [GPIO\\_vSetPinInputType\(\)](#), [GPIO\\_vSetPinOutputType\(\)](#), and [GPIO\\_vSetPinValue\(\)](#).

Here is the caller graph for this function:

### 8.16.3.2 GPIO\_vGetPinSpan()

```
static t_u8 GPIO_vGetPinSpan (
    t_GPIO_Pins tPin ) [static]
```

This function gets the pin span.

This function gets the pin span, which is the number of bits to shift to reach the target pin mode and configuration bits

#### Parameters

in	<i>tPin</i>	The target pin
----	-------------	----------------

#### Returns

t\_u8 The pin span

```
74 {
75     /* Get the pin span (the number of bits to shift to reach the target pin mode and configuration bits)
76     */
77     return (t_u8)((tPin <= GPIO_Pins_7) ? tPin : (tPin - GPIO_Pins_8)) * PIN_SHIFT_VALUE;
```

References [GPIO\\_Pins\\_7](#), [GPIO\\_Pins\\_8](#), and [PIN\\_SHIFT\\_VALUE](#).

Referenced by [GPIO\\_vSetPinDirection\(\)](#), [GPIO\\_vSetPinInputType\(\)](#), and [GPIO\\_vSetPinOutputType\(\)](#).

Here is the caller graph for this function:

### 8.16.3.3 GPIO\_vSetPinInputTypePullUpDown()

```
static void GPIO_vSetPinInputTypePullUpDown (
    t_GPIOx_RegisterMap * pu32PortBaseAddress,
    t_GPIO_Pins tPin,
    t_GPIO_Input_Type tInputType ) [static]
```

This function is used to set the pin input type.

This function is used to set the pin input type in case of it is pull-up or pull-down in the ODR register

#### Parameters

in	<i>pu32PortBaseAddress</i>	The base address of the GPIO port
in	<i>tPin</i>	The target pin
in	<i>tInputType</i>	The input type of the target pin (pull-up or pull-down)

#### See also

[t\\_GPIO\\_Input\\_Type](#)

```
88 {
89     if (tInputType >= GPIO_Input_Type_Pull_Down)
90     {
91         /* Get the pin input type (pull-up or pull-down) */
92         t_u8 u8PinInputType = GET_BIT(tInputType, 0);
```

```

93
94     /* Set the pull-up/pull-down resistor */
95     pu32PortBaseAddress->ODR = (pu32PortBaseAddress->ODR & PIN_RESET_ODR_MASK(tPin)) |
    (t_u32)(u8PinInputType << tPin);
96 }
97 else
98 {
99     /* Do nothing */
100 }
101 }

```

References [GET\\_BIT](#), [GPIO\\_Input\\_Type\\_Pull\\_Down](#), [t\\_GPIOx\\_RegisterMap::ODR](#), and [PIN\\_RESET\\_ODR\\_MASK](#).

Referenced by [GPIO\\_vSetPinInputType\(\)](#).

Here is the caller graph for this function:

## 8.17 MCAL/index.h File Reference

## 8.18 MCAL/RCC/RCC\_config.h File Reference

This file contains the configuration parameters for the RCC module.

```

#include "../LIB/LSTD_VALUES.h"
#include "RCC_interface.h"

```

Include dependency graph for RCC\_config.h: This graph shows which files directly or indirectly include this file:

### Macros

- [#define RCC\\_PLL\\_SRC \(RCC\\_PLLSource\\_HSE\)](#)  
*This macro defines the PLL source.*
- [#define RCC\\_PLL\\_MUL \(RCC\\_PLLMulFactors\\_9\)](#)  
*This macro defines the PLL multiplication factor.*
- [#define RCC\\_PLL\\_HSE\\_DIVIDE\\_BY\\_2 \(TRUE\)](#)  
*This macro defines the PLL entry HSE divider.*
- [#define RCC\\_SYSTEM\\_CLOCK\\_SOURCE \(RCC\\_SystemClock\\_PLL\)](#)  
*This macro defines the system clock source.*
- [#define RCC\\_MCO\\_SOURCE \(RCC\\_MCOSources\\_NoClock\)](#)  
*This macro defines the MCO source.*
- [#define RCC\\_ADC\\_PRESCALER \(RCC\\_ADCPrescaler\\_DividedBy4\)](#)  
*This macro defines the ADC clock prescaler.*
- [#define RCC\\_AHB\\_PRESCALER \(RCC\\_AHBPrescaler\\_NotDivided\)](#)  
*This macro defines the AHB clock prescaler.*
- [#define RCC\\_APB1\\_PRESCALER \(RCC\\_APBPrescaler\\_NotDivided\)](#)  
*This macro defines the APB1 clock prescaler.*
- [#define RCC\\_APB2\\_PRESCALER \(RCC\\_APBPrescaler\\_NotDivided\)](#)  
*This macro defines the APB2 clock prescaler.*
- [#define RCC\\_USB\\_PRESCALER \(RCC\\_USBPrescaler\\_1\)](#)  
*This macro defines the USB clock prescaler.*
- [#define RCC\\_ENABLE\\_CSS \(FALSE\)](#)  
*This macro defines the clock security system state.*

### 8.18.1 Detailed Description

This file contains the configuration parameters for the RCC module.

#### Author

Mohamed Alaa

#### Version

1.0.0

This file contains the configuration parameters for the RCC module.

#### Date

2023-06-18

## 8.19 MCAL/RCC/RCC\_interface.h File Reference

This file is the interface file for the RCC module.

This graph shows which files directly or indirectly include this file:

### Enumerations

- enum [RCC\\_PLLMulFactors](#) {  
[RCC\\_PLLMulFactors\\_2](#) = 0 , [RCC\\_PLLMulFactors\\_3](#) , [RCC\\_PLLMulFactors\\_4](#) , [RCC\\_PLLMulFactors\\_5](#) ,  
[RCC\\_PLLMulFactors\\_6](#) , [RCC\\_PLLMulFactors\\_7](#) , [RCC\\_PLLMulFactors\\_8](#) , [RCC\\_PLLMulFactors\\_9](#) ,  
[RCC\\_PLLMulFactors\\_10](#) , [RCC\\_PLLMulFactors\\_11](#) , [RCC\\_PLLMulFactors\\_12](#) , [RCC\\_PLLMulFactors\\_13](#) ,  
[RCC\\_PLLMulFactors\\_14](#) , [RCC\\_PLLMulFactors\\_15](#) , [RCC\\_PLLMulFactors\\_16](#) }  
*This enum contains the multiplication factors of the PLL.*
- enum [RCC\\_AHBPrescaler](#) {  
[RCC\\_AHBPrescaler\\_NotDivided](#) = 0 , [RCC\\_AHBPrescaler\\_DividedBy2](#) = 0b1000 , [RCC\\_AHBPrescaler\\_DividedBy4](#)  
, [RCC\\_AHBPrescaler\\_DividedBy8](#) ,  
[RCC\\_AHBPrescaler\\_DividedBy16](#) , [RCC\\_AHBPrescaler\\_DividedBy64](#) , [RCC\\_AHBPrescaler\\_DividedBy128](#)  
, [RCC\\_AHBPrescaler\\_DividedBy256](#) ,  
[RCC\\_AHBPrescaler\\_DividedBy512](#) }  
*This enum contains the prescaler factors of the AHB bus.*
- enum [RCC\\_APBPrescaler](#) {  
[RCC\\_APBPrescaler\\_NotDivided](#) = 0 , [RCC\\_APBPrescaler\\_DividedBy2](#) = 0b100 , [RCC\\_APBPrescaler\\_DividedBy4](#)  
, [RCC\\_APBPrescaler\\_DividedBy8](#) ,  
[RCC\\_APBPrescaler\\_DividedBy16](#) }  
*This enum contains the prescaler factors of the APB (1 & 2) bus.*
- enum [RCC\\_ADCPrescaler](#) { [RCC\\_ADCPrescaler\\_DividedBy2](#) = 0 , [RCC\\_ADCPrescaler\\_DividedBy4](#) ,  
[RCC\\_ADCPrescaler\\_DividedBy6](#) , [RCC\\_ADCPrescaler\\_DividedBy8](#) }  
*This enum contains the prescaler factors of the ADC bus.*
- enum [RCC\\_SystemClock](#) { [RCC\\_SystemClock\\_HSI](#) = 0 , [RCC\\_SystemClock\\_HSE](#) , [RCC\\_SystemClock\\_PLL](#)  
}
  
*This enum contains the system clock sources.*
- enum [RCC\\_PLLSource](#) { [RCC\\_PLLSource\\_HSI\\_DividedBy2](#) = 0 , [RCC\\_PLLSource\\_HSE](#) }

*This enum contains the PLL clock sources.*

- enum `RCC_MCOSources` {  
`RCC_MCOSources_NoClock` = 0 , `RCC_MCOSources_SystemClock` = 0b100 , `RCC_MCOSources_HSI` ,  
`RCC_MCOSources_HSE` ,  
`RCC_MCOSources_PLL_DividedBy2` }

*This enum contains the MCO sources.*

- enum `RCC_USBPrescaler` { `RCC_USBPrescaler_1_5` = 0 , `RCC_USBPrescaler_1` }

*This enum contains the USB prescaler factors.*

- enum `t_RCC_APB2Peripherals` {  
`RCC_APB2Peripherals_AFIO` = 0 , `RCC_APB2Peripherals_PORTA` = 2 , `RCC_APB2Peripherals_PORTB` ,  
`RCC_APB2Peripherals_PORTC` ,  
`RCC_APB2Peripherals_PORTD` , `RCC_APB2Peripherals_PORTE` , `RCC_APB2Peripherals_PORTF` ,  
`RCC_APB2Peripherals_PORTG` ,  
`RCC_APB2Peripherals_ADC1` , `RCC_APB2Peripherals_ADC2` , `RCC_APB2Peripherals_TIM1` , `RCC_APB2Peripherals_SPI1`  
, `RCC_APB2Peripherals_USART1` = 14 , `RCC_APB2Peripherals_ADC3` }

*This enum contains the APB2 peripherals that are connected to the APB2 bus.*

- enum `t_RCC_APB1Peripherals` {  
`RCC_APB1Peripherals_TIM2` = 0 , `RCC_APB1Peripherals_TIM3` , `RCC_APB1Peripherals_TIM4` ,  
`RCC_APB1Peripherals_TIM5` ,  
`RCC_APB1Peripherals_TIM6` , `RCC_APB1Peripherals_TIM7` , `RCC_APB1Peripherals_WWDG` = 11 ,  
`RCC_APB1Peripherals_SPI2` = 14 ,  
`RCC_APB1Peripherals_SPI3` , `RCC_APB1Peripherals_USART2` = 17 , `RCC_APB1Peripherals_USART3` ,  
`RCC_APB1Peripherals_UART4` ,  
`RCC_APB1Peripherals_UART5` , `RCC_APB1Peripherals_I2C1` , `RCC_APB1Peripherals_I2C2` , `RCC_APB1Peripherals_USB`  
, `RCC_APB1Peripherals_CAN` = 25 , `RCC_APB1Peripherals_BKP` = 27 , `RCC_APB1Peripherals_PWR` ,  
`RCC_APB1Peripherals_DAC` }

*This enum contains the APB1 peripherals that are connected to the APB1 bus.*

- enum `t_RCC_AHBPeripherals` { `RCC_AHBPeripherals_DMA1` = 0 , `RCC_AHBPeripherals_DMA2` ,  
`RCC_AHBPeripherals_CRC` = 5 }

*This enum contains the AHB peripherals that are connected to the AHB bus.*

## Functions

- void `RCC_vInit` (void)

*This function initializes the RCC peripheral.*

- void `RCC_vEnablePeripheralABP2` (`t_RCC_APB2Peripherals` enuPeripheral)

*This function enables the clock of a peripheral connected to the APB2 bus.*

- void `RCC_vDisablePeripheralABP2` (`t_RCC_APB2Peripherals` enuPeripheral)

*This function disables the clock of a peripheral connected to the APB2 bus.*

- void `RCC_vEnablePeripheralABP1` (`t_RCC_APB1Peripherals` enuPeripheral)

*This function enables the clock of a peripheral connected to the APB1 bus.*

- void `RCC_vDisablePeripheralABP1` (`t_RCC_APB1Peripherals` enuPeripheral)

*This function disables the clock of a peripheral connected to the APB1 bus.*

- void `RCC_vEnablePeripheralAHB` (`t_RCC_AHBPeripherals` enuPeripheral)

*This function enables the clock of a peripheral connected to the AHB bus.*

- void `RCC_vDisablePeripheralAHB` (`t_RCC_AHBPeripherals` enuPeripheral)

*This function disables the clock of a peripheral connected to the AHB bus.*

### 8.19.1 Detailed Description

This file is the interface file for the RCC module.

**Author**

Mohamed Alaa

**Version**

1.0.0

This file contains the prototypes of the functions of the RCC module.

**Date**

2023-06-18

## 8.20 MCAL/RCC/RCC\_private.h File Reference

This file contains the private macros and registers for the RCC module.

```
#include "../LIB/LSTD_HW_REGS.h"
#include "../LIB/LSTD_TYPES.h"
#include "../LIB/LSTD_VALUES.h"
#include "../LIB/LSTD_SHARED.h"
Include dependency graph for RCC_private.h:
```

## 8.21 MCAL/RCC/RCC\_program.c File Reference

This file contains the implementation for the RCC module.

```
#include "../LIB/LSTD_TYPES.h"
#include "../LIB/LSTD_BITMATH.h"
#include "../LIB/LSTD_COMPILER.h"
#include "../LIB/LSTD_VALUES.h"
#include "RCC_private.h"
#include "RCC_interface.h"
#include "RCC_config.h"
Include dependency graph for RCC_program.c:
```

## Functions

- static void [RCC\\_vInitHSEClock](#) (void)  
*This function initializes the HSE clock.*
- static void [RCC\\_vInitHSIClock](#) (void)  
*This function initializes the HSI clock.*
- static void [RCC\\_vInitPLL](#) (void)  
*This function initializes the PLL clock.*
- static void [RCC\\_vInitSystemClock](#) (void)  
*This function initializes the system clock.*
- static void [RCC\\_vInitMCO](#) (void)  
*This function initializes the MCO.*
- static void [RCC\\_vSetAPB2PeripheralClockStatus](#) (t\_RCC\_APB2Peripherals enuPeripheral, t\_bool bStatus)  
*This function initializes a certain peripheral clock on APB2 bus.*
- static void [RCC\\_vSetAPB1PeripheralClockStatus](#) (t\_RCC\_APB1Peripherals enuPeripheral, t\_bool bStatus)  
*This function initializes a certain peripheral clock on APB1 bus.*
- static void [RCC\\_vSetAHBPeripheralClockStatus](#) (t\_RCC\_AHBPeripherals enuPeripheral, t\_bool bStatus)  
*This function initializes a certain peripheral clock on AHB bus.*
- static void [RCC\\_vSetADCPrescaler](#) (void)  
*This function sets the ADC prescaler.*
- static void [RCC\\_vSetUSBPrescaler](#) (void)  
*This function sets the USB prescaler.*
- static void [RCC\\_vSetAHBPrescaler](#) (void)  
*This function sets the AHB prescaler.*
- static void [RCC\\_vSetAPB1Prescaler](#) (void)  
*This function sets the APB1 prescaler.*
- static void [RCC\\_vSetAPB2Prescaler](#) (void)  
*This function sets the APB2 prescaler.*
- void [RCC\\_vInit](#) (void)  
*This function initializes the RCC peripheral.*
- void [RCC\\_vEnablePeripheralABP2](#) (t\_RCC\_APB2Peripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the APB2 bus.*
- void [RCC\\_vDisablePeripheralABP2](#) (t\_RCC\_APB2Peripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the APB2 bus.*
- void [RCC\\_vEnablePeripheralABP1](#) (t\_RCC\_APB1Peripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the APB1 bus.*
- void [RCC\\_vDisablePeripheralABP1](#) (t\_RCC\_APB1Peripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the APB1 bus.*
- void [RCC\\_vEnablePeripheralAHB](#) (t\_RCC\_AHBPeripherals enuPeripheral)  
*This function enables the clock of a peripheral connected to the AHB bus.*
- void [RCC\\_vDisablePeripheralAHB](#) (t\_RCC\_AHBPeripherals enuPeripheral)  
*This function disables the clock of a peripheral connected to the AHB bus.*

### 8.21.1 Detailed Description

This file contains the implementation for the RCC module.

Author

Mohamed Alaa



**Version**

1.0.0

This file contains the implementation for the RCC module.

**Date**

2023-06-18

**8.21.2 Function Documentation****8.21.2.1 RCC\_vInitHSEClock()**

```
static void RCC_vInitHSEClock (  
    void ) [static]
```

This function initializes the HSE clock.

This function initializes the HSE clock and waits until it is ready

```
23 {  
24     /* Enable HSE bypass */  
25     RCC.CR.HSEBYP = TRUE;  
26     /* Enable HSE */  
27     RCC.CR.HSEON = TRUE;  
28     /* Wait until HSE is ready */  
29     while (RCC.CR.HSERDY == FALSE)  
30         ;  
31  
32     if (RCC_ENABLE_CSS == TRUE)  
33     {  
34         /* Enable CSS */  
35         RCC.CR.CSSON = TRUE;  
36     }  
37     else  
38     {  
39         /* Disable CSS */  
40         RCC.CR.CSSON = FALSE;  
41     }  
42 }
```

References [FALSE](#), [RCC](#), [RCC\\_ENABLE\\_CSS](#), and [TRUE](#).

Referenced by [RCC\\_vInit\(\)](#), and [RCC\\_vInitPLL\(\)](#).

Here is the caller graph for this function:

**8.21.2.2 RCC\_vInitHSIClock()**

```
static void RCC_vInitHSIClock (  
    void ) [static]
```

This function initializes the HSI clock.

This function initializes the HSI clock and waits until it is ready

```
49 {  
50     /* Enable HSI */  
51     RCC.CR.HSION = TRUE;  
52     /* Wait until HSI is ready */  
53     while (RCC.CR.HSIRDY == FALSE)  
54         ;  
55 }
```

References [FALSE](#), [RCC](#), and [TRUE](#).

Referenced by [RCC\\_vInit\(\)](#), and [RCC\\_vInitPLL\(\)](#).

Here is the caller graph for this function:

### 8.21.2.3 RCC\_vInitPLL()

```
static void RCC_vInitPLL (
    void ) [static]
```

This function initializes the PLL clock.

This function initializes the PLL clock and waits until it is ready

```
62 {
63     if (RCC_PLL_SRC == RCC_PLLSource_HSE)
64     {
65         RCC_vInitHSEClock();
66     }
67     else if (RCC_PLL_SRC == RCC_PLLSource_HSI_DividedBy2)
68     {
69         RCC_vInitHSIClock();
70     }
71     else
72     {
73         /* Do nothing */
74     }
75
76     /* Set PLL source to HSE */
77     RCC.CFGR.PLLSRC = RCC_PLL_SRC;
78     /* Set PLL multiplication factor */
79     RCC.CFGR.PLLMUL = RCC_PLL_MUL;
80     /* Set PLL entry HSE divider */
81     RCC.CFGR.PLLXTPRE = RCC_PLL_HSE_DIVIDE_BY_2;
82     /* Enable PLL */
83     RCC.CR.PLLON = TRUE;
84     /* Wait until PLL is ready */
85     while (RCC.CR.PLLRDY == FALSE)
86         ;
87 }
```

References [FALSE](#), [RCC](#), [RCC\\_PLL\\_HSE\\_DIVIDE\\_BY\\_2](#), [RCC\\_PLL\\_MUL](#), [RCC\\_PLL\\_SRC](#), [RCC\\_PLLSource\\_HSE](#), [RCC\\_PLLSource\\_HSI\\_DividedBy2](#), [RCC\\_vInitHSEClock\(\)](#), [RCC\\_vInitHSIClock\(\)](#), and [TRUE](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the call graph for this function: Here is the caller graph for this function:

### 8.21.2.4 RCC\_vInitSystemClock()

```
static void RCC_vInitSystemClock (
    void ) [static]
```

This function initializes the system clock.

This function initializes the system clock

```
94 {
95     /* Set system clock source */
96     RCC.CFGR.SW = RCC_SYSTEM_CLOCK_SOURCE;
97     /* Wait until system clock source is set */
98     while (RCC.CFGR.SWS != RCC_SYSTEM_CLOCK_SOURCE)
99         ;
100 }
```

References [RCC](#), and [RCC\\_SYSTEM\\_CLOCK\\_SOURCE](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

### 8.21.2.5 RCC\_vInitMCO()

```
static void RCC_vInitMCO (
    void ) [static]
```

This function initializes the MCO.

This function initializes the MCO (Microcontroller Clock Output) and waits until it is ready

```
107 {
108     /* Set MCO source */
109     RCC.CFGR.MCO = RCC_MCO_SOURCE;
110 }
```

References [RCC](#), and [RCC\\_MCO\\_SOURCE](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

### 8.21.2.6 RCC\_vSetAPB2PeripheralClockStatus()

```
static void RCC_vSetAPB2PeripheralClockStatus (
    t_RCC_APB2Peripherals enuPeripheral,
    t_bool bStatus ) [static]
```

This function initializes a certain peripheral clock on APB2 bus.

This function initializes a certain peripheral clock on APB2 bus

#### Parameters

in	<i>enuPeripheral</i>	Peripheral to enable/disable its clock
in	<i>bStatus</i>	Enable/Disable peripheral clock

#### See also

##### RCC\_APB2Peripherals

```
120 {
121     switch (enuPeripheral)
122     {
123     case RCC_APB2Peripherals_AFIO:
124         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, AFIOEN, bStatus);
125         break;
126     case RCC_APB2Peripherals_PORTA:
127         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPAEN, bStatus);
128         break;
129     case RCC_APB2Peripherals_PORTB:
130         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPBEN, bStatus);
131         break;
132     case RCC_APB2Peripherals_PORTC:
133         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPCEN, bStatus);
134         break;
135     case RCC_APB2Peripherals_PORTD:
136         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPDEN, bStatus);
137         break;
138     case RCC_APB2Peripherals_PORTE:
139         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPEEN, bStatus);
140         break;
141     case RCC_APB2Peripherals_PORTF:
142         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPFEN, bStatus);
143         break;
144     case RCC_APB2Peripherals_PORTG:
145         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, IOPGEN, bStatus);
```

```

146         break;
147     case RCC_APB2Peripherals_ADC1:
148         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, ADC1EN, bStatus);
149         break;
150     case RCC_APB2Peripherals_ADC2:
151         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, ADC2EN, bStatus);
152         break;
153     case RCC_APB2Peripherals_TIM1:
154         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, TIM1EN, bStatus);
155         break;
156     case RCC_APB2Peripherals_SPI1:
157         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, SPI1EN, bStatus);
158         break;
159     case RCC_APB2Peripherals_USART1:
160         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, USART1EN, bStatus);
161         break;
162     case RCC_APB2Peripherals_ADC3:
163         RCC_SET_REGISTER_BIT_STATUS (APB2ENR, ADC3EN, bStatus);
164         break;
165     default:
166         /* Do nothing */
167         break;
168     }
169 }

```

References [RCC\\_APB2Peripherals\\_ADC1](#), [RCC\\_APB2Peripherals\\_ADC2](#), [RCC\\_APB2Peripherals\\_ADC3](#), [RCC\\_APB2Peripherals\\_AFIO](#), [RCC\\_APB2Peripherals\\_PORTA](#), [RCC\\_APB2Peripherals\\_PORTB](#), [RCC\\_APB2Peripherals\\_PORTC](#), [RCC\\_APB2Peripherals\\_PORTD](#), [RCC\\_APB2Peripherals\\_PORTE](#), [RCC\\_APB2Peripherals\\_PORTF](#), [RCC\\_APB2Peripherals\\_PORTG](#), [RCC\\_APB2Peripherals\\_SPI1](#), [RCC\\_APB2Peripherals\\_TIM1](#), [RCC\\_APB2Peripherals\\_USART1](#), and [RCC\\_SET\\_REGISTER\\_BIT\\_STATUS](#).

Referenced by [RCC\\_vDisablePeripheralABP2\(\)](#), and [RCC\\_vEnablePeripheralABP2\(\)](#).

Here is the caller graph for this function:

### 8.21.2.7 RCC\_vSetAPB1PeripheralClockStatus()

```

static void RCC_vSetAPB1PeripheralClockStatus (
    t_RCC_APB1Peripherals enuPeripheral,
    t_bool bStatus ) [static]

```

This function initializes a certain peripheral clock on APB1 bus.

This function initializes a certain peripheral clock on APB1 bus

#### Parameters

in	<i>enuPeripheral</i>	Peripheral to enable/disable its clock
in	<i>bStatus</i>	Enable/Disable peripheral clock

#### See also

[t\\_RCC\\_APB1Peripherals](#)

```

179 {
180     switch (enuPeripheral)
181     {
182     case RCC_APB1Peripherals_TIM2:
183         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM2EN, bStatus);
184         break;
185     case RCC_APB1Peripherals_TIM3:
186         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM3EN, bStatus);
187         break;
188     case RCC_APB1Peripherals_TIM4:
189         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM4EN, bStatus);
190         break;
191     case RCC_APB1Peripherals_TIM5:
192         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM5EN, bStatus);

```

```

193         break;
194     case RCC_APB1Peripherals_TIM6:
195         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM6EN, bStatus);
196         break;
197     case RCC_APB1Peripherals_TIM7:
198         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, TIM7EN, bStatus);
199         break;
200     case RCC_APB1Peripherals_WWDG:
201         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, WWDGEN, bStatus);
202         break;
203     case RCC_APB1Peripherals_SPI2:
204         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, SPI2EN, bStatus);
205         break;
206     case RCC_APB1Peripherals_SPI3:
207         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, SPI3EN, bStatus);
208         break;
209     case RCC_APB1Peripherals_USART2:
210         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, USART2EN, bStatus);
211         break;
212     case RCC_APB1Peripherals_USART3:
213         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, USART3EN, bStatus);
214         break;
215     case RCC_APB1Peripherals_UART4:
216         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, UART4EN, bStatus);
217         break;
218     case RCC_APB1Peripherals_UART5:
219         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, UART5EN, bStatus);
220         break;
221     case RCC_APB1Peripherals_I2C1:
222         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, I2C1EN, bStatus);
223         break;
224     case RCC_APB1Peripherals_I2C2:
225         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, I2C2EN, bStatus);
226         break;
227     case RCC_APB1Peripherals_USB:
228         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, USBEN, bStatus);
229         break;
230     case RCC_APB1Peripherals_CAN:
231         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, CANEN, bStatus);
232         break;
233     case RCC_APB1Peripherals_BKP:
234         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, BKPEN, bStatus);
235         break;
236     case RCC_APB1Peripherals_PWR:
237         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, PWREN, bStatus);
238         break;
239     case RCC_APB1Peripherals_DAC:
240         RCC_SET_REGISTER_BIT_STATUS (APB1ENR, DACEN, bStatus);
241         break;
242     default:
243         /* Do nothing */
244         break;
245     }
246 }

```

References [RCC\\_APB1Peripherals\\_BKP](#), [RCC\\_APB1Peripherals\\_CAN](#), [RCC\\_APB1Peripherals\\_DAC](#), [RCC\\_APB1Peripherals\\_I2C1](#), [RCC\\_APB1Peripherals\\_I2C2](#), [RCC\\_APB1Peripherals\\_PWR](#), [RCC\\_APB1Peripherals\\_SPI2](#), [RCC\\_APB1Peripherals\\_SPI3](#), [RCC\\_APB1Peripherals\\_TIM2](#), [RCC\\_APB1Peripherals\\_TIM3](#), [RCC\\_APB1Peripherals\\_TIM4](#), [RCC\\_APB1Peripherals\\_TIM5](#), [RCC\\_APB1Peripherals\\_TIM6](#), [RCC\\_APB1Peripherals\\_TIM7](#), [RCC\\_APB1Peripherals\\_UART4](#), [RCC\\_APB1Peripherals\\_UART5](#), [RCC\\_APB1Peripherals\\_USART2](#), [RCC\\_APB1Peripherals\\_USART3](#), [RCC\\_APB1Peripherals\\_USB](#), [RCC\\_APB1Peripherals\\_WWDG](#), and [RCC\\_SET\\_REGISTER\\_BIT\\_STATUS](#).

Referenced by [RCC\\_vDisablePeripheralABP1\(\)](#), and [RCC\\_vEnablePeripheralABP1\(\)](#).

Here is the caller graph for this function:

### 8.21.2.8 RCC\_vSetAHBPeripheralClockStatus()

```

static void RCC_vSetAHBPeripheralClockStatus (
    t_RCC_AHBPeripherals enumPeripheral,
    t_bool bStatus ) [static]

```

This function initializes a certain peripheral clock on AHB bus.

This function initializes a certain peripheral clock on AHB bus

## Parameters

in	<i>enuPeripheral</i>	Peripheral to enable/disable its clock
in	<i>bStatus</i>	Enable/Disable peripheral clock

## See also

[t\\_RCC\\_AHBPeripherals](#)

```

256 {
257     switch (enuPeripheral)
258     {
259     case RCC_AHBPeripherals_DMA1:
260         RCC_SET_REGISTER_BIT_STATUS(AHBENR, DMA1EN, bStatus);
261         break;
262     case RCC_AHBPeripherals_DMA2:
263         RCC_SET_REGISTER_BIT_STATUS(AHBENR, DMA2EN, bStatus);
264         break;
265     case RCC_AHBPeripherals_CRC:
266         RCC_SET_REGISTER_BIT_STATUS(AHBENR, CRCEN, bStatus);
267         break;
268     default:
269         /* Do nothing */
270         break;
271     }
272 }

```

References [RCC\\_AHBPeripherals\\_CRC](#), [RCC\\_AHBPeripherals\\_DMA1](#), [RCC\\_AHBPeripherals\\_DMA2](#), and [RCC\\_SET\\_REGISTER\\_BIT\\_STATUS](#).

Referenced by [RCC\\_vDisablePeripheralAHB\(\)](#), and [RCC\\_vEnablePeripheralAHB\(\)](#).

Here is the caller graph for this function:

**8.21.2.9 RCC\_vSetADCPrescaler()**

```

static void RCC_vSetADCPrescaler (
    void ) [static]

```

This function sets the ADC prescaler.

This function sets the ADC prescaler to be used by ADC

```

279 {
280     /* Set ADC prescaler */
281     RCC.CFGR.ADCPRE = RCC_ADC_PRESCALER;
282 }

```

References [RCC](#), and [RCC\\_ADC\\_PRESCALER](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

**8.21.2.10 RCC\_vSetUSBPrescaler()**

```

static void RCC_vSetUSBPrescaler (
    void ) [static]

```

This function sets the USB prescaler.

This function sets the USB prescaler to be used by USB

```

289 {
290     /* Set USB prescaler */
291     RCC.CFGR.USBPRE = RCC_USB_PRESCALER;
292 }

```

References [RCC](#), and [RCC\\_USB\\_PRESCALER](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

### 8.21.2.11 RCC\_vSetAHBPrescaler()

```
static void RCC_vSetAHBPrescaler (  
    void ) [static]
```

This function sets the AHB prescaler.

This function sets the AHB prescaler to be used by AHB bus

```
299 {  
300     /* Set AHB prescaler */  
301     RCC.CFGR.HPRE = RCC_AHB_PRESCALER;  
302 }
```

References [RCC](#), and [RCC\\_AHB\\_PRESCALER](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

### 8.21.2.12 RCC\_vSetAPB1Prescaler()

```
static void RCC_vSetAPB1Prescaler (  
    void ) [static]
```

This function sets the APB1 prescaler.

This function sets the APB1 prescaler to be used by APB1 bus

```
309 {  
310     /* Set APB1 prescaler */  
311     RCC.CFGR.PPRE1 = RCC_APB1_PRESCALER;  
312 }
```

References [RCC](#), and [RCC\\_APB1\\_PRESCALER](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

### 8.21.2.13 RCC\_vSetAPB2Prescaler()

```
static void RCC_vSetAPB2Prescaler (  
    void ) [static]
```

This function sets the APB2 prescaler.

This function sets the APB2 prescaler to be used by APB2 bus

```
319 {  
320     /* Set APB2 prescaler */  
321     RCC.CFGR.PPRE2 = RCC_APB2_PRESCALER;  
322 }
```

References [RCC](#), and [RCC\\_APB2\\_PRESCALER](#).

Referenced by [RCC\\_vInit\(\)](#).

Here is the caller graph for this function:

## 8.22 main.c File Reference

```
#include "APPS/APPS_main.h"
```

Include dependency graph for main.c:

### Functions

- int [main](#) (void)

### 8.22.1 Function Documentation

#### 8.22.1.1 main()

```
int main (
    void )
4 {
5     vAPPS\_main();
6
7     for (;;)
8     {
9     }
10 }
```

References [vAPPS\\_main\(\)](#).

Here is the call graph for this function:



# Index

/github/workspace/README.md, [113](#)

\_\_pad0\_\_

- [t\\_RCC\\_AHBENR, 66](#)
- [t\\_RCC\\_APB1ENR, 69](#)
- [t\\_RCC\\_APB1RSTR, 75](#)
- [t\\_RCC\\_APB2ENR, 80](#)
- [t\\_RCC\\_APB2RSTR, 85](#)
- [t\\_RCC\\_BDCR, 89](#)
- [t\\_RCC\\_CFGR, 93](#)
- [t\\_RCC\\_CIR, 97](#)
- [t\\_RCC\\_CR, 101](#)
- [t\\_RCC\\_CSR, 105](#)

\_\_pad1\_\_

- [t\\_RCC\\_AHBENR, 66](#)
- [t\\_RCC\\_APB1ENR, 69](#)
- [t\\_RCC\\_APB1RSTR, 75](#)
- [t\\_RCC\\_APB2ENR, 82](#)
- [t\\_RCC\\_APB2RSTR, 87](#)
- [t\\_RCC\\_BDCR, 89](#)
- [t\\_RCC\\_CFGR, 94](#)
- [t\\_RCC\\_CIR, 98](#)
- [t\\_RCC\\_CR, 103](#)
- [t\\_RCC\\_CSR, 105](#)

\_\_pad2\_\_

- [t\\_RCC\\_APB1ENR, 70](#)
- [t\\_RCC\\_APB1RSTR, 76](#)
- [t\\_RCC\\_APB2ENR, 83](#)
- [t\\_RCC\\_APB2RSTR, 87](#)
- [t\\_RCC\\_BDCR, 90](#)
- [t\\_RCC\\_CIR, 99](#)
- [t\\_RCC\\_CR, 103](#)

\_\_pad3\_\_

- [t\\_RCC\\_APB1ENR, 71](#)
- [t\\_RCC\\_APB1RSTR, 77](#)
- [t\\_RCC\\_CIR, 100](#)

\_\_pad4\_\_

- [t\\_RCC\\_APB1ENR, 71](#)
- [t\\_RCC\\_APB1RSTR, 78](#)

\_\_pad5\_\_

- [t\\_RCC\\_APB1ENR, 72](#)
- [t\\_RCC\\_APB1RSTR, 79](#)

ADC1EN

- [t\\_RCC\\_APB2ENR, 82](#)

ADC1RST

- [t\\_RCC\\_APB2RSTR, 86](#)

ADC2EN

- [t\\_RCC\\_APB2ENR, 82](#)

ADC2RST

- [t\\_RCC\\_APB2RSTR, 86](#)

ADC3EN

- [t\\_RCC\\_APB2ENR, 83](#)

ADC3RST

- [t\\_RCC\\_APB2RSTR, 87](#)

ADCPRE

- [t\\_RCC\\_CFGR, 92](#)

AFIOEN

- [t\\_RCC\\_APB2ENR, 80](#)

AFIORST

- [t\\_RCC\\_APB2RSTR, 84](#)

AHBENR

- [t\\_RCC\\_RegisterMap, 109](#)

APB1ENR

- [t\\_RCC\\_RegisterMap, 109](#)

APB1RSTR

- [t\\_RCC\\_RegisterMap, 109](#)

APB2ENR

- [t\\_RCC\\_RegisterMap, 109](#)

APB2RSTR

- [t\\_RCC\\_RegisterMap, 109](#)

APPS/APPS\_main.c, [111](#)

APPS/APPS\_main.h, [112](#)

APPS/README.md, [113](#)

APPS/TestingGPIO/TestApp\_TestingGPIO\_main.c, [113](#)

APPS/TestingGPIO/TestApp\_TestingGPIO\_main.h, [113](#)

APPS\_main.c

- [TESTING\\_APPLICATION\\_MAIN\\_FUNC, 112](#)

- [TestingGPIO, 112](#)

BASE\_ADDRESS\_PORT\_A

- [GPIO Addresses, 34](#)

BASE\_ADDRESS\_PORT\_B

- [GPIO Addresses, 34](#)

BASE\_ADDRESS\_PORT\_C

- [GPIO Addresses, 34](#)

BASE\_ADDRESS\_PORT\_D

- [GPIO Addresses, 35](#)

BASE\_ADDRESS\_PORT\_E

- [GPIO Addresses, 35](#)

BASE\_ADDRESS\_PORT\_F

- [GPIO Addresses, 36](#)

BASE\_ADDRESS\_PORT\_G

- [GPIO Addresses, 36](#)

BDCR

- [t\\_RCC\\_RegisterMap, 109](#)

BDRST

- [t\\_RCC\\_BDCR, 90](#)

Bit Manipulation Math Macros, [11](#)

- [CLEAR\\_BIT, 12](#)

- [GET\\_BIT, 12](#)

- SET\_BIT, [11](#)
- TOGGLE\_BIT, [12](#)
- BKPEN
  - t\_RCC\_APB1ENR, [72](#)
- BKPRST
  - t\_RCC\_APB1RSTR, [78](#)
- BRR
  - t\_GPIOx\_RegisterMap, [64](#)
- BSRR
  - t\_GPIOx\_RegisterMap, [64](#)
- CANEN
  - t\_RCC\_APB1ENR, [71](#)
- CANRST
  - t\_RCC\_APB1RSTR, [78](#)
- CFGR
  - t\_RCC\_RegisterMap, [108](#)
- CIR
  - t\_RCC\_RegisterMap, [108](#)
- CLEAR\_BIT
  - Bit Manipulation Math Macros, [12](#)
- Compiler standard macros, [13](#)
  - CONST, [13](#)
  - CONSTP2CONST, [15](#)
  - CONSTP2VAR, [14](#)
  - P2CONST, [14](#)
  - P2FUNC, [15](#)
  - P2VAR, [14](#)
  - STATIC, [13](#)
  - VOLATILE, [14](#)
- CONST
  - Compiler standard macros, [13](#)
- CONSTP2CONST
  - Compiler standard macros, [15](#)
- CONSTP2VAR
  - Compiler standard macros, [14](#)
- CR
  - t\_RCC\_RegisterMap, [108](#)
- CRCEN
  - t\_RCC\_AHBENR, [66](#)
- CRH
  - t\_GPIOx\_RegisterMap, [64](#)
- CRL
  - t\_GPIOx\_RegisterMap, [63](#)
- CSR
  - t\_RCC\_RegisterMap, [109](#)
- CSSC
  - t\_RCC\_CIR, [99](#)
- CSSF
  - t\_RCC\_CIR, [97](#)
- CSSON
  - t\_RCC\_CR, [103](#)
- DACEN
  - t\_RCC\_APB1ENR, [72](#)
- DACRST
  - t\_RCC\_APB1RSTR, [78](#)
- DEFAULT\_PIN\_INPUT\_TYPE
  - GPIO Module, [25](#)
- DEFAULT\_PIN\_OUTPUT\_TYPE
  - GPIO Module, [25](#)
- DMA1EN
  - t\_RCC\_AHBENR, [65](#)
- DMA2EN
  - t\_RCC\_AHBENR, [65](#)
- FALSE
  - Standard values, [23](#)
- GET\_BIT
  - Bit Manipulation Math Macros, [12](#)
- GPIO Addresses, [33](#)
  - BASE\_ADDRESS\_PORT\_A, [34](#)
  - BASE\_ADDRESS\_PORT\_B, [34](#)
  - BASE\_ADDRESS\_PORT\_C, [34](#)
  - BASE\_ADDRESS\_PORT\_D, [35](#)
  - BASE\_ADDRESS\_PORT\_E, [35](#)
  - BASE\_ADDRESS\_PORT\_F, [36](#)
  - BASE\_ADDRESS\_PORT\_G, [36](#)
  - GPIO\_A, [34](#)
  - GPIO\_B, [34](#)
  - GPIO\_C, [35](#)
  - GPIO\_D, [35](#)
  - GPIO\_E, [35](#)
  - GPIO\_F, [36](#)
  - GPIO\_G, [36](#)
- GPIO Module, [23](#)
  - DEFAULT\_PIN\_INPUT\_TYPE, [25](#)
  - DEFAULT\_PIN\_OUTPUT\_TYPE, [25](#)
  - GPIO\_Direction\_Input, [27](#)
  - GPIO\_Direction\_Output\_10MHz, [27](#)
  - GPIO\_Direction\_Output\_2MHz, [27](#)
  - GPIO\_Direction\_Output\_50MHz, [27](#)
  - GPIO\_Input\_Type\_Analog, [28](#)
  - GPIO\_Input\_Type\_Floating, [28](#)
  - GPIO\_Input\_Type\_Pull\_Down, [28](#)
  - GPIO\_Input\_Type\_Pull\_Up, [28](#)
  - GPIO\_Output\_Type\_Alternate\_Open\_Drain, [27](#)
  - GPIO\_Output\_Type\_Alternate\_Push\_Pull, [27](#)
  - GPIO\_Output\_Type\_Open\_Drain, [27](#)
  - GPIO\_Output\_Type\_Push\_Pull, [27](#)
  - GPIO\_Pins\_0, [26](#)
  - GPIO\_Pins\_1, [26](#)
  - GPIO\_Pins\_10, [26](#)
  - GPIO\_Pins\_11, [26](#)
  - GPIO\_Pins\_12, [26](#)
  - GPIO\_Pins\_13, [26](#)
  - GPIO\_Pins\_14, [26](#)
  - GPIO\_Pins\_15, [26](#)
  - GPIO\_Pins\_2, [26](#)
  - GPIO\_Pins\_3, [26](#)
  - GPIO\_Pins\_4, [26](#)
  - GPIO\_Pins\_5, [26](#)
  - GPIO\_Pins\_6, [26](#)
  - GPIO\_Pins\_7, [26](#)
  - GPIO\_Pins\_8, [26](#)
  - GPIO\_Pins\_9, [26](#)
  - GPIO\_Ports\_A, [25](#)

- GPIO\_Ports\_B, [25](#)
- GPIO\_Ports\_C, [25](#)
- GPIO\_Ports\_D, [25](#)
- GPIO\_Ports\_E, [25](#)
- GPIO\_Ports\_F, [25](#)
- GPIO\_Ports\_G, [25](#)
- GPIO\_tGetPinValue, [32](#)
- GPIO\_Value\_High, [28](#)
- GPIO\_Value\_Low, [28](#)
- GPIO\_vSetPinDirection, [28](#)
- GPIO\_vSetPinInputType, [29](#)
- GPIO\_vSetPinOutputType, [30](#)
- GPIO\_vSetPinValue, [31](#)
- t\_GPIO\_Direction, [26](#)
- t\_GPIO\_Input\_Type, [27](#)
- t\_GPIO\_Output\_Type, [27](#)
- t\_GPIO\_Pins, [26](#)
- t\_GPIO\_Ports, [25](#)
- t\_GPIO\_Value, [28](#)
- GPIO Pins Constants, [37](#)
  - PIN\_CONFIGURATION\_BITS\_SHIFT\_VALUE, [38](#)
  - PIN\_RESET\_CONFIGURATION\_MASK\_VALUE, [38](#)
  - PIN\_RESET\_CONFIGURATIONS\_MASK, [39](#)
  - PIN\_RESET\_MASK, [38](#)
  - PIN\_RESET\_MASK\_VALUE, [37](#)
  - PIN\_RESET\_ODR\_MASK, [39](#)
  - PIN\_RESET\_ODR\_MASK\_VALUE, [38](#)
  - PIN\_RESET\_SHIFT\_VALUE, [38](#)
  - PIN\_SHIFT\_VALUE, [37](#)
- GPIO Registers, [32](#)
- GPIO\_A
  - GPIO Addresses, [34](#)
- GPIO\_B
  - GPIO Addresses, [34](#)
- GPIO\_C
  - GPIO Addresses, [35](#)
- GPIO\_D
  - GPIO Addresses, [35](#)
- GPIO\_Direction\_Input
  - GPIO Module, [27](#)
- GPIO\_Direction\_Output\_10MHz
  - GPIO Module, [27](#)
- GPIO\_Direction\_Output\_2MHz
  - GPIO Module, [27](#)
- GPIO\_Direction\_Output\_50MHz
  - GPIO Module, [27](#)
- GPIO\_E
  - GPIO Addresses, [35](#)
- GPIO\_F
  - GPIO Addresses, [36](#)
- GPIO\_G
  - GPIO Addresses, [36](#)
- GPIO\_Input\_Type\_Analog
  - GPIO Module, [28](#)
- GPIO\_Input\_Type\_Floating
  - GPIO Module, [28](#)
- GPIO\_Input\_Type\_Pull\_Down
  - GPIO Module, [28](#)
- GPIO\_Input\_Type\_Pull\_Up
  - GPIO Module, [28](#)
- GPIO\_Output\_Type\_Alternate\_Open\_Drain
  - GPIO Module, [27](#)
- GPIO\_Output\_Type\_Alternate\_Push\_Pull
  - GPIO Module, [27](#)
- GPIO\_Output\_Type\_Open\_Drain
  - GPIO Module, [27](#)
- GPIO\_Output\_Type\_Push\_Pull
  - GPIO Module, [27](#)
- GPIO\_Pins\_0
  - GPIO Module, [26](#)
- GPIO\_Pins\_1
  - GPIO Module, [26](#)
- GPIO\_Pins\_10
  - GPIO Module, [26](#)
- GPIO\_Pins\_11
  - GPIO Module, [26](#)
- GPIO\_Pins\_12
  - GPIO Module, [26](#)
- GPIO\_Pins\_13
  - GPIO Module, [26](#)
- GPIO\_Pins\_14
  - GPIO Module, [26](#)
- GPIO\_Pins\_15
  - GPIO Module, [26](#)
- GPIO\_Pins\_2
  - GPIO Module, [26](#)
- GPIO\_Pins\_3
  - GPIO Module, [26](#)
- GPIO\_Pins\_4
  - GPIO Module, [26](#)
- GPIO\_Pins\_5
  - GPIO Module, [26](#)
- GPIO\_Pins\_6
  - GPIO Module, [26](#)
- GPIO\_Pins\_7
  - GPIO Module, [26](#)
- GPIO\_Pins\_8
  - GPIO Module, [26](#)
- GPIO\_Pins\_9
  - GPIO Module, [26](#)
- GPIO\_Ports\_A
  - GPIO Module, [25](#)
- GPIO\_Ports\_B
  - GPIO Module, [25](#)
- GPIO\_Ports\_C
  - GPIO Module, [25](#)
- GPIO\_Ports\_D
  - GPIO Module, [25](#)
- GPIO\_Ports\_E
  - GPIO Module, [25](#)
- GPIO\_Ports\_F
  - GPIO Module, [25](#)
- GPIO\_Ports\_G
  - GPIO Module, [25](#)
- GPIO\_program.c

- GPIO\_vGetPinSpan, [124](#)
- GPIO\_vGetPortAddress, [124](#)
- GPIO\_vSetPinInputTypePullUpDown, [125](#)
- IS\_PIN\_IN\_LOW\_REGISTER, [123](#)
- GPIO\_tGetPinValue
  - GPIO Module, [32](#)
- GPIO\_Value\_High
  - GPIO Module, [28](#)
- GPIO\_Value\_Low
  - GPIO Module, [28](#)
- GPIO\_vGetPinSpan
  - GPIO\_program.c, [124](#)
- GPIO\_vGetPortAddress
  - GPIO\_program.c, [124](#)
- GPIO\_vSetPinDirection
  - GPIO Module, [28](#)
- GPIO\_vSetPinInputType
  - GPIO Module, [29](#)
- GPIO\_vSetPinInputTypePullUpDown
  - GPIO\_program.c, [125](#)
- GPIO\_vSetPinOutputType
  - GPIO Module, [30](#)
- GPIO\_vSetPinValue
  - GPIO Module, [31](#)
- Hardware registers macro-functions, [15](#)
  - REGISTER, [16](#)
  - REGISTER\_ADDRESS, [16](#)
  - REGISTER\_U16, [17](#)
  - REGISTER\_U32, [18](#)
  - REGISTER\_U8, [17](#)
- HPRE
  - t\_RCC\_CFGR, [92](#)
- HSEBYP
  - t\_RCC\_CR, [102](#)
- HSEON
  - t\_RCC\_CR, [102](#)
- HSERDY
  - t\_RCC\_CR, [102](#)
- HSERDYC
  - t\_RCC\_CIR, [99](#)
- HSERDYF
  - t\_RCC\_CIR, [96](#)
- HSERDYIE
  - t\_RCC\_CIR, [98](#)
- HSICAL
  - t\_RCC\_CR, [102](#)
- HSION
  - t\_RCC\_CR, [101](#)
- HSIRDY
  - t\_RCC\_CR, [101](#)
- HSIRDYC
  - t\_RCC\_CIR, [99](#)
- HSIRDYF
  - t\_RCC\_CIR, [96](#)
- HSIRDYIE
  - t\_RCC\_CIR, [98](#)
- HSITRIM
  - t\_RCC\_CR, [101](#)
- I2C1EN
  - t\_RCC\_APB1ENR, [70](#)
- I2C1RST
  - t\_RCC\_APB1RSTR, [77](#)
- I2C2EN
  - t\_RCC\_APB1ENR, [71](#)
- I2C2RST
  - t\_RCC\_APB1RSTR, [77](#)
- IDR
  - t\_GPIOx\_RegisterMap, [64](#)
- IOPAEN
  - t\_RCC\_APB2ENR, [80](#)
- IOPARST
  - t\_RCC\_APB2RSTR, [85](#)
- IOPBEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPBRST
  - t\_RCC\_APB2RSTR, [85](#)
- IOPCEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPCRST
  - t\_RCC\_APB2RSTR, [85](#)
- IOPDEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPDRST
  - t\_RCC\_APB2RSTR, [85](#)
- IOPEEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPERST
  - t\_RCC\_APB2RSTR, [85](#)
- IOPFEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPFRST
  - t\_RCC\_APB2RSTR, [86](#)
- IOPGEN
  - t\_RCC\_APB2ENR, [81](#)
- IOPGRST
  - t\_RCC\_APB2RSTR, [86](#)
- IS\_PIN\_IN\_LOW\_REGISTER
  - GPIO\_program.c, [123](#)
- IWDGRSTF
  - t\_RCC\_CSR, [106](#)
- LIB/LSTD\_BITMATH.h, [114](#)
- LIB/LSTD\_COMPILER.h, [115](#)
- LIB/LSTD\_HW\_REGS.h, [115](#)
- LIB/LSTD\_SHARED.h, [116](#)
- LIB/LSTD\_TYPES.h, [117](#)
- LIB/LSTD\_VALUES.h, [118](#)
- LPWRRSTF
  - t\_RCC\_CSR, [107](#)
- LSEBYP
  - t\_RCC\_BDCR, [89](#)
- LSEON
  - t\_RCC\_BDCR, [88](#)
- LSERDY
  - t\_RCC\_BDCR, [88](#)
- LSERDYC
  - t\_RCC\_CIR, [99](#)

- LSERDYF
  - t\_RCC\_CIR, [96](#)
- LSERDYIE
  - t\_RCC\_CIR, [98](#)
- LSION
  - t\_RCC\_CSR, [105](#)
- LSIRDY
  - t\_RCC\_CSR, [105](#)
- LSIRDYC
  - t\_RCC\_CIR, [98](#)
- LSIRDYF
  - t\_RCC\_CIR, [96](#)
- LSIRDYIE
  - t\_RCC\_CIR, [97](#)
- main
  - main.c, [138](#)
- main.c, [138](#)
  - main, [138](#)
- MCAL Peripherals, [39](#)
- MCAL/GPIO/GPIO\_config.h, [119](#)
- MCAL/GPIO/GPIO\_interface.h, [119](#)
- MCAL/GPIO/GPIO\_private.h, [121](#)
- MCAL/GPIO/GPIO\_program.c, [122](#)
- MCAL/index.h, [126](#)
- MCAL/RCC/RCC\_config.h, [126](#)
- MCAL/RCC/RCC\_interface.h, [127](#)
- MCAL/RCC/RCC\_private.h, [129](#)
- MCAL/RCC/RCC\_program.c, [129](#)
- MCO
  - t\_RCC\_CFGR, [94](#)
- NULL
  - Standard values, [23](#)
- ODR
  - t\_GPIOx\_RegisterMap, [64](#)
- P2CONST
  - Compiler standard macros, [14](#)
- P2FUNC
  - Compiler standard macros, [15](#)
- P2VAR
  - Compiler standard macros, [14](#)
- PIN\_CONFIGURATION\_BITS\_SHIFT\_VALUE
  - GPIO Pins Constants, [38](#)
- PIN\_RESET\_CONFIGURATION\_MASK\_VALUE
  - GPIO Pins Constants, [38](#)
- PIN\_RESET\_CONFIGURATIONS\_MASK
  - GPIO Pins Constants, [39](#)
- PIN\_RESET\_MASK
  - GPIO Pins Constants, [38](#)
- PIN\_RESET\_MASK\_VALUE
  - GPIO Pins Constants, [37](#)
- PIN\_RESET\_ODR\_MASK
  - GPIO Pins Constants, [39](#)
- PIN\_RESET\_ODR\_MASK\_VALUE
  - GPIO Pins Constants, [38](#)
- PIN\_RESET\_SHIFT\_VALUE
  - GPIO Pins Constants, [38](#)
- PIN\_SHIFT\_VALUE
  - GPIO Pins Constants, [37](#)
- PINRSTF
  - t\_RCC\_CSR, [106](#)
- PLLMUL
  - t\_RCC\_CFGR, [93](#)
- PLLON
  - t\_RCC\_CR, [103](#)
- PLLRDY
  - t\_RCC\_CR, [103](#)
- PLLRDYC
  - t\_RCC\_CIR, [99](#)
- PLLRDYF
  - t\_RCC\_CIR, [97](#)
- PLLRDYIE
  - t\_RCC\_CIR, [98](#)
- PLLSRC
  - t\_RCC\_CFGR, [93](#)
- PLLXTPRE
  - t\_RCC\_CFGR, [93](#)
- PORRSTF
  - t\_RCC\_CSR, [106](#)
- PPRE1
  - t\_RCC\_CFGR, [92](#)
- PPRE2
  - t\_RCC\_CFGR, [92](#)
- PWREN
  - t\_RCC\_APB1ENR, [72](#)
- PWRRST
  - t\_RCC\_APB1RSTR, [78](#)
- RCC
  - RCC Addresses, [58](#)
- RCC Addresses, [57](#)
  - RCC, [58](#)
  - RCC\_BASE\_ADDRESS, [58](#)
  - RCC\_SET\_REGISTER\_BIT\_STATUS, [58](#)
- RCC Configuration, [40](#)
  - RCC\_ADC\_PRESCALER, [42](#)
  - RCC\_AHB\_PRESCALER, [42](#)
  - RCC\_APB1\_PRESCALER, [43](#)
  - RCC\_APB2\_PRESCALER, [43](#)
  - RCC\_ENABLE\_CSS, [44](#)
  - RCC\_MCO\_SOURCE, [42](#)
  - RCC\_PLL\_HSE\_DIVIDE\_BY\_2, [41](#)
  - RCC\_PLL\_MUL, [41](#)
  - RCC\_PLL\_SRC, [41](#)
  - RCC\_SYSTEM\_CLOCK\_SOURCE, [41](#)
  - RCC\_USB\_PRESCALER, [43](#)
- RCC exported functions, [52](#)
  - RCC\_vDisablePeripheralABP1, [55](#)
  - RCC\_vDisablePeripheralABP2, [54](#)
  - RCC\_vDisablePeripheralAHB, [55](#)
  - RCC\_vEnablePeripheralABP1, [54](#)
  - RCC\_vEnablePeripheralABP2, [53](#)
  - RCC\_vEnablePeripheralAHB, [55](#)
  - RCC\_vInit, [53](#)
- RCC Interface Options, [44](#)

RCC\_ADCPrescaler, 48  
 RCC\_ADCPrescaler\_DividedBy2, 48  
 RCC\_ADCPrescaler\_DividedBy4, 48  
 RCC\_ADCPrescaler\_DividedBy6, 48  
 RCC\_ADCPrescaler\_DividedBy8, 48  
 RCC\_AHBPeripherals\_CRC, 52  
 RCC\_AHBPeripherals\_DMA1, 52  
 RCC\_AHBPeripherals\_DMA2, 52  
 RCC\_AHBPrescaler, 47  
 RCC\_AHBPrescaler\_DividedBy128, 47  
 RCC\_AHBPrescaler\_DividedBy16, 47  
 RCC\_AHBPrescaler\_DividedBy2, 47  
 RCC\_AHBPrescaler\_DividedBy256, 47  
 RCC\_AHBPrescaler\_DividedBy4, 47  
 RCC\_AHBPrescaler\_DividedBy512, 47  
 RCC\_AHBPrescaler\_DividedBy64, 47  
 RCC\_AHBPrescaler\_DividedBy8, 47  
 RCC\_AHBPrescaler\_NotDivided, 47  
 RCC\_APB1Peripherals\_BKP, 51  
 RCC\_APB1Peripherals\_CAN, 51  
 RCC\_APB1Peripherals\_DAC, 51  
 RCC\_APB1Peripherals\_I2C1, 51  
 RCC\_APB1Peripherals\_I2C2, 51  
 RCC\_APB1Peripherals\_PWR, 51  
 RCC\_APB1Peripherals\_SPI2, 51  
 RCC\_APB1Peripherals\_SPI3, 51  
 RCC\_APB1Peripherals\_TIM2, 51  
 RCC\_APB1Peripherals\_TIM3, 51  
 RCC\_APB1Peripherals\_TIM4, 51  
 RCC\_APB1Peripherals\_TIM5, 51  
 RCC\_APB1Peripherals\_TIM6, 51  
 RCC\_APB1Peripherals\_TIM7, 51  
 RCC\_APB1Peripherals\_UART4, 51  
 RCC\_APB1Peripherals\_UART5, 51  
 RCC\_APB1Peripherals\_USART2, 51  
 RCC\_APB1Peripherals\_USART3, 51  
 RCC\_APB1Peripherals\_USB, 51  
 RCC\_APB1Peripherals\_WWDG, 51  
 RCC\_APB2Peripherals\_ADC1, 50  
 RCC\_APB2Peripherals\_ADC2, 50  
 RCC\_APB2Peripherals\_ADC3, 50  
 RCC\_APB2Peripherals\_AFIO, 50  
 RCC\_APB2Peripherals\_PORTA, 50  
 RCC\_APB2Peripherals\_PORTB, 50  
 RCC\_APB2Peripherals\_PORTC, 50  
 RCC\_APB2Peripherals\_PORTD, 50  
 RCC\_APB2Peripherals\_PORTE, 50  
 RCC\_APB2Peripherals\_PORTF, 50  
 RCC\_APB2Peripherals\_PORTG, 50  
 RCC\_APB2Peripherals\_SPI1, 50  
 RCC\_APB2Peripherals\_TIM1, 50  
 RCC\_APB2Peripherals\_USART1, 50  
 RCC\_APBPrescaler, 47  
 RCC\_APBPrescaler\_DividedBy16, 48  
 RCC\_APBPrescaler\_DividedBy2, 48  
 RCC\_APBPrescaler\_DividedBy4, 48  
 RCC\_APBPrescaler\_DividedBy8, 48  
 RCC\_APBPrescaler\_NotDivided, 47  
 RCC\_MCOSources, 49  
 RCC\_MCOSources\_HSE, 49  
 RCC\_MCOSources\_HSI, 49  
 RCC\_MCOSources\_NoClock, 49  
 RCC\_MCOSources\_PLL\_DividedBy2, 49  
 RCC\_MCOSources\_SystemClock, 49  
 RCC\_PLLMulFactors, 46  
 RCC\_PLLMulFactors\_10, 46  
 RCC\_PLLMulFactors\_11, 46  
 RCC\_PLLMulFactors\_12, 46  
 RCC\_PLLMulFactors\_13, 46  
 RCC\_PLLMulFactors\_14, 46  
 RCC\_PLLMulFactors\_15, 46  
 RCC\_PLLMulFactors\_16, 46  
 RCC\_PLLMulFactors\_2, 46  
 RCC\_PLLMulFactors\_3, 46  
 RCC\_PLLMulFactors\_4, 46  
 RCC\_PLLMulFactors\_5, 46  
 RCC\_PLLMulFactors\_6, 46  
 RCC\_PLLMulFactors\_7, 46  
 RCC\_PLLMulFactors\_8, 46  
 RCC\_PLLMulFactors\_9, 46  
 RCC\_PLLSource, 49  
 RCC\_PLLSource\_HSE, 49  
 RCC\_PLLSource\_HSI\_DividedBy2, 49  
 RCC\_SystemClock, 48  
 RCC\_SystemClock\_HSE, 48  
 RCC\_SystemClock\_HSI, 48  
 RCC\_SystemClock\_PLL, 48  
 RCC\_USBPrescaler, 49  
 RCC\_USBPrescaler\_1, 50  
 RCC\_USBPrescaler\_1\_5, 50  
 t\_RCC\_AHBPeripherals, 52  
 t\_RCC\_APB1Peripherals, 51  
 t\_RCC\_APB2Peripherals, 50  
 RCC Module, 56  
 RCC Registers, 56  
 RCC\_ADC\_PRESCALER  
     RCC Configuration, 42  
 RCC\_ADCPrescaler  
     RCC Interface Options, 48  
 RCC\_ADCPrescaler\_DividedBy2  
     RCC Interface Options, 48  
 RCC\_ADCPrescaler\_DividedBy4  
     RCC Interface Options, 48  
 RCC\_ADCPrescaler\_DividedBy6  
     RCC Interface Options, 48  
 RCC\_ADCPrescaler\_DividedBy8  
     RCC Interface Options, 48  
 RCC\_AHB\_PRESCALER  
     RCC Configuration, 42  
 RCC\_AHBPeripherals\_CRC  
     RCC Interface Options, 52  
 RCC\_AHBPeripherals\_DMA1  
     RCC Interface Options, 52  
 RCC\_AHBPeripherals\_DMA2  
     RCC Interface Options, 52  
 RCC\_AHBPrescaler

- RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy128
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy16
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy2
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy256
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy4
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy512
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy64
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_DividedBy8
  - RCC Interface Options, [47](#)
- RCC\_AHBPrescaler\_NotDivided
  - RCC Interface Options, [47](#)
- RCC\_APB1\_PRESCALER
  - RCC Configuration, [43](#)
- RCC\_APB1Peripherals\_BKP
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_CAN
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_DAC
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_I2C1
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_I2C2
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_PWR
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_SPI2
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_SPI3
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM2
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM3
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM4
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM5
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM6
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_TIM7
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_UART4
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_UART5
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_USART2
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_USART3
  - RCC Interface Options, [51](#)
- RCC\_APB1Peripherals\_USB
  - RCC Interface Options, [51](#)
- RCC\_APB2Peripherals\_WWDG
  - RCC Interface Options, [51](#)
- RCC\_APB2\_PRESCALER
  - RCC Configuration, [43](#)
- RCC\_APB2Peripherals\_ADC1
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_ADC2
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_ADC3
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_AFIO
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTA
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTB
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTC
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTD
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTE
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTF
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_PORTG
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_SPI1
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_TIM1
  - RCC Interface Options, [50](#)
- RCC\_APB2Peripherals\_USART1
  - RCC Interface Options, [50](#)
- RCC\_APBPrescaler
  - RCC Interface Options, [47](#)
- RCC\_APBPrescaler\_DividedBy16
  - RCC Interface Options, [48](#)
- RCC\_APBPrescaler\_DividedBy2
  - RCC Interface Options, [48](#)
- RCC\_APBPrescaler\_DividedBy4
  - RCC Interface Options, [48](#)
- RCC\_APBPrescaler\_DividedBy8
  - RCC Interface Options, [48](#)
- RCC\_APBPrescaler\_NotDivided
  - RCC Interface Options, [47](#)
- RCC\_BASE\_ADDRESS
  - RCC Addresses, [58](#)
- RCC\_ENABLE\_CSS
  - RCC Configuration, [44](#)
- RCC\_MCO\_SOURCE
  - RCC Configuration, [42](#)
- RCC\_MCOSources
  - RCC Interface Options, [49](#)
- RCC\_MCOSources\_HSE
  - RCC Interface Options, [49](#)
- RCC\_MCOSources\_HSI
  - RCC Interface Options, [49](#)
- RCC\_MCOSources\_NoClock



- RCC Interface Options, [49](#)
- RCC\_MCOSources\_PLL\_DividedBy2
  - RCC Interface Options, [49](#)
- RCC\_MCOSources\_SystemClock
  - RCC Interface Options, [49](#)
- RCC\_PLL\_HSE\_DIVIDE\_BY\_2
  - RCC Configuration, [41](#)
- RCC\_PLL\_MUL
  - RCC Configuration, [41](#)
- RCC\_PLL\_SRC
  - RCC Configuration, [41](#)
- RCC\_PLLMulFactors
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_10
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_11
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_12
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_13
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_14
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_15
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_16
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_2
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_3
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_4
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_5
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_6
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_7
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_8
  - RCC Interface Options, [46](#)
- RCC\_PLLMulFactors\_9
  - RCC Interface Options, [46](#)
- RCC\_PLLSource
  - RCC Interface Options, [49](#)
- RCC\_PLLSource\_HSE
  - RCC Interface Options, [49](#)
- RCC\_PLLSource\_HSI\_DividedBy2
  - RCC Interface Options, [49](#)
- RCC\_program.c
  - RCC\_vInitHSEClock, [131](#)
  - RCC\_vInitHSIClock, [131](#)
  - RCC\_vInitMCO, [132](#)
  - RCC\_vInitPLL, [131](#)
  - RCC\_vInitSystemClock, [132](#)
  - RCC\_vSetADCPreScaler, [136](#)
  - RCC\_vSetAHBPeripheralClockStatus, [135](#)
  - RCC\_vSetAHBPrescaler, [136](#)
  - RCC\_vSetAPB1PeripheralClockStatus, [134](#)
  - RCC\_vSetAPB1Prescaler, [137](#)
  - RCC\_vSetAPB2PeripheralClockStatus, [133](#)
  - RCC\_vSetAPB2Prescaler, [137](#)
  - RCC\_vSetUSBPrescaler, [136](#)
- RCC\_SET\_REGISTER\_BIT\_STATUS
  - RCC Addresses, [58](#)
- RCC\_SYSTEM\_CLOCK\_SOURCE
  - RCC Configuration, [41](#)
- RCC\_SystemClock
  - RCC Interface Options, [48](#)
- RCC\_SystemClock\_HSE
  - RCC Interface Options, [48](#)
- RCC\_SystemClock\_HSI
  - RCC Interface Options, [48](#)
- RCC\_SystemClock\_PLL
  - RCC Interface Options, [48](#)
- RCC\_USB\_PRESCALER
  - RCC Configuration, [43](#)
- RCC\_USBPrescaler
  - RCC Interface Options, [49](#)
- RCC\_USBPrescaler\_1
  - RCC Interface Options, [50](#)
- RCC\_USBPrescaler\_1\_5
  - RCC Interface Options, [50](#)
- RCC\_vDisablePeripheralABP1
  - RCC exported functions, [55](#)
- RCC\_vDisablePeripheralABP2
  - RCC exported functions, [54](#)
- RCC\_vDisablePeripheralAHB
  - RCC exported functions, [55](#)
- RCC\_vEnablePeripheralABP1
  - RCC exported functions, [54](#)
- RCC\_vEnablePeripheralABP2
  - RCC exported functions, [53](#)
- RCC\_vEnablePeripheralAHB
  - RCC exported functions, [55](#)
- RCC\_vInit
  - RCC exported functions, [53](#)
- RCC\_vInitHSEClock
  - RCC\_program.c, [131](#)
- RCC\_vInitHSIClock
  - RCC\_program.c, [131](#)
- RCC\_vInitMCO
  - RCC\_program.c, [132](#)
- RCC\_vInitPLL
  - RCC\_program.c, [131](#)
- RCC\_vInitSystemClock
  - RCC\_program.c, [132](#)
- RCC\_vSetADCPreScaler
  - RCC\_program.c, [136](#)
- RCC\_vSetAHBPeripheralClockStatus
  - RCC\_program.c, [135](#)
- RCC\_vSetAHBPrescaler
  - RCC\_program.c, [136](#)
- RCC\_vSetAPB1PeripheralClockStatus
  - RCC\_program.c, [134](#)
- RCC\_vSetAPB1Prescaler



- RCC\_program.c, [137](#)
- RCC\_vSetAPB2PeripheralClockStatus
  - RCC\_program.c, [133](#)
- RCC\_vSetAPB2Prescaler
  - RCC\_program.c, [137](#)
- RCC\_vSetUSBPrescaler
  - RCC\_program.c, [136](#)
- REGISTER
  - Hardware registers macro-functions, [16](#)
- REGISTER\_ADDRESS
  - Hardware registers macro-functions, [16](#)
- REGISTER\_U16
  - Hardware registers macro-functions, [17](#)
- REGISTER\_U32
  - Hardware registers macro-functions, [18](#)
- REGISTER\_U8
  - Hardware registers macro-functions, [17](#)
- RMVF
  - t\_RCC\_CSR, [105](#)
- RTCEN
  - t\_RCC\_BDCR, [89](#)
- RTCSEL
  - t\_RCC\_BDCR, [89](#)
- SET\_BIT
  - Bit Manipulation Math Macros, [11](#)
- SET\_REGISTER\_BIT\_STATUS
  - Shared library, [61](#)
- SFTRSTF
  - t\_RCC\_CSR, [106](#)
- Shared library, [60](#)
  - SET\_REGISTER\_BIT\_STATUS, [61](#)
- SPI1EN
  - t\_RCC\_APB2ENR, [82](#)
- SPI1RST
  - t\_RCC\_APB2RSTR, [86](#)
- SPI2EN
  - t\_RCC\_APB1ENR, [69](#)
- SPI2RST
  - t\_RCC\_APB1RSTR, [76](#)
- SPI3EN
  - t\_RCC\_APB1ENR, [69](#)
- SPI3RST
  - t\_RCC\_APB1RSTR, [76](#)
- Standard data types, [19](#)
  - t\_bool, [20](#)
  - t\_c8, [20](#)
  - t\_fl32, [22](#)
  - t\_fl64, [22](#)
  - t\_s16, [21](#)
  - t\_s32, [21](#)
  - t\_s64, [21](#)
  - t\_s8, [20](#)
  - t\_u16, [20](#)
  - t\_u32, [21](#)
  - t\_u64, [21](#)
  - t\_u8, [20](#)
- Standard Library, [18](#)
- Standard values, [22](#)
  - FALSE, [23](#)
  - NULL, [23](#)
  - TRUE, [23](#)
- STATIC
  - Compiler standard macros, [13](#)
- SW
  - t\_RCC\_CFGR, [91](#)
- SWS
  - t\_RCC\_CFGR, [91](#)
- t\_bool
  - Standard data types, [20](#)
- t\_c8
  - Standard data types, [20](#)
- t\_fl32
  - Standard data types, [22](#)
- t\_fl64
  - Standard data types, [22](#)
- t\_GPIO\_Direction
  - GPIO Module, [26](#)
- t\_GPIO\_Input\_Type
  - GPIO Module, [27](#)
- t\_GPIO\_Output\_Type
  - GPIO Module, [27](#)
- t\_GPIO\_Pins
  - GPIO Module, [26](#)
- t\_GPIO\_Ports
  - GPIO Module, [25](#)
- t\_GPIO\_Value
  - GPIO Module, [28](#)
- t\_GPIOx\_RegisterMap, [63](#)
  - BRR, [64](#)
  - BSRR, [64](#)
  - CRH, [64](#)
  - CRL, [63](#)
  - IDR, [64](#)
  - ODR, [64](#)
- t\_RCC\_AHBENR, [65](#)
  - \_\_pad0\_\_, [66](#)
  - \_\_pad1\_\_, [66](#)
  - CRCEN, [66](#)
  - DMA1EN, [65](#)
  - DMA2EN, [65](#)
- t\_RCC\_AHBPeripherals
  - RCC Interface Options, [52](#)
- t\_RCC\_APB1ENR, [66](#)
  - \_\_pad0\_\_, [69](#)
  - \_\_pad1\_\_, [69](#)
  - \_\_pad2\_\_, [70](#)
  - \_\_pad3\_\_, [71](#)
  - \_\_pad4\_\_, [71](#)
  - \_\_pad5\_\_, [72](#)
  - BKPEN, [72](#)
  - CANEN, [71](#)
  - DACEN, [72](#)
  - I2C1EN, [70](#)
  - I2C2EN, [71](#)
  - PWREN, [72](#)
  - SPI2EN, [69](#)

SPI3EN, [69](#)  
 TIM2EN, [68](#)  
 TIM3EN, [68](#)  
 TIM4EN, [68](#)  
 TIM5EN, [68](#)  
 TIM6EN, [68](#)  
 TIM7EN, [68](#)  
 UART4EN, [70](#)  
 UART5EN, [70](#)  
 USART2EN, [70](#)  
 USART3EN, [70](#)  
 USBEN, [71](#)  
 WWDGEN, [69](#)  
 t\_RCC\_APB1Peripherals  
   RCC Interface Options, [51](#)  
 t\_RCC\_APB1RSTR, [73](#)  
   \_\_pad0\_\_, [75](#)  
   \_\_pad1\_\_, [75](#)  
   \_\_pad2\_\_, [76](#)  
   \_\_pad3\_\_, [77](#)  
   \_\_pad4\_\_, [78](#)  
   \_\_pad5\_\_, [79](#)  
 BKPRST, [78](#)  
 CANRST, [78](#)  
 DACRST, [78](#)  
 I2C1RST, [77](#)  
 I2C2RST, [77](#)  
 PWRRST, [78](#)  
 SPI2RST, [76](#)  
 SPI3RST, [76](#)  
 TIM2RST, [74](#)  
 TIM3RST, [74](#)  
 TIM4RST, [74](#)  
 TIM5RST, [74](#)  
 TIM6RST, [75](#)  
 TIM7RST, [75](#)  
 UART4RST, [77](#)  
 UART5RST, [77](#)  
 USART2RST, [76](#)  
 USART3RST, [76](#)  
 USBRST, [77](#)  
 WWDGRST, [75](#)  
 t\_RCC\_APB2ENR, [79](#)  
   \_\_pad0\_\_, [80](#)  
   \_\_pad1\_\_, [82](#)  
   \_\_pad2\_\_, [83](#)  
 ADC1EN, [82](#)  
 ADC2EN, [82](#)  
 ADC3EN, [83](#)  
 AFIOEN, [80](#)  
 IOPAEN, [80](#)  
 IOPBEN, [81](#)  
 IOPCEN, [81](#)  
 IOPDEN, [81](#)  
 IOPEEN, [81](#)  
 IOPFEN, [81](#)  
 IOPGEN, [81](#)  
 SPI1EN, [82](#)  
 TIM1EN, [82](#)  
 USART1EN, [82](#)  
 t\_RCC\_APB2Peripherals  
   RCC Interface Options, [50](#)  
 t\_RCC\_APB2RSTR, [83](#)  
   \_\_pad0\_\_, [85](#)  
   \_\_pad1\_\_, [87](#)  
   \_\_pad2\_\_, [87](#)  
 ADC1RST, [86](#)  
 ADC2RST, [86](#)  
 ADC3RST, [87](#)  
 AFIORST, [84](#)  
 IOPARST, [85](#)  
 IOPBRST, [85](#)  
 IOPCRST, [85](#)  
 IOPDRST, [85](#)  
 IOPERST, [85](#)  
 IOPFRST, [86](#)  
 IOPGRST, [86](#)  
 SPI1RST, [86](#)  
 TIM1RST, [86](#)  
 USART1RST, [87](#)  
 t\_RCC\_BDCR, [88](#)  
   \_\_pad0\_\_, [89](#)  
   \_\_pad1\_\_, [89](#)  
   \_\_pad2\_\_, [90](#)  
 BDRST, [90](#)  
 LSEBYP, [89](#)  
 LSEON, [88](#)  
 LSERDY, [88](#)  
 RTCEN, [89](#)  
 RTCSEL, [89](#)  
 t\_RCC\_CFGR, [90](#)  
   \_\_pad0\_\_, [93](#)  
   \_\_pad1\_\_, [94](#)  
 ADCPRE, [92](#)  
 HPRE, [92](#)  
 MCO, [94](#)  
 PLLMUL, [93](#)  
 PLLSRC, [93](#)  
 PLLXTPRE, [93](#)  
 PPRE1, [92](#)  
 PPRE2, [92](#)  
 SW, [91](#)  
 SWS, [91](#)  
 USBPRE, [93](#)  
 t\_RCC\_CIR, [94](#)  
   \_\_pad0\_\_, [97](#)  
   \_\_pad1\_\_, [98](#)  
   \_\_pad2\_\_, [99](#)  
   \_\_pad3\_\_, [100](#)  
 CSSC, [99](#)  
 CSSF, [97](#)  
 HSERDYC, [99](#)  
 HSERDYF, [96](#)  
 HSERDYIE, [98](#)  
 HSIRDYC, [99](#)  
 HSIRDYF, [96](#)

- HSIRDYIE, [98](#)
- LSERDYC, [99](#)
- LSERDYF, [96](#)
- LSERDYIE, [98](#)
- LSIRDYC, [98](#)
- LSIRDYF, [96](#)
- LSIRDYIE, [97](#)
- PLLRDYC, [99](#)
- PLLRDYF, [97](#)
- PLLRDYIE, [98](#)
- t\_RCC\_CR, [100](#)
  - \_\_pad0\_\_, [101](#)
  - \_\_pad1\_\_, [103](#)
  - \_\_pad2\_\_, [103](#)
- CSSON, [103](#)
- HSEBYP, [102](#)
- HSEON, [102](#)
- HSERDY, [102](#)
- HSICAL, [102](#)
- HSION, [101](#)
- HSIRDY, [101](#)
- HSITRIM, [101](#)
- PLLON, [103](#)
- PLLRDY, [103](#)
- t\_RCC\_CSR, [104](#)
  - \_\_pad0\_\_, [105](#)
  - \_\_pad1\_\_, [105](#)
- IWDGRSTF, [106](#)
- LPWRRSTF, [107](#)
- LSION, [105](#)
- LSIRDY, [105](#)
- PINRSTF, [106](#)
- PORRSTF, [106](#)
- RMVF, [105](#)
- SFTRSTF, [106](#)
- WWDGRSTF, [107](#)
- t\_RCC\_RegisterMap, [107](#)
  - AHBENR, [109](#)
  - APB1ENR, [109](#)
  - APB1RSTR, [109](#)
  - APB2ENR, [109](#)
  - APB2RSTR, [109](#)
  - BDCR, [109](#)
  - CFGR, [108](#)
  - CIR, [108](#)
  - CR, [108](#)
  - CSR, [109](#)
- t\_s16
  - Standard data types, [21](#)
- t\_s32
  - Standard data types, [21](#)
- t\_s64
  - Standard data types, [21](#)
- t\_s8
  - Standard data types, [20](#)
- t\_u16
  - Standard data types, [20](#)
- t\_u32
  - Standard data types, [21](#)
- t\_u64
  - Standard data types, [21](#)
- t\_u8
  - Standard data types, [20](#)
- Testing Applications, [58](#)
  - vAPPS\_main, [59](#)
  - vTestApp\_TestingGPIO\_main, [59](#)
- TESTING\_APPLICATION\_MAIN\_FUNC
  - APPS\_main.c, [112](#)
- TestingGPIO
  - APPS\_main.c, [112](#)
- TIM1EN
  - t\_RCC\_APB2ENR, [82](#)
- TIM1RST
  - t\_RCC\_APB2RSTR, [86](#)
- TIM2EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM2RST
  - t\_RCC\_APB1RSTR, [74](#)
- TIM3EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM3RST
  - t\_RCC\_APB1RSTR, [74](#)
- TIM4EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM4RST
  - t\_RCC\_APB1RSTR, [74](#)
- TIM5EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM5RST
  - t\_RCC\_APB1RSTR, [74](#)
- TIM6EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM6RST
  - t\_RCC\_APB1RSTR, [75](#)
- TIM7EN
  - t\_RCC\_APB1ENR, [68](#)
- TIM7RST
  - t\_RCC\_APB1RSTR, [75](#)
- TOGGLE\_BIT
  - Bit Manipulation Math Macros, [12](#)
- TRUE
  - Standard values, [23](#)
- UART4EN
  - t\_RCC\_APB1ENR, [70](#)
- UART4RST
  - t\_RCC\_APB1RSTR, [77](#)
- UART5EN
  - t\_RCC\_APB1ENR, [70](#)
- UART5RST
  - t\_RCC\_APB1RSTR, [77](#)
- USART1EN
  - t\_RCC\_APB2ENR, [82](#)
- USART1RST
  - t\_RCC\_APB2RSTR, [87](#)
- USART2EN
  - t\_RCC\_APB1ENR, [70](#)

USART2RST  
    t\_RCC\_APB1RSTR, [76](#)  
USART3EN  
    t\_RCC\_APB1ENR, [70](#)  
USART3RST  
    t\_RCC\_APB1RSTR, [76](#)  
USBEN  
    t\_RCC\_APB1ENR, [71](#)  
USBPRE  
    t\_RCC\_CFGR, [93](#)  
USBRST  
    t\_RCC\_APB1RSTR, [77](#)  
  
vAPPS\_main  
    Testing Applications, [59](#)  
VOLATILE  
    Compiler standard macros, [14](#)  
vTestApp\_TestingGPIO\_main  
    Testing Applications, [59](#)  
  
WWDGEN  
    t\_RCC\_APB1ENR, [69](#)  
WWDGRST  
    t\_RCC\_APB1RSTR, [75](#)  
WWDGRSTF  
    t\_RCC\_CSR, [107](#)